ECUC 93

European Convex Users Conference



The Metacomputing Concept: Supercomputing Power without Boundaries

PROCEEDINGS

Proceedings of the European Convex Users Conference

27 - 29 October, 1993

Bilbao, Spain

COPYRIGHT©1993 CENTRO VASCO DE SUPERCOMPUTACION (ESZ-CVS)
LABEIN Centro de Investigación Tecnológica
Parque Tecnológico Edificio 101
48016 Zamudio, Bizkaia (España)

I.S.B.N.:84-88734-01-8
Impreso en España, *Printed in Spain*

Impresión: Imprenta Berekintza, S.L. Heros, 7, 48009
Bilbao

ECUC93 European Convex Users Conference

The Metacomputing Concept: Supercomputing Power without Boundaries

Basque Country Technology Park (Zamudio), Bilbao, Spain 27-29 October, 1993

Table of contents

(14/10/93)

Wednesday, 27 October, 1993

NA **Keynote Address by**
Prof. Hans Zima
University of Vienna, Austria

"Safety Parameters System"
Jorge Peña
CSN, Spain

"Update on Language, Tool, and Scientific Library Products"
Presley Smith
Convex, USA

"Two-Dimensional Flow Simulations"
Anders H. Nielsen
Risø, Denmark

"Bringing MPP to the Masses"
Bob Masson
Convex, USA

"Standard PDEs Solvers v. Domain Decomposition Solvers in a CONVEX C-3480"
José M. Cela
CEPBA, Spain

"C-XSC: a C++ Class Library for Extended Scientific Computing"
Ulrich Kulisch
Karlsruhe University, Germany

"Superexcited Molecules: Large Memory to the Rescue"
James R. Henderson
ULCC, UK

"Metacomputing: from the Desktop to the Teraflop"
Mike Tighe
Convex, USA

"Numerical Simulation of Impact Problems"
Joaquín Martí
Principia, Spain

"Shopping for the Machine Room of the Future or Buying Meta-Computing on the Open Market "
Jerry McAllister
Michigan State University, USA

"X Terminals at CSN Network"
Mª Luisa de la Cruz
CSN, Spain

"Netwide User Data Administration System"
Elisabeth Dregger-Cappel
Düsseldorf University, Germany

"AVS: New Modules and Options"
Markus Pleier
Convex, Germany

"Contributions on Critical Aspects for Security in LANs"
Javier Areitio
Deusto University, Spain

Thursday, 28 October, 1993

"Efficient Implementation of Block-Iterative Techniques for Non-Linear Coupled Problems"
Gabriel Bugeda
Cataluña University, Spain

"The Role of Convex with Southwest Research Institute and the Upper Atmosphere Research Satellite"
Karen Birkelbach
SRI, San Antonio, USA

"European Mass-Storage Activities"
Chris Hardy
Convex, UK

"Supercomputing Aid in Neural Networks Research"
Santiago Rementería
Labein, Spain

"CONVEX—On the way to a European Corporation"
John Hughes
Convex, Europe

"Convex's Software Products for RISC Workstations"
Jon Gelsey,
Convex, USA

"File Servers—The Users' Perspective"
Jørgen Olsen
DOU, Denmark

"Expanding to New Markets—Targets and Chances"
Jim Balthazar
Convex, USA

"Mass-Storage—Status and Outlook"
Dan Henderson
Convex, USA

NA **"CONVEX C4: The Next Generation"**
Frank Baetke
Convex, Europe

"The Event Daemon Exposed"
Kevin Ashley
ULCC, UK

"Input-Queue Scheduler"
Wolfgang Stahl
DKRZ Hamburg, Germany

"The History, Present Status, and Future Plans of the Academic Computer Centre CYFRONET-CRACOW"
Marek Ksiezyk, Zofia Mosurska, Marian Noga
CYFRONET-CRACOW, Poland

"PISCIS: Interactive Processor for Nuclear Fuel Management"
Antonio Crespo
Uitesa, Spain

"COMP.SYS.CONVEX, Sharing Infromation around the World"
Volker A. Brandt
Bonn University, Germany

"Experience of Installation and Optimization of the VICTORIA Code"
Manfred Kunicke
Research Centre Rossendorf, Germany

"Info-Convex and the User Group Library"
Karen Birkelbach
SRI, San Antonio, USA

"Real-Time Simulations with Real Elements"
Thierry Benne-Darses, Anne-Claude Medina
LRBA, France

"Customer Satisfaction Issues"
Larry Abernathy
Convex, USA

NA **"CONVEX Computer Corporation—Status and Outlook"**
Jim Balthazar
Convex, USA

Friday 29 October, 1993

"The PARMACS Message-Passing Model Features and Relations to other Concepts"
Karl Solchenbach
PALLAS, Germany

"The European Parallel Competence Centre"
Ruud van der Pas
Convex, The Netherlands

"Parallel Distributed Computing in Quantum Chemistry"
Antonio Márquez
Sevilla University, Spain

"A Fair View of MPP Parallelism"
Greg Astfalk
Convex, USA

"Towards the Implementation of the LINDA Model into PVM"
Vincenzo de Florio
SASIAM, Italy

"A Library to Create a Distributed Environment through a PVM-Based Remote Function Servers"
Vincenzo de Florio
SASIAM, Italy

"Distributed Memory, Parallel Computing with Dynamic Load-Balancing on Heterogeneous Network of Workstations and a CONVEX 3210 Computer"
J. Moscinski
CYFRONET-CRACOW, Poland

"A CONVEX-HP/Workstations—Cluster Used for Algebraic Computations"
T. Friedrich, and W. Laßner
Leipzig University, Germany

"Scalable Parallel Processing"
Steve Wallach
Convex, USA

EUROPEAN CONVEX USERS CONFERENCE

Wednesday, 27 October, 1993

SAFETY PARAMETERS SYSTEM

J. Peña, J. R. Alvarez, J. Garralón
Consejo de Seguridad Nuclear, Madrid, Spain

ABSTRACT

Nuclear power plant data are received at the Operations Center of the Consejo de Seguridad Nuclear in emergency situations. In order to achieve the required graphic interface a Safety Parameters System (SPS) is being developed. The system will run in a UNIX environment and use the Xwindow capabilities. The received data are stored in such a way that it can be easily used for further analysis (simulation and forecasting) and training activities.

INTRODUCTION

The Spanish Consejo de Seguridad Nuclear (CSN) is using the SCADA WINDOWS[1] system to acquire data from nuclear power plants in the event of an emergency at any plant. SCADA WINDOWS provides a direct real-time transfer of data from licensee plant computers to the CSN Operations Center.

The arrived data are validated and stored in a VAXserver 3100. The SCADA WINDOWS runs on a VAXstation 3100 and provides a user interface based on tables. A general overview of the hardware layout is presented in Figure 1. The components of the system are connected via DECNET and both hardware and software are redundant to increase reliability.

SCADA WINDOWS is a program that alerts for emergency situations, provides information on communication lines status and displays parameters values.

Although SCADA WINDOWS is being used, it offers a poor graphical capability to plot data on a screen. Additionally, simulation computer codes are not implemented in the above mentioned VAX environment and, therefore, integration of the received plant data with the CSN analysis methods is not available.

The Operations Center has access to a CONVEX C220, which is the mainframe supporting the intensive calculation and most of the computational tools at the CSN.

In order to achieve the required graphic interface and prepare data to perform simulation and forecasting with already existing computer codes a Safety Parameters System (SPS) has been developed for the CONVEX environment.

Plant data will be transferred from the VAX to the CONVEX environment via two serial port connections between terminal servers (redundancy will be always preserved).

SPS will collect, prepare and plot that data in a X terminal screen. Usual graphical capabilities like parameter selection, multi-curve plots and monitoring will be available.

GENERAL DESCRIPTION

The system will consist of task-oriented modules (processes) which will communicate each other using well known UNIX mechanisms (signals, pipes, sockets and shared memory segments). SPS will conceptually have two different parts:

Data reception
The system will automatically activate data reception when an emergency situation in a nuclear power plant is established. Data transmitted will be raw stored and prepared/expanded to match the requirements of the data monitorization programs.

Data monitorization
It comprises:

- A selection menu showing plants in emergency.
- A data driver. It access data, at user request, and send them to representation.
- A data visualization module. It represents graphically data received.

Figure 2 shows a general outline of the system. A more detailed description of each part follows.

DATA RECEPTION

Two different lines are transmitting simultaneously the same data. One of these lines is considered to be the main line, the other being the secondary.

A specific process is full time dedicated to the following tasks:

- Data reading from sockets.
- Error detection.
- Data store in files and shared memory.

An ASCII coded string is received from nuclear power plants every 30 sec. In case of emergency the string contains 1 Kbyte (approximately, depends on the plant) of plant data, otherwise a 14 bytes health message is received.

Initially that specific process reads a file containing general information for controlled power plants (generic names for further related files, number of parameters to be received), opens a disk file and creates a shared memory segment, both for every plant. Once an emergency situation is established and data are transmitted, these steps follow:

- Data are checked against predefined patterns so some errors can be detected. When that happens data are rejected and fetched from the secondary transmission line, which, in turn, is checked.

- Validated data are sequentially stored on a specific file (history file). This serves as backup file and can be used as data source for further analysis.

- The same data are loaded on a shared memory segment replacing former data. This area is intended to keep the most recently data arrived. Further processes requiring this type of data will only need to link this area.

Shared memory is designed to hold not only arrived data but also information obtained from other means, such as:

- New foreseen parameters, not arrived through communication lines, whose status is known by other means.

- Parameters calculated from arrived data. Criteria used to perform these calculations are set in a specific file which is read when the process starts execution.

A continuous cumulative copy of shared memory onto a disk file is done. This file will be used by the visualization processes to plot the whole sequence.

DATA MONITORIZATION

As already mentioned, data monitorization involves three different programs:

- PSM (Plants State Menu)
  Is a separated program invoked by each user who wants to monitor the system. It displays a menu showing a list of all nuclear power plants and their current state and, when requested, activates a PDD program for a particular plant.

- PDD (Plant Data Driver)
  It is the SPS component which accesses plant data and processes them to be displayed by ICI. It uses CANAL functions for the communication with ICI.

- ICI (IsoVu Canal Interface)
  This program is used by PDD as graphic interface. It is created with IsoVu functions.

This modularity has permitted to utilize already existing software and facilitate current and future developments.

A more detailed description of these programs follows.

PSM

This independent program can be invoked as many times as users need to monitor safety parameters. The corresponding process accesses shared memory to know about the current state of every plant. The program uses the IsoVu Library to open and manage a window with push buttons in the X window environment. It displays an interactive graphic menu when started.

The graphic menu consists on a table of all active nuclear power plants of Spain. Each plant has an associated press button that can be activated by the "mouse" pointer. When the button is activated, a summary of the transmission state is presented if the associated plant is in normal state, or the PDD program for this plant is invoked to monitor all the data values received for this plant if it is in emergency state.

This program is structured with an initialization part, a main continuous loop and some IsoVu callbacks. Firstly, the graphic file for PSM is read and IsoVu functions are initialized. The main loop checks the current state of each central, seeking the values in the shared memory. The IsoVu callbacks are assigned to push buttons in the graphic menu and, when invoked, they launch a new PDD process for the corresponding nuclear plant.

The names of the plants in graphic menu are of different colours depending of the state of the plant. They can change when an emergency event occurs in any plant to inform the user which is currently working with this program.

PDD

PDD is the SPS component which processes nuclear plant parameters. It uses the ICI interface through CANAL functions to communicate with the user. It reads several files (configuration, predefined plots) and the current data from nuclear plant stored in the shared memory. Different processes can be simultaneously monitoring the same plant emergency.

This process transmits and receives two kinds of variables to and from ICI: data variables and control variables. Data variables are assigned parameters received from the plant. The control variables are internal variables which have assigned the options selected by the user in the graphic menus, for example, when requesting new plots or asking for more information on safety parameters.

The program initializes all variables, reads the configuration file for the specific plant, executes through CANAL functions the program ICI and starts the main loop to send and receive the new incoming safety parameters from the plant.

The names and order of management for the variables, windows and options are stored in the configuration file. There is a configuration file for every nuclear plant in the SPS. The parameter properties are stored in a database file for each plant containing the name, identification, description, units, range, etc.

The ICI program deals with all problems related to graphic display and user interaction. The program PDD sends to ICI the names of first window and of the corresponding variables for display.

While the new data are received from the plant, PDD sends the values of parameters and receive the new manual data following the ICI protocol for communication.

ICI

ICI[2] is an interactive interface between a data source program and the user through the graphical environment provided by IsoVu functions and libraries. It presents sequentially, in graphical form, the values of incoming variables and obtains values from the user. The transmission between ICI and PDD is made through CANAL functions.

The ICI process is a child process of PDD (master) program. The communication protocol allows the master program to control the creation of new windows, operations with variables, etc.

Independently, graphic files have been designed with the IsoVu Display Builder. These files are read by IsoVu functions to display the primitive objects and operations. It is possible to create and edit these graphic files without any recompilation of the ICI program. The master program must send to ICI the names of the graphic files and objects contained in them.

After initialization ICI program waits for user inputs or events until it receives a signal requesting control to accept a new transmission. When that happened it receives the control vector from master program. The control vector contains a "next command" field which determines the next action to be taken by ICI. A normal action is the reception of the output values for actual variables from master. After this action the ICI program sends to the master the input values stored for user interactions. The cycle repeats until a special end action is taken by ICI or PDD program.

ICI is a general program that supports different levels of running depending on the degree of sophistication of master programs. At the lower level it is possible to have implicit names, auto-scales, permanent input or output variables and transmission of all values every time. This level of running is easier to manage and does not require much control from master program. The higher level allows explicit names, prefixed scales, dynamic input or output

variables, creation and deletion of windows and transmission of values by list of indexes of changed variables only. For each of above mentioned of levels of running there is a mode control in ICI.

UTILITIES

CANAL

CANAL[3] is a set of functions designed to communicate father and children programs through the Unix socket pairs. The functions provide the following features:

- Simplification of the socket operations, such as read, write, open, close, blocking, error and hang up detection and large buffers control.

- Child processes management through internal tables indexed by an integer value.

- Synchronous or asynchronous transmissions.

- Arrangement of messages in items of different type.

In the terminology of this interface there is an abstract concept, *the canal*, bidirectional way of communication between two programs. A *canal* is a whole structure that contains information on:

- the underlying socket,
- the related program,
- the name of the associated file,
- the blocking mode, and
- the state of the transmission.

These functions take care of the problems and errors in transmissions, and provide buffers for long messages (greater than page size).

When a *canal* is created, a child program with user supplied arguments is activated and a socket pair is opened. A *subcanal* must be created if a new socket with a previously activated child program is necessary. The *canal* has a data type associated with it. This data type defines the length of every item transmitted by the socket. The read or write operations are made on arrays supplied by the user. All CANAL functions return an error code in case of failure of the operation. Some error conditions generate a message to *stderr* explaining the reason of the error. Only when a *canal* is open it is possible to send or receive data through the associated socket.

Canals can be partially or totally closed. If a *canal* is partially closed the socket is shutdown and no more communication is allowed through this *canal*, however it is possible to open other *subcanals* with the same child program. Total close of a *canal* implies the end of all transmissions and subsequent opening of *subcanals* with the same program is not allowed.

ISOVU

IsoVu[4] is an interactive software system which allows the user to create colour graphics displays containing both dynamic input and output facilities. IsoVu can be logically divided into the following elements:

Display Builder
It is a tool which enables the user to design and construct customised displays.

Application Builder
It provides an interactive facility to define the dynamic behaviour of graphic displays constructed by the Display Builder.

Programmer's Library
It provides a flexible toolkit with which data from the user's application can be combined with the display constructed in the Display Builder, to produce a dynamic interface.

CONCLUSION

The Safety Parameter System has been designed to provide, in emergency situations, comprehensive access to information derived from nuclear power plants in Spain. Nuclear plant control systems send data values of selected safety parameters through the established communication lines. This information, when received, is checked and transmitted to the CSN mainframe, where it is available for the monitor processes.

The information treatment is spread over several processes which communicate each other using UNIX mechanisms.

An important part of the information required to run SPS is resident in ASCII files so it can be easily changed or updated. This method avoids code reprogramming and recompiling for a new use or situation.

The structure of the ICI program and of the CANAL functions set make possible the connection of SPS with other programs.

PRESENT STATUS AND FUTURE DEVELOPMENTS

A basic graphical interface for all spanish nuclear power plants is already available. At the present time it provides:

- Monitorization of the more representative current parameters.
- Multi-curve plots of selected parameters from a list.

It is foreseen to add new capabilities to the system, like:

- Visualization of predefined plots from a catalog.
- Visualization of plots representing a user-defined combination of parameters.
- Introduction of data known by another means.

Moreover, connection to simulators and training activities are aspects under consideration at CSN.

REFERENCES

[1] *SCADA WINDOWS. Sistema de adquisición de datos para comunicaciones*, ELIOP S.A., Avda. de Manoteras s/n, 28050 Madrid.

[2] J. Ramón Alvarez, *ICI. IsoVu Canal Interface*, INF-ICI-92/01, CSN (1992)

[3] J. Ramón Alvarez, *CANAL. Set of Functions to Communicate Programs through the Unix Socket Pairs*, INF-CANAL-92/01, CSN (1992)

[4] *ISOVU Software System*, Isograph, Lloyds House, 22 Lloyd Street, Manchester M2 5WA, United Kingdom.

Figure 1. General Hardware Configuration

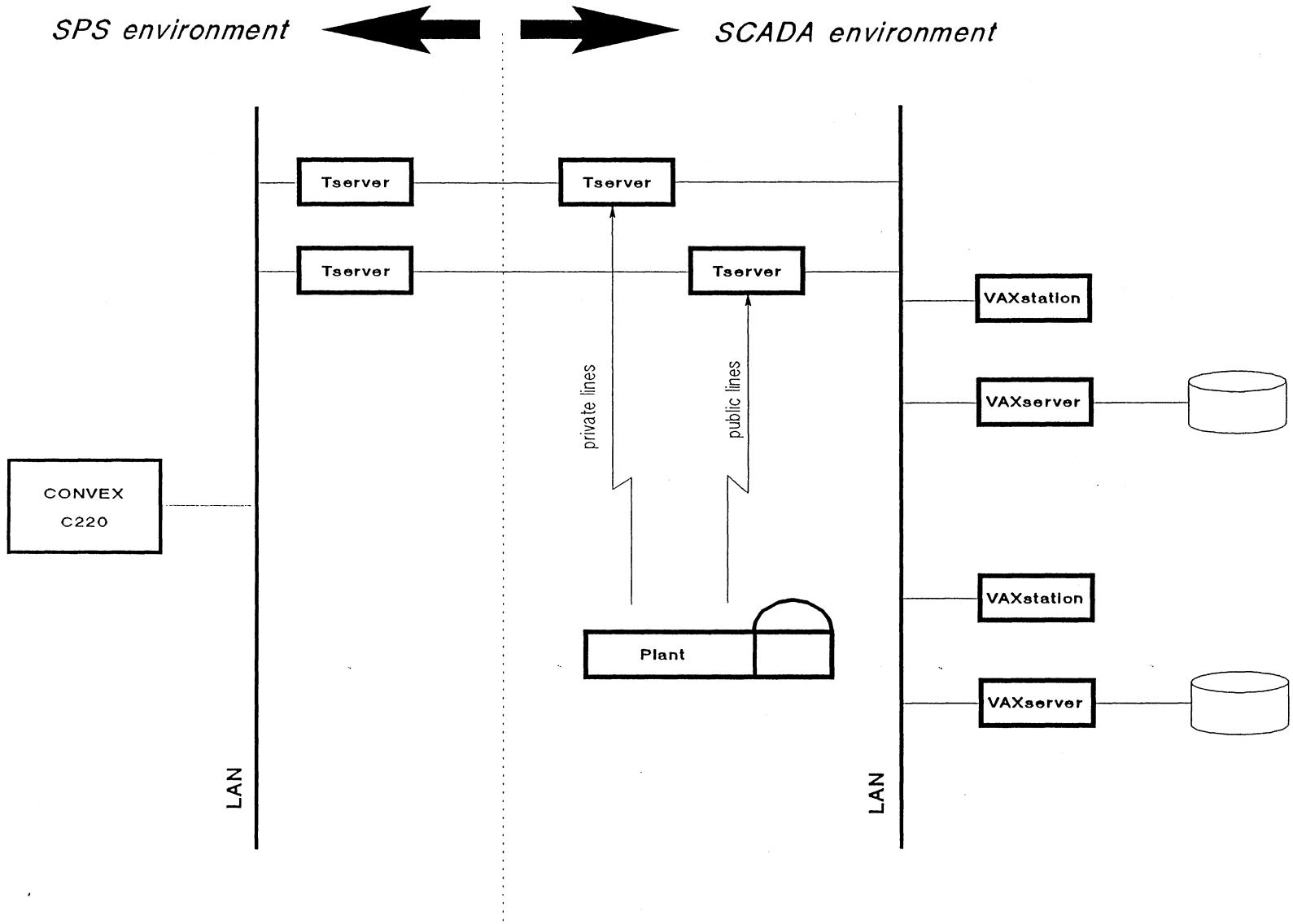
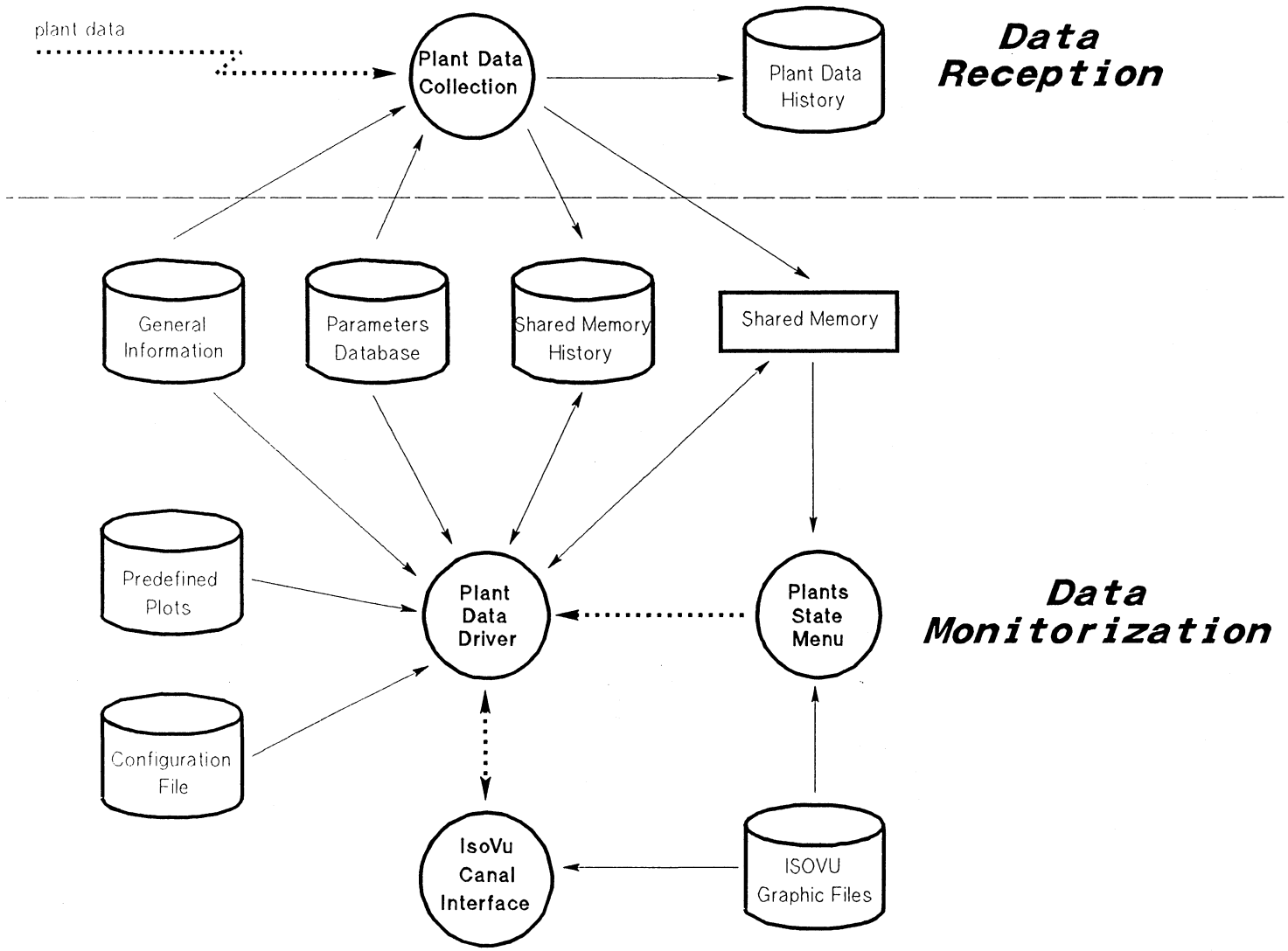


Figure 2. Safety Parameters System Overview

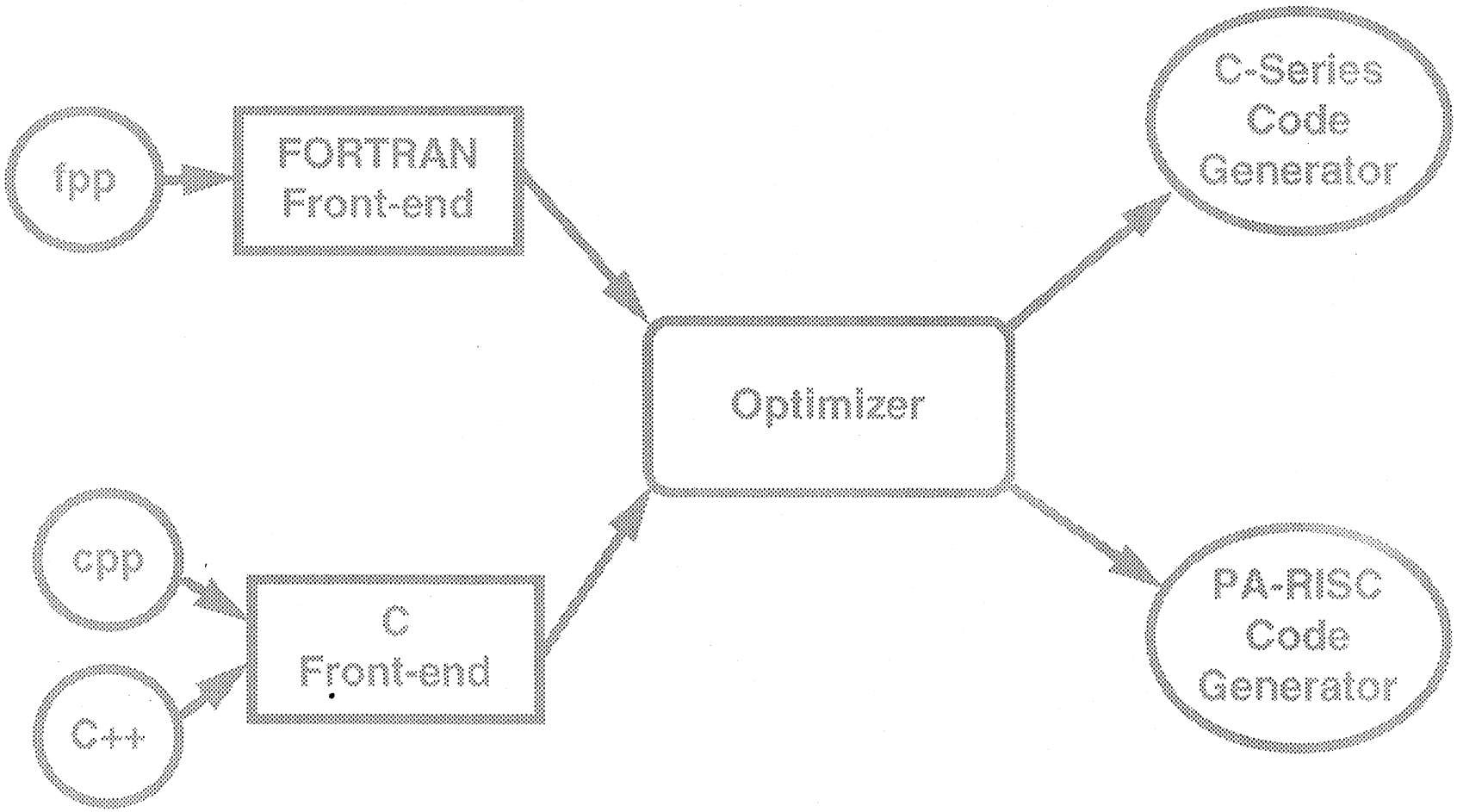


Development Software Update



- Agenda
  - Compilers
  - Tools
  - Scientific Libraries
  - Directions

CONVEX Compiler Family



C4 Compiler Enhancements



- **Machine Dependent Optimizations**
  - Support for new instructions and new registers
  - Enhanced dependency analysis capabilities
  - Construct induction values for r+r indexing
  - C4 vector scheduling and bank conflict optimization
  - Store reordering
  - Integer div changed to multiply/add/shift
  - Improved “a” and “s” register selection
- **Code Generator Improvements**
  - New peephole optimizer
  - Optimized COMPLEX data type instruction generation
- **New Machine Independent Optimzations**
  - Parallelization of loops with calls that don’t require synchronization with the Application Compiler

Application Compiler Enhancements



- **Application Compiler**
  - Improved dependency analysis
  - Automatic parallelization of loops with calls
- **Augmented Dependence Analysis**
  - Compute direction and distance vectors
  - Improved interchange support
  - More accurate dependence information maintained
- **Array Section Analysis**
  - Performed while determining the side effects of procedure calls
  - Describes the geometry of accesses to an array
- **Automatic Parallelization of Loops with Calls**
  - Uses the results of array section analysis to determine the side-effects of procedure calls
  - Does not incur the code space overhead of inlining

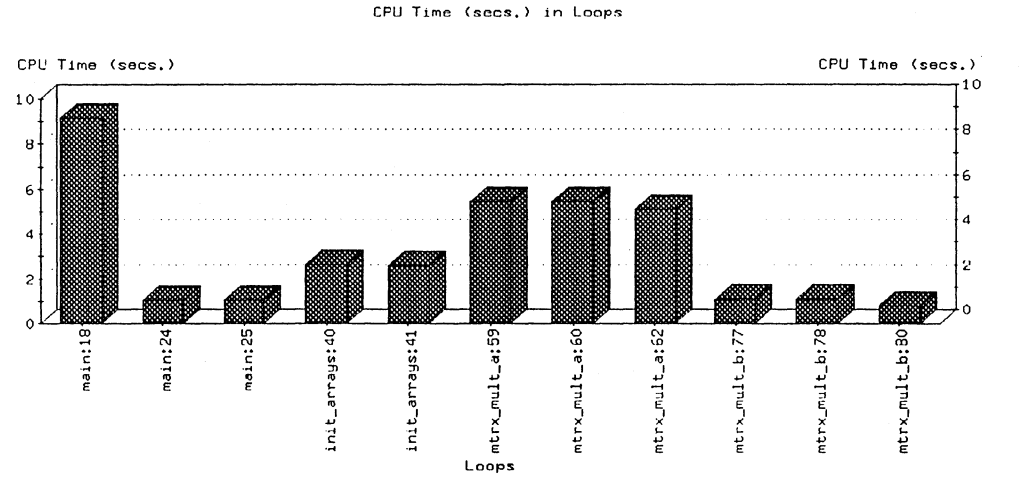
Tools



- **CXdb V3.0 New Features**
  - Integrated with the Application Compiler
  - Navigation in the presence of moderate to large numbers of threads
- **CXpa V3.0 New Features**
  - Support for C4 performance monitor
    - » FLOPS
    - » Number of cache misses
    - » Number of vector load or store operations
  - More performance metrics across all regions
    - » Wall clock time
    - » Process virtual time
  - Bar charts with source clickback

CXpa Example

mtrx_loop.pdf



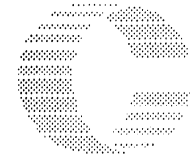
Customize...

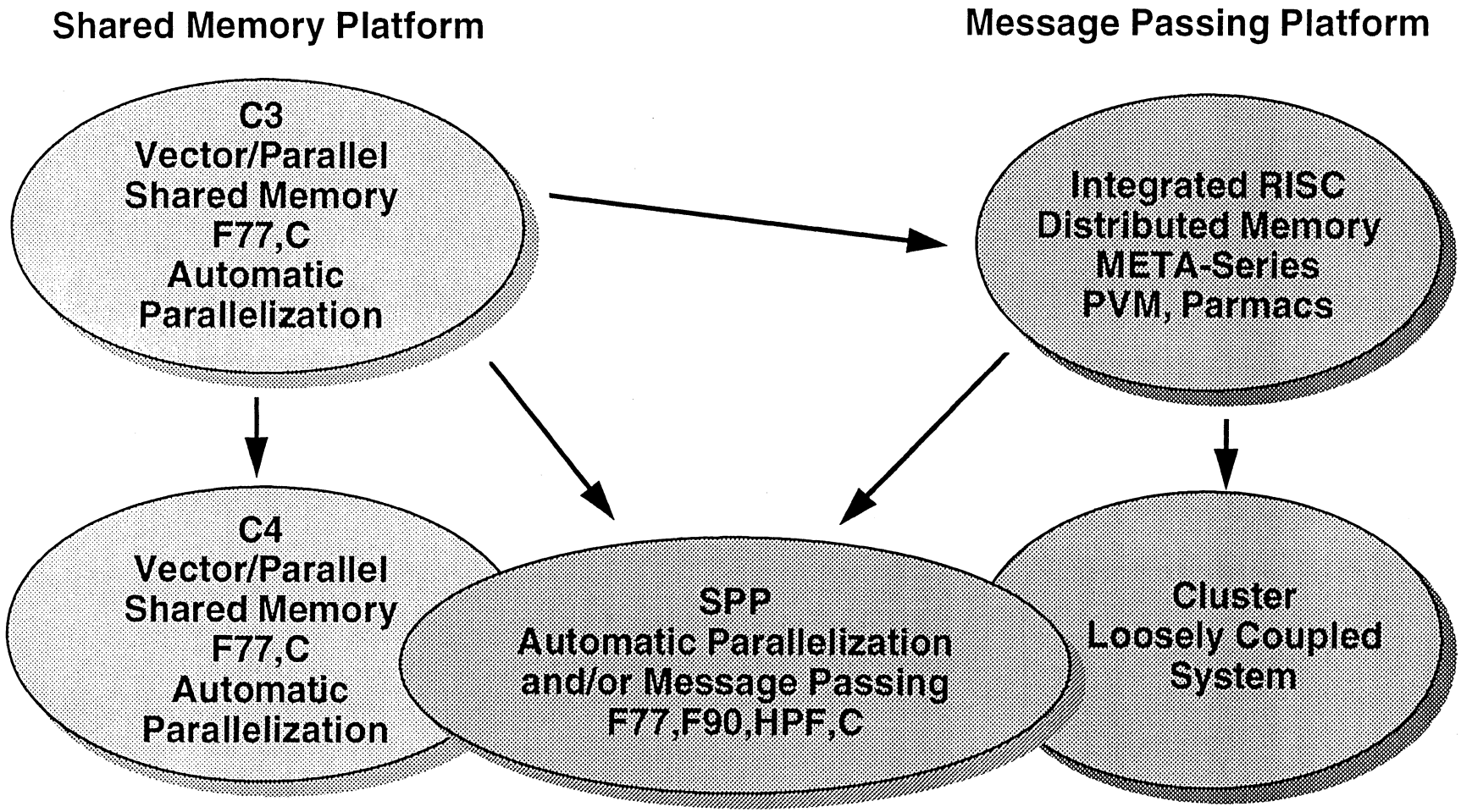
Source Code File : mtrx_mult.c

```
    50
    51  mtrx_mult_a(a, b, c, base
    52  int a[SIZE][SIZE] ;
    53  int b[SIZE][SIZE] ;
    54  int c[SIZE][SIZE] ;
    55  int base ;
 R  56  {
    57     int i, j, k ;
    58
 1  59     for (i=0; i<base; i++)
 1  60         for (j=0; j<base; j++) {
    61             c[i][j] = 0 ;
 1  62             for (k=0; k<base; k++)
    63                 c[i][j] += a[i][k] * b[k][j] ;
    64         }
    65  }
    66
    67
    68  void mtrx_mult_b(a, b, c, base)
```

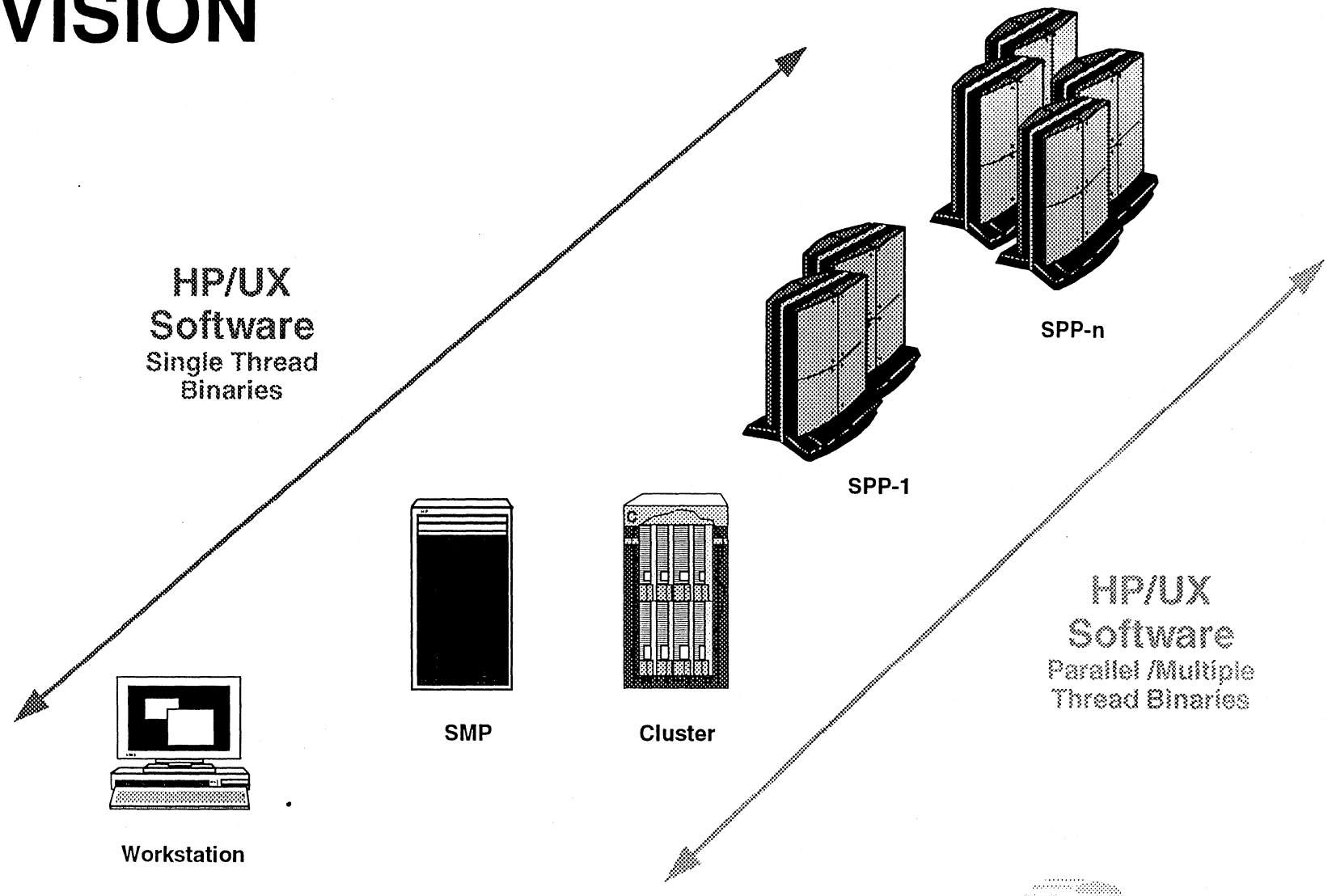
CONVEX

SPP - Platform for Programmers





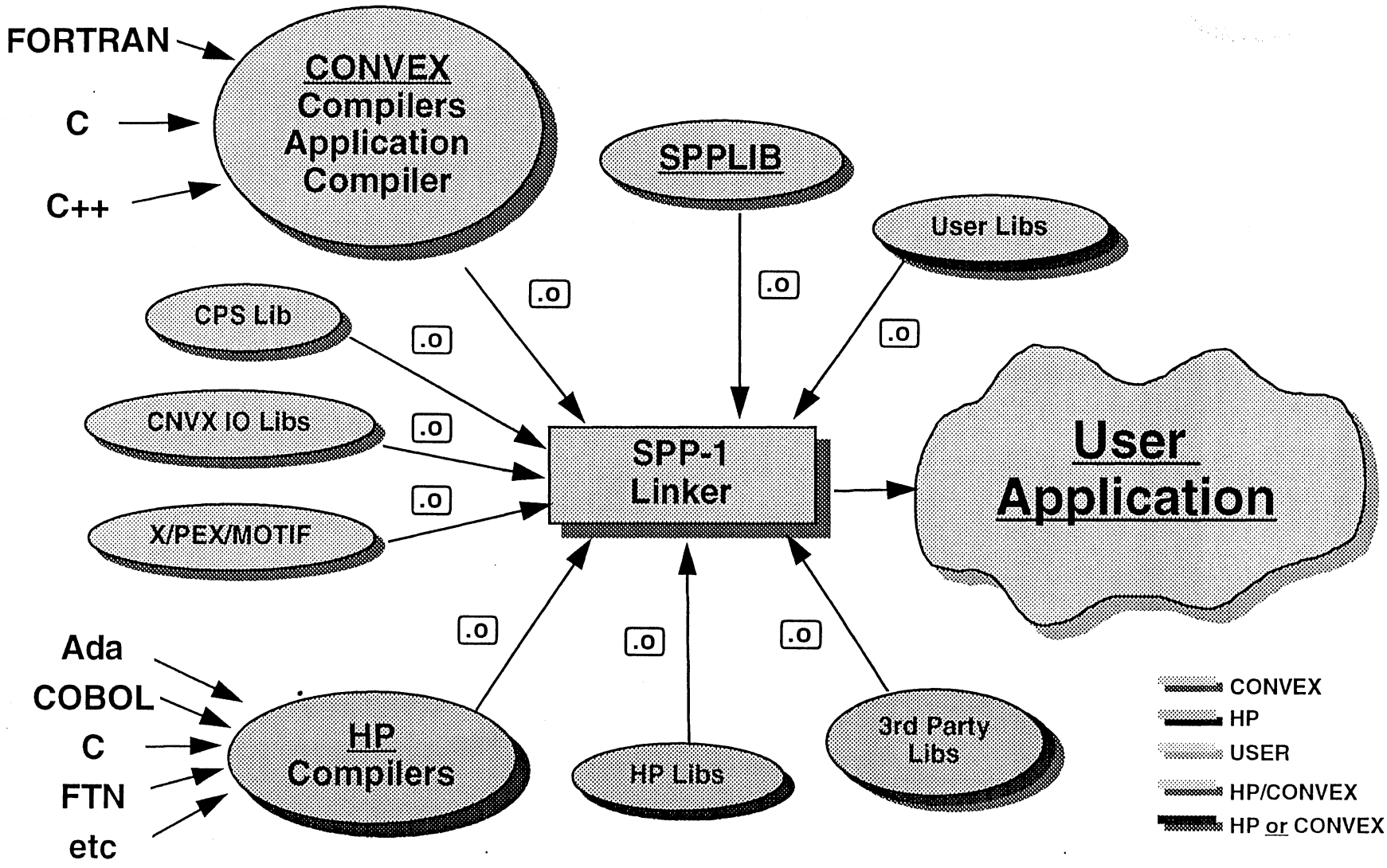
VISION



HEWLETT PACKARD

CONVEX

HP Interoperability



SPP Programming Model



- Primary Focus -- Ease of Use
  - Geared toward porting existing applications
  - High degree of automatic parallelism detection
- Globally Shared Distributed Virtual Memory
- Support for Explicit Message Passing
  - For porting existing MPP and cluster system codes
  - ORNL PVM V3.1 based
- Support for Explicit Synchronization
  - Allows for barrier and gate (lock) synchronization
  - Supports explicit critical sections
- Allows the Sophisticated User to Control the Machine
  - Memory hierarchy exposed via directives
  - Two level explicit parallelism

SPP-1 Compiler Specifics



- **Procedure Compilers**
  - High performance PA-RISC code generator
  - Leverage off vector technology for improved data locality and cache reuse
  - Improved HP compatibility
- **PA-RISC Code Generator**
  - PA-RISC specific instruction scheduling
  - Global register allocation
  - Software prefetch and pipelining
- **Data Locality Optimizations**
  - Loop blocking
  - Loop unroll and jam
- **Application Compiler**
  - Automatic parallelism of loops with calls

Development Methodology



- Leverage HP technology where possible
- Focus on Optimization
  - This has been the compiler focus since day one
  - First vectorization, then small scale parallel, now massively parallel
- Focus on Code Generation in Initial Release
- Continuous Improvement
  - Vectorizer/Optimizer rewrite
  - Vector/Parallel restructuring
  - New dependency analysis phase
- Keep the Defect Count Low
  - Compilers need to be production quality from day one
  - Strong software development methodology
  - Leverage the C-Series for testing of new code where possible

SPPLIB Optimization Levels



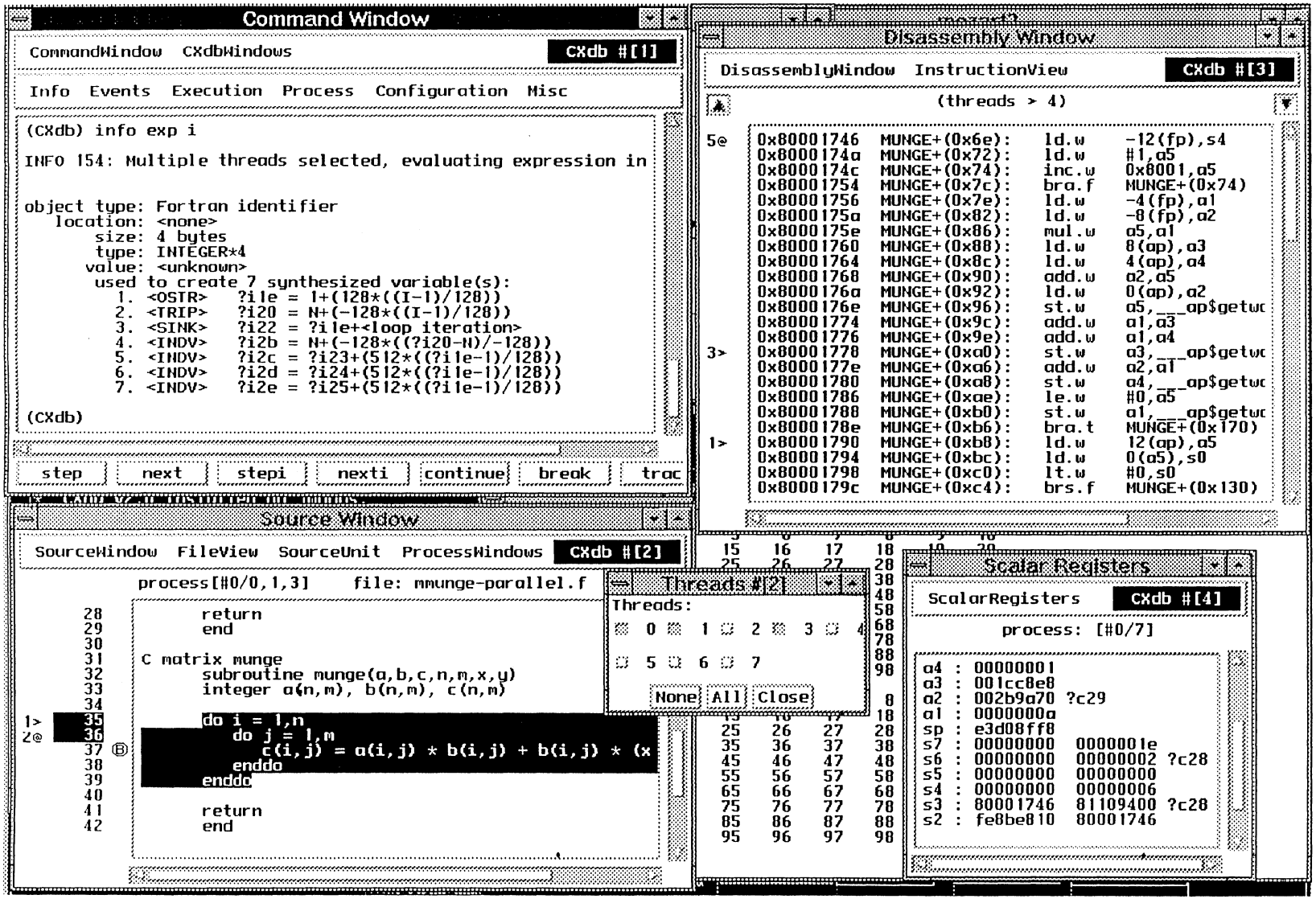
- **Single Processor Loop Level**
  - Loop unrolling
  - Assembly coding
  - Instruction scheduling
- **Single Processor Algorithmic Level**
  - Data locality in FORTRAN
  - Loop blocking and unrolling
- **Single Node**
  - Parallel FORTRAN with CPS library support
- **Multiple Nodes**
  - Data parallel FORTRAN with CPS library support
  - Utilizes programming model constructs

SPPLIB V1.0 Functionality

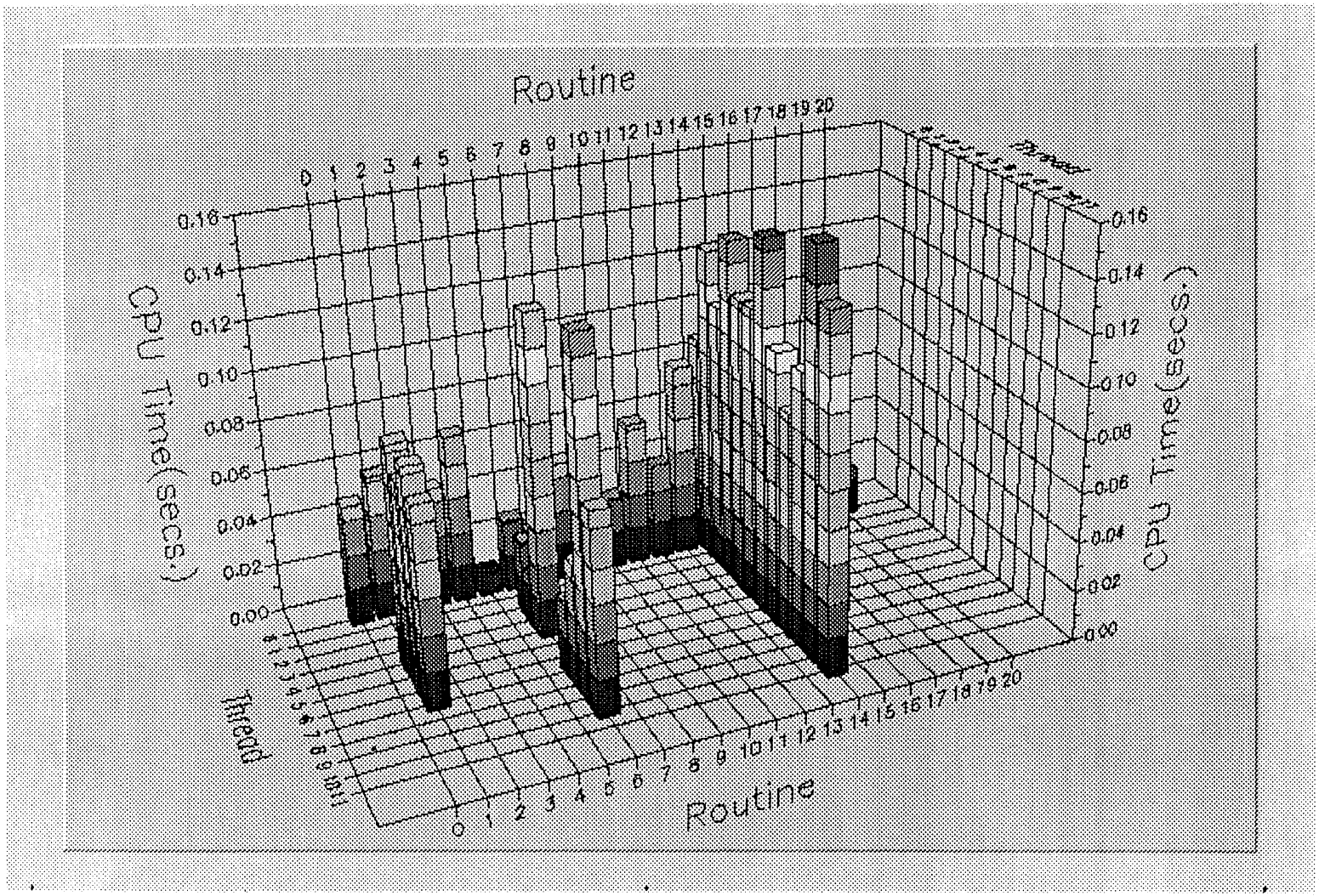


Library Name	Title	Multi-node Parallel	Single-Node Parallel	Single CPU Opt	ASM Opt
VECLIB	BLAS 1	None	None	Some	Some
VECLIB	BLAS 2 & 3	Some	Some	Some	Some
VECLIB	LINPACK	Some	Some	Some	Some
VECLIB	EISPACK	Some	Some	Some	None
VECLIB	Skyline	None	None	None	None
VECLIB	FFTs	All	All	All	Some
VEBLIB	Corr and Conv	All	All	All	All
VECLIB	Recurrences	None	None	None	None
VECLIB	Miscellaneous	None	None	None	None
LAPACK	All	Some	Some	Some	Some
SCILIB	All	Some	Some	Some	Some

CXdb Example



CXpa Example

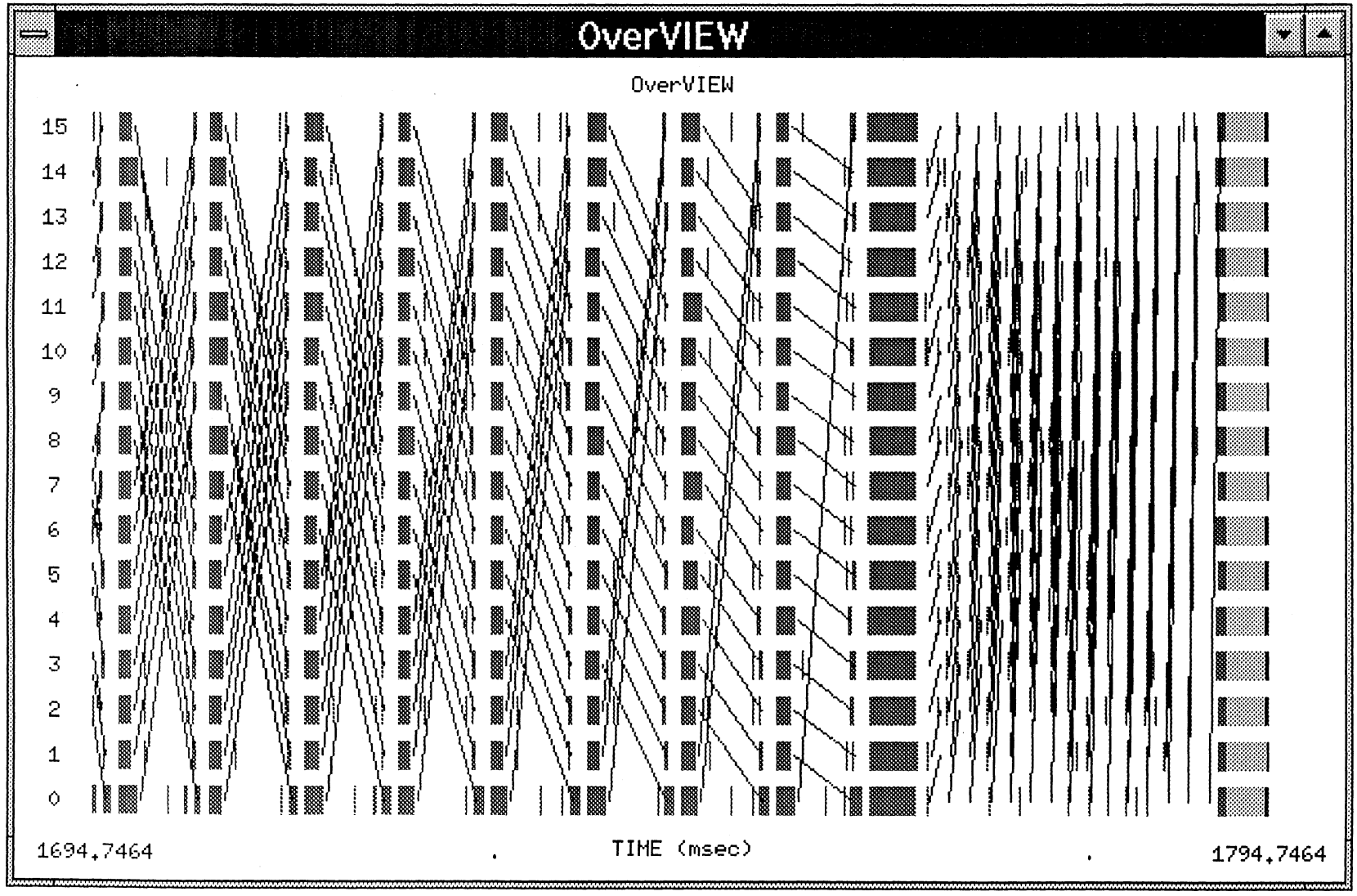


AIMS Features



- Trace based profiling tool
- Provides complete profiling environment
  - Automatic instrumentation of source code
  - Library routines for logging events
  - Trace visualizer
- Can correlate trace data to source code
- Features
  - User can define and display states
  - User can select which events are to generate data
  - Loop and routines can be automatically instrumented
  - Support both C and Fortran

AIMS Example



TWO DIMENSIONAL FLOW SIMULATIONS

E.A. Coutsias[1], J.P. Lynov, **A.H. Nielsen**, M. Nielsen, J. Juul Rasmussen and B. Stenum
Association EURATOM Risø National Laboratory, Optics and Fluid Dynamics Department, DK-4000 Roskilde, Denmark
[1]Department of Mathematics and Statistics, University of New Mexico, Albuquerque, NM 87131, USA

Introduction

Coherent vortical structures play an important role in the time evolution of large-scale geophysical flows and have been studied increasingly over the past decades. In the oceans and in the atmosphere, vortices are abundant; the high and low-pressure areas on weather maps are in fact huge eddies embedded in the global zonal currents. Large-scale eddies are also present in virtually all parts of the worlds oceans. Oceanic eddies are observed to have large horizontal dimensions, 10 to 100 km, typically, and to have long life times, in some cases even a couple of years [1]. It is obvious that these vortices play an important role in the transport of physical properties such as momentum, heat, salt and biochemical components. Under typical atmospheric and oceanic conditions the two-dimensionality of the motion on larger scales is governed by the planetary rotation or the density stratification, or a combination of these (in addition to the dynamical restrictions imposed by the 'flatness' of the domain itself). Two dimensionality can also be established in other ways, for example by external magnetic-field [2] or by enclosing the flow in an essentially two dimensional domain. Such situations are met in magneto-hydrodynamic flow, in plasmas and in accretion disks of neutron stars. The understanding of these coherent structures are thus not only relevant to geophysical flows, but also to other fields of physics, such as plasma physics and astrophysics.
The common structure is a monopole with swirling velocity of the same sign everywhere. The best known of all monopolar structures is probably Jupiter's great red spot. In this presentation we will examine a special kind of coherent structures, dipoles, and their interaction with solid (noslip) boundaries. A dipole consist of two closely packed counter rotating vortices, locked to each other, enclosing the fluid inside them. These kinds of structures are interesting in view of their non-zero linear momentum transporting them over large distances compared to their own radius. So even though dipoles were created far from boundaries their self-propelling properties will eventually bring them close to boundaries and interact with them. At Risø we have investigated such interactions both experimentally and numerically. The experiment consists of a rotating water tank, where the dipoles are created by injecting a jet of water. These dipole are made to collide with obstacles of different shapes. Having obtained an experimental set of data the parameters, such as speed and dipole size, were transferred into a spectral code implemented on Risø CONVEX C220 computer. Here it is possible to examine the interaction in detail and for a large range of dipole speeds and different kinds of obstacles. The numerical and experimental results will be presented in form of a video.

Numerical setup

A two dimensional flow can, in this context, be well described by the two dimensional Navier Stokes equations, which consist of the momentum equation

$$\frac{\partial \vec{v}}{\partial t} + (\vec{v} \cdot \nabla)\vec{v} = \nu \nabla^2 \vec{v} - \frac{1}{\rho}\nabla p \quad , \tag{1}$$

and the continuity equation

$$\nabla \cdot \vec{v} = 0 \quad . \tag{2}$$

Here $\vec{v} = (u, v)$ is the flow velocity, ν the kinematic viscosity, ρ the density and p the pressure. These equations describes the time evolution of a incompressible flow. The form of eq. (1) and (2) is not suitable for solving numerically as we have two differential equation to handle, one for each coordinate. Instead we take the curl of eq. (1) to obtain the vorticity equation

$$\frac{\partial \omega}{\partial t} + (\nabla \psi \times \hat{z}) \cdot \nabla \omega = \nu \nabla^2 \omega \quad , \tag{3}$$

where $\omega \hat{z} = \nabla \times \vec{v}$ is the vorticity, and ψ the stream function, defined as: $\vec{v} = \nabla \psi \times \hat{z}$, related to the vorticity by the Poisson equation

$$\nabla^2 \psi = -\omega \quad . \tag{4}$$

Eq. (3) and (4) is a set of nonlinear differential equations and analytical solutions can only under special conditions be found. Generally one have to turn to numerical methods to follow the dynamical evolution of such flows. At Risø we have developed a spectral code, where the solution to the vorticity equation is built up as a superposition of individual waves (or modes), see e.g. [3] and [4]. Such spectral codes are very accurate compared to e.g. codes based on the finite difference methods.

$$\begin{aligned} \omega(x,y,t) &= \sum_{n=0}^{N} \sum_{m=0}^{M} \omega_{nm}(t) \exp(inx) \arccos(m(\cos y)) \\ \psi(x,y,t) &= \sum_{n=0}^{N} \sum_{m=0}^{M} \psi_{nm}(t) \exp(inx) \arccos(m(\cos y)) \end{aligned} \quad , \tag{5}$$

i.e. we are expanding the flow in Fourier modes in the x-direction and Chebyshev polynomials in the y-direction. For a given distribution of $\omega(x,y,t)$ and $\psi(x,y,t)$, the coefficients, $\omega_{nm}(t)$ and $\psi_{nm}(t)$, are found by a Fast Fourier Transformation (FFT). Inserting these expansion into eqs. (3) and (4) we can follow the time evolution of each of the modes. The second term on the left hand side of eq. (3) will become a convolution sum and cannot be evaluated directly. Instead the two functions, $\nabla \psi \times \hat{z}$ and $\nabla \omega$, are transformed into real space by the FFT, where the multiplication are performed in a grid and the result is then transformed back into mode space. These transformations are very expensive in computer time, for our case 70 % of the total run time is used for that, and set a limit of how many modes we can employ for expanding the flow. An efficient FFT is hereby crucial for such kind of numerical investigations.

In our numerical investigations we have used two different geometries, a two dimensional periodic channel where the flow is confined between two solid planes, and an annular geometry where the flow is confined between two concentric cylinders. In both cases the use of Chebyshev polynomials ensures the enforcement of the no-slip boundary condition. We have implemented these computer codes on our local CONVEX C220 computer and a typical run time is of the order of 1 - 2 days.

Experimental setup

The experiment consists of a square water tank with a flat bottom and side walls of length 1 m. Into this tank various cylindrical wall structures can be placed. The tank is filled with water to a level of 15 cm and brought into solid body rotation with an angular frequency of 0.63 s-1. A dipole is created by injecting a jet of water colored with dye into the mid-plane of the water for a period of 5 s. Our setup is similar to that described by Flierl et al. [5], who, however, did not investigate dipole-wall interactions. After a short transient period following the jet injection, the solid body rotation causes the flow to be very nearly two-dimensional as described by the Taylor-Proudman theorem.

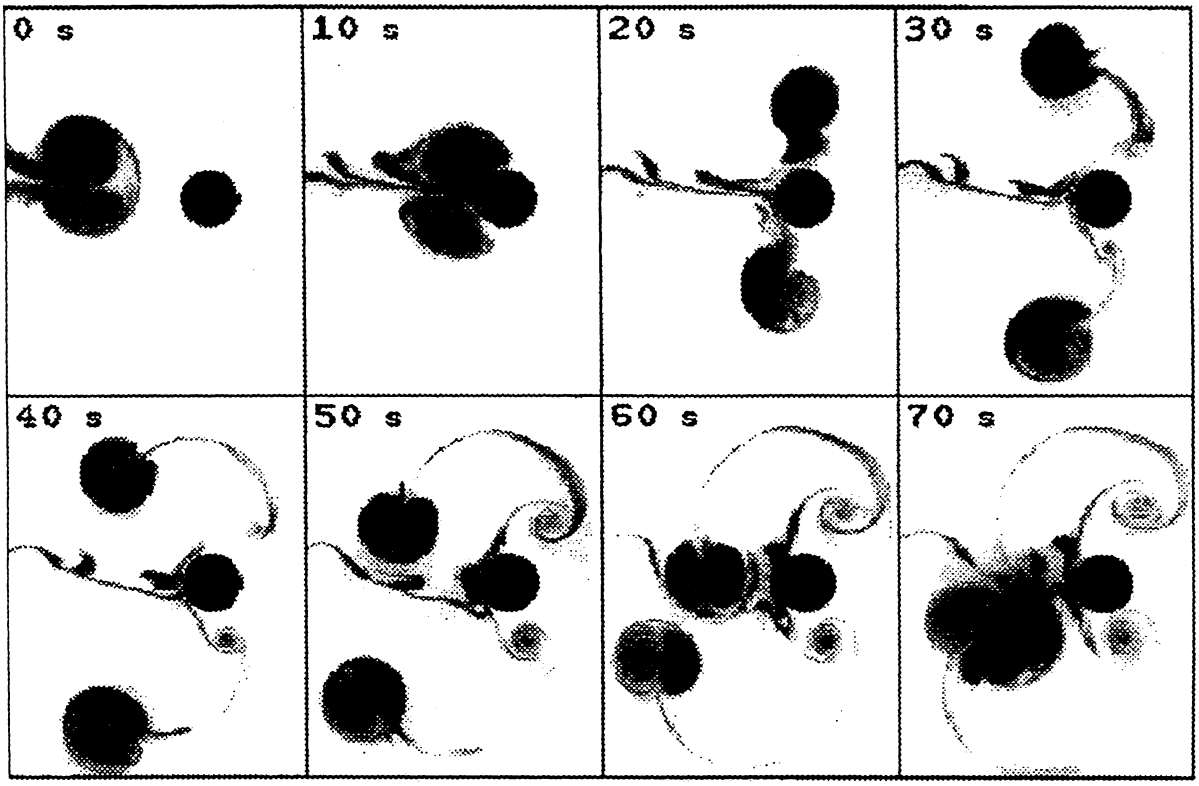


Figure 1: Experimental result showing a dipole interacting with a cylinder.

Results

In our studies, a dipole was made to collided with straight as well as with concave and convex curved walls. An experimental example of this is shown in Figure 1, where a symmetric dipole is interacting with a cylinder. Figure 2 show the corresponding numerical simulation where we have used an initial condition fitted to the experiment. We summarize here the main aspects of this interaction. As the dipole approaches the wall its velocity field will generate an opposite-signed wall vorticity layer in order for the flow to satisfy the no-slip boundary condition. These thicken due to viscosity. At time=10 these wall layers has rolled tightly into secondary vortices which couple with the opposing primary lobes. The primary dipole splits into two new dipoles, strong enough to leave the cylinder, time=20. These new dipolar structures do not have balanced amounts of positive and negative vorticity and consequently they have curved trajectories. At time=60 they have performed half a circle and collide with each other in front of the cylinder. As the original collision was not a head on collision the two dipoles will not perform a symmetric path. The consequence is that at time=70 somewhat complicated structures are formed.

Figure 2 shows a computer simulation of the experiment in Figure 1. The dipole initialized here is a socalled Lamb dipole with a linear relationship between the vorticity and the stream function. As seen in the figure the overall dynamics is the same as in the experiment with the exception of the time-scale. We believe that this it is due to inaccuracy in determining the experimental values of the speed and size of the dipole. Having an accurate computer code it is possible to examine the flow in detail such as calculating the total amount of vorticity generated on the cylinder and examine a larger range of speeds and sizes of the dipoles. Hereby the computer simulation and experiment supplement each other.

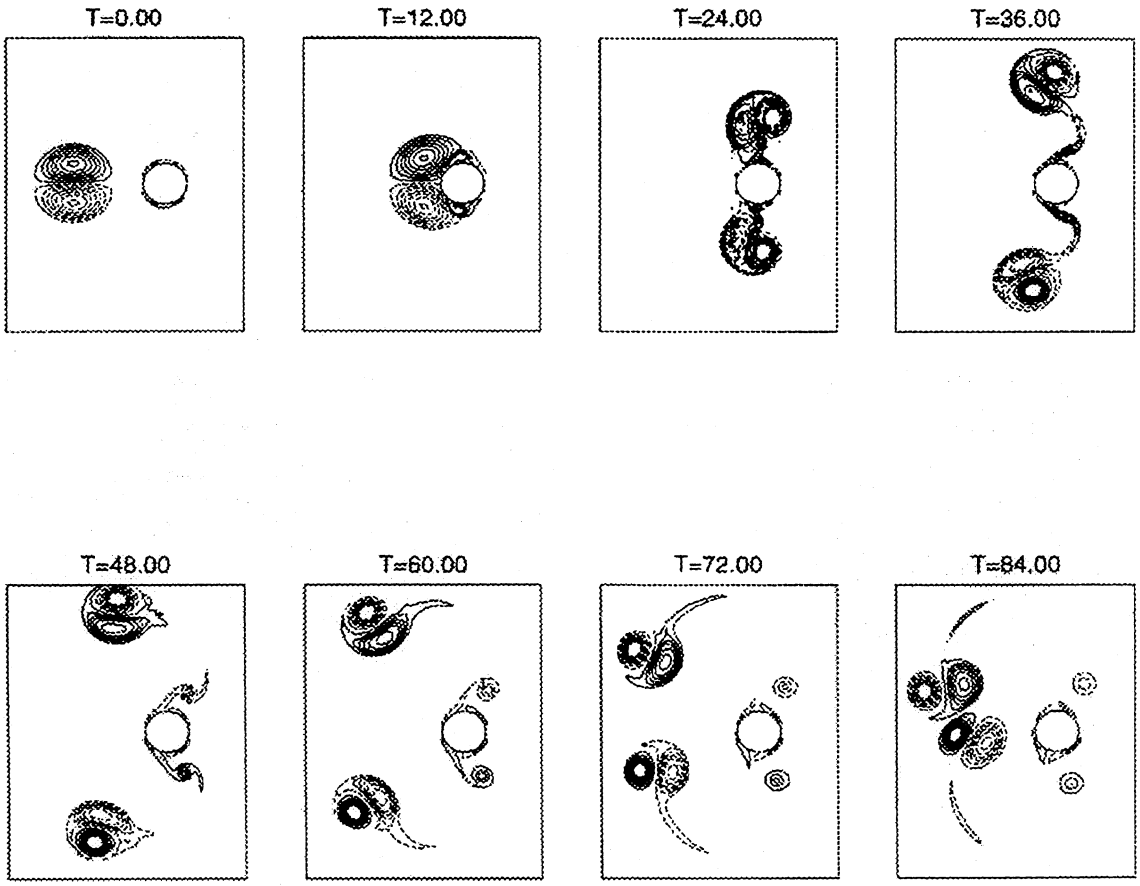


Figure 2: Numerical simulation for the same setup as the experiment shown i Figure 1.

References

[1] A.R. Robinson, "Eddies in marine science", Springer Verlag, Berlin, 1983 609 pp.

[2] T. Huld, A.H. Nielsen, H.L. Pecseli and J. Juul Rasmussen, "Coherent structures in two-dimensional turbulence", Phys. Fluids B **3**, 1991 p. 1609,

[3] E.A. Coutsias and J.P. Lynov, "Fundamental interaction of vortical structures with boundary layers in two-dimensional flows", Physica D **51**, 1991 p. 482.

[4] D. Gottlieb and S.A. Orszag, "Numerical analysis of spectral methods; theory and applications", SIAM CBMS-NSF reg. conf. ser. in Appl. math. vol 26 (SIAM, Philadelphia 1977).

[5] G.R. Flierl, M.E. Stern and J.A. Whitehead, "The physical significance of modons; Laboratory experiments and general integral constraints", Dyn. Atmos. Oceans, **7** 1983 p. 233.

"Bringing MPP to the Masses"

Bob Masson
Convex Computer Corp

The Promises of MPP

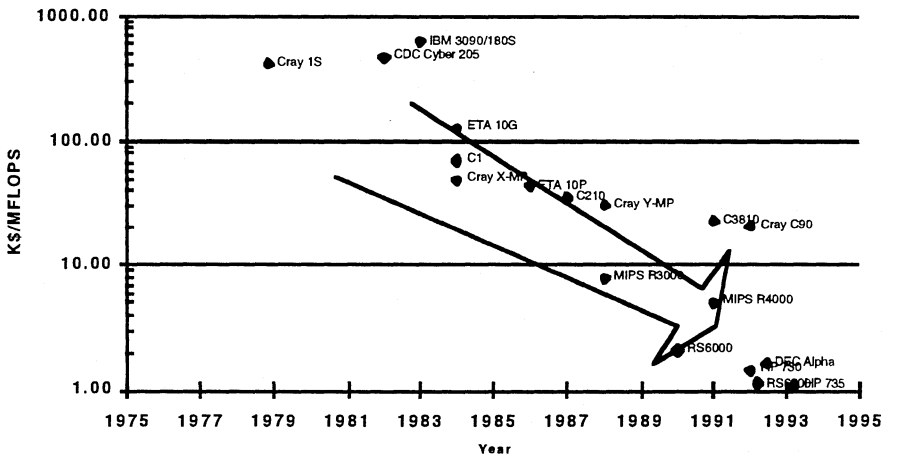
"**Analysts believe that MPP will affect the computer industry in the same way that microcomputers did in the 1980s...**"
- New York Times June 15 1992

"**Supercomputers are entering a new era which may involve the gradual disappearance of the machines in favor of massively parallel processing (MPP) computing equipment...**"
- IEEE Spectrum Sept 1992

"**[Oracle Corp. CEO Larry] Ellison says that massively parallel processing will vastly reduce the cost of current computational technologies.**"
- Los Angeles Times August 5 1993

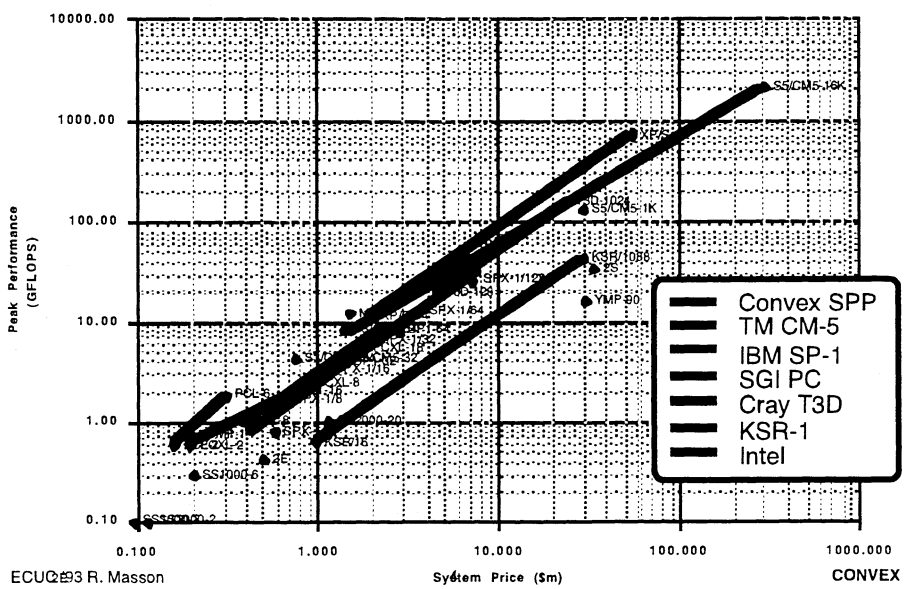
Price/performance ($/MFLOPS)

- Promise is gigaFLOPS for nanoPESETAS
- Improvements in price/performance for 100x100 LINPACK



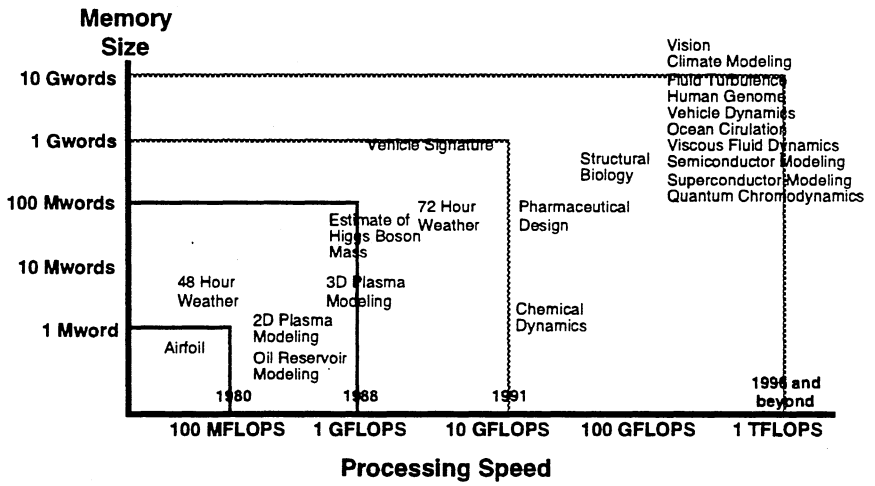
Scalability (and expandibility)

- Scalability means consistent price/performance



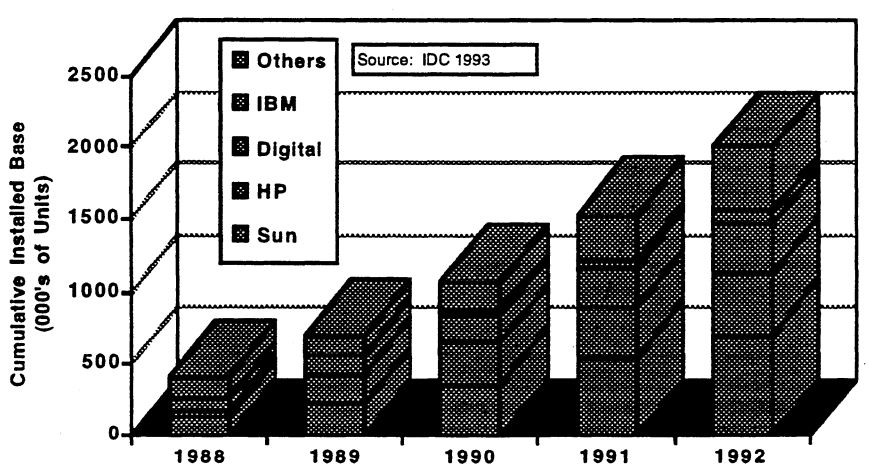
TeraFLOPS and Terabytes

- The promise of extremely high performance
- The U.S. "Grand Challenges"



Open Systems

- Promise is consistent user interfaces and admin tools (e.g. COSE)



Worldwide Workstation Installed Base, 1988-1992

The Pitfalls of MPP

"In Supercomputing, Superconfusion"
- Business Week March 22 1993

"Massively parallel processor systems cannot match the performance of a vector supercomputer..."
- Cray Research VP Sept 1992

"While some hardware is available to US businesses, the costs are still astronomical, and there is an insufficient amount of software to make the most of the technology"
- IEEE Spectrum Sept 1992

The Programming Pitfall

- Programming Models
- Language Standardization

PVM?

Interconnect Topology?

Hypercube?

FORTRAN 66?

Bisectional Bandwidth?

SIMD vs MIMD?

Toroid?

HPF?

Distributed vs Shared Memory?

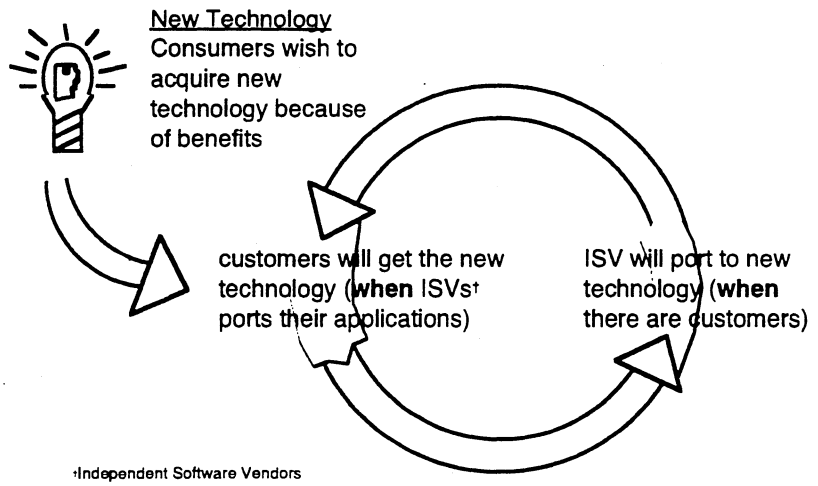
C++?

Fortran 90?

FORTRAN 77?

Applications Availability ("Catch-22")

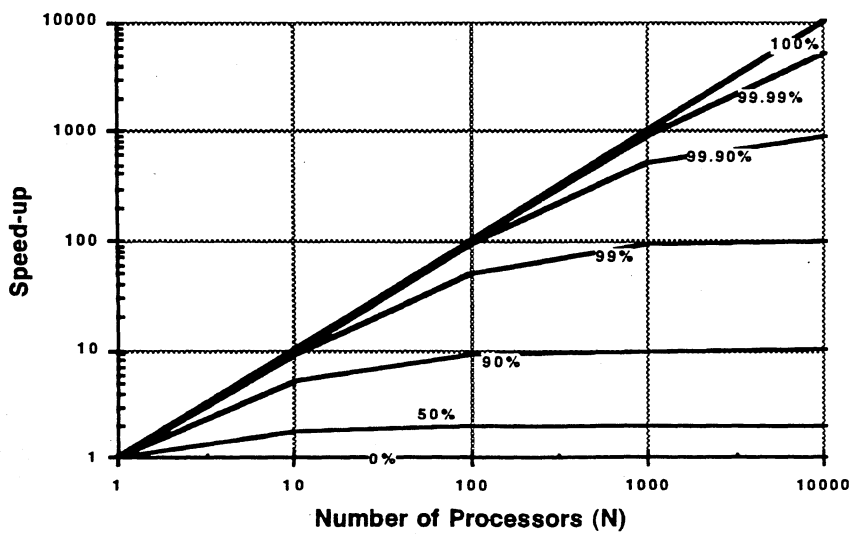
- **Threre's a natural barrier to porting applications**



†Independent Software Vendors

Limitations of Parallelization

- **It takes a high degree of parallelization to use 000's processors**



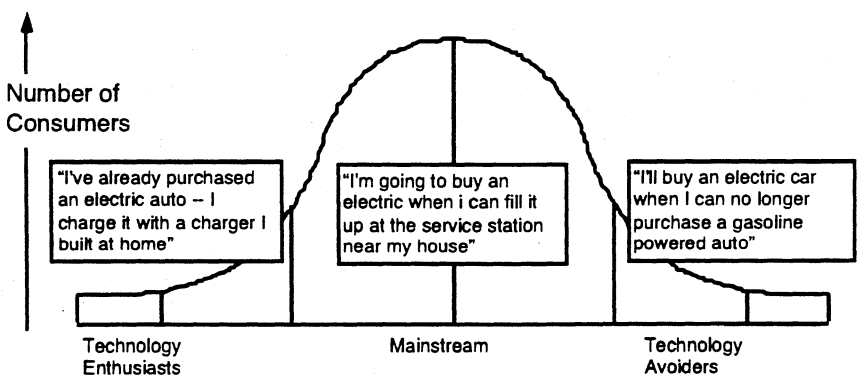
Promises vs Pitfalls

- **Consumer must weigh promises against pitfalls**
  - the mainstream customer will adopt when the benefits outweigh the risks; the risks will be reduced as time goes on and the innovators iron out the bugs
- **Mainstream customer is in production environment**
  - So the task of "bringing MPP to the mainstream customer" involves living up to as many of the promises of MPP as possible, while solving, circumventing, or hiding as many of the pitfalls as possible.

Promises	Pitfalls
• Price/performance	• Programming models
• TFLOPS and Tbytes	• Architectural standards
• Scalability	• Limits of parallelization
• Expandibility	• Language standardization
• Open Systems	• Applications availability

The Mainstream Customer

- **Accepts proven technology**
- **Demands an entire solution -- not just the technology**

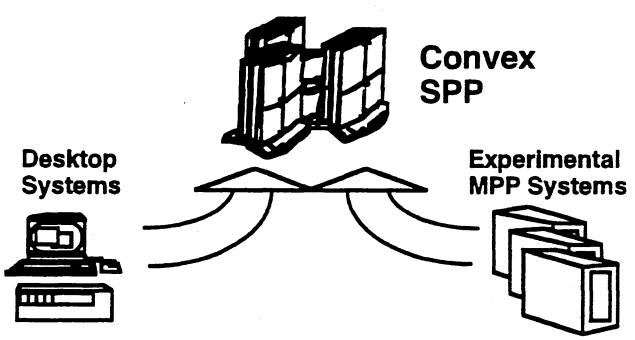


The Convex MPP system

- **Is called SPP**
- **Because**
  - the term MPP is ambiguous today (massively parallel? moderately parallel? masochistically parallel?)
  - the Convex implementation is a superset of traditional MIMD MPP systems
  - the programming model of the machine lends itself to being viewed as a number of MP implementations;
  - it is geared towards the mainstream "production" customer, not just MPP experimenters
  - it is scalable, and it is parallel
- **Is geared toward the mainstream customer**

Convex SPP system roots

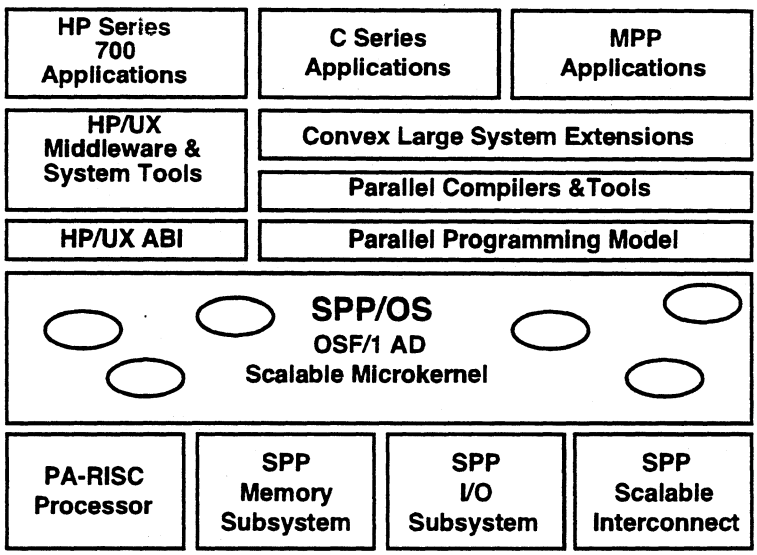
- **Combination of workstation programming model and MPP architecture**



- **Familiar programming model**
- **Common Open Software Environment**
- **Thousands of applications**

- **Massively parallel architecture**
- **Scalable CPU performance**
- **Large physical memory**
- **High bandwidth I/O**

SPP-1 Operating System

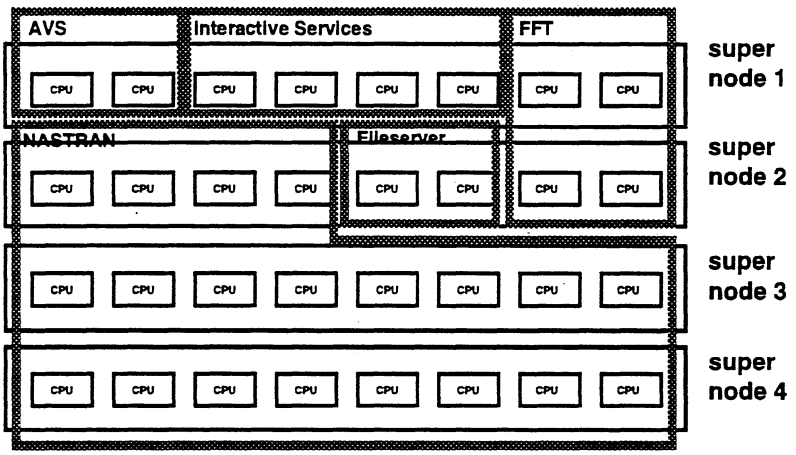


SPP-1 Operating System

- **Scalable distributed microkernel technology**
  - OS functions distributed throughout complex
  - OS resources scale with system performance
- **HP/UX Application Binary Interface (ABI) compatible**
  - Common Open Software Environment (COSE)
  - Provides over 4,500 HP/UX applications
- **Sophisticated processor partitioning**
  - Allows jobs to run in sub-complexes that match the job's parallelization requirements
  - Can be dynamically changed to match user load and application mix
- **Multiuser**
  - Multiple users can access machine simultaneously
  - Runs single-threaded editors, compilers, electronic mail and so on simultaneously with parallel jobs

SPP-1 Process Scheduling

- CPUs organized into sub-complexes
- Each sub-complex separately scheduled
- Allows most efficient use of processors



Automatic Parallelizing Compilers

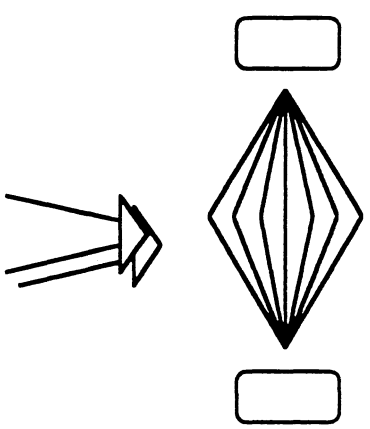
Automatic Parallelization

- Convex C Series C & F77
- Cray CFT
- Sun UNIX F77
- DEC VMS FTN
- IBM FORTRAN
- HP/UX F77
- Fortran 90
- C
- C++

User-directed Parallelization

- Compiler directives
- Fortran 90
- HPF
- PCF synchronization
- PVM

Parallelized Applications



SPP Programming Model

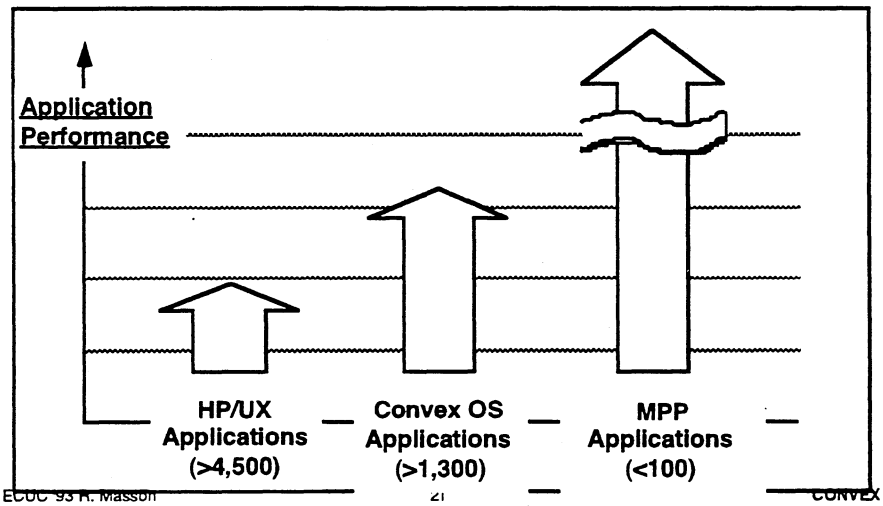
- **Primary focus -- ease of use**
  - Geared toward porting and parallelizing existing applications
  - C, ANSI C and C++
  - Full FORTRAN 77 with DEC, Cray, Sun, and HP extensions
  - HPF and Fortran 90
- **Global Virtual Distributed Shared Memory**
  - Key to ease-of-use for existing programs
  - No need for major mind-set shift
  - Increases productivity for application development & porting
- **Automatic and Explicit Parallelization**
  - Shared memory parallelization via compilers
  - No need to convert FORTRAN 77 (or 66!) to Fortran 90
  - Message passing parallelization via directives and libraries

SPP Programming Model (Cont'd)

- **Support for explicit message passing**
  - For porting existing MPP codes
  - High performance message passing library based on hardware and OS primitives
  - Supports Parallel Virtual Machine (PVM) standard
- **Support for explicit synchronization**
  - Parallel Computing Forum (PCF)-like synchronization primitives
  - Allows for barrier, sequence, and post/wait synchronizers
  - Supports explicit critical sections
- **Allows the sophisticated user to control the machine**
  - Memory hierarchy exposed via directives
  - Two-level explicit parallelism
  - Explicit data distribution as defined by HPF

Applications Availability

- **HP/UX Applications Binary Interface compatibility**
- **Automatic parallelizing compilers**
- **EasyApp development environment**

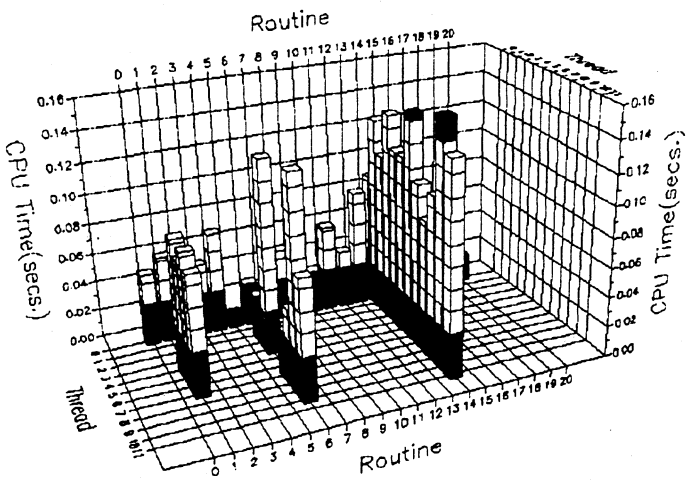


Application and Porting Environment

- **EasyApp: An application development and porting environment dedicated to increasing productivity of application developers**
- **Features:**
  - Application development and debugging tools
  - Application performance enhancement tools
  - Extensive use of GUIs (graphical user interfaces)
  - Supports development and porting of both shared memory and message passing parallelism
  - Uses built-in hardware system performance monitors
- **Leverages Convex's history of lead in compiler and development tools technologies**
- **Geared specifically towards quickly generating MPP applications**

EasyApp Thread Profiler

- Points out parallel routine performance
- Highlights areas for further optimizations



Summary

- The Convex SPP-1 is the next generation MPP
- Geared toward the production-oriented user
- A combination of MPP technology and workstation programming model

STANDARD PDEs SOLVERS v.s. DOMAIN DECOMPOSITION SOLVERS IN A CONVEX C-3480

José M. Cela, Albert Pujolar, Juan J. Navarro
Departamento de Arquitectura de Computadores
Universidad Politécnica de Cataluña

Abstract:

In this paper we shall consider experimental results on a CONVEX 34080 vector multiprocessor for the solution of algebraic linear systems of equations. Such systems arise from the discretization of a Partial Differential Equation in Fluid Dynamics, using the Finite Element Method. We consider non-symmetric matrices. In this paper we explain parallel algorithms for the solution of such systems. Specifically we consider some iterative Krylov subspace methods with a RILU(k) preconditioner, and a Domain Decomposition method called Modified Schur Complement.

Key words: Iterative methods, ILU preconditioners, domain decomposition.

1. Introduction

In this paper we shall consider experimental results on a CONVEX 3480 vector multiprocessor for the solution of algebraic linear systems of equations arising from the discretization of a Partial Differential Equation using the Finite Element Method.

$$Ax = f \qquad \text{with } x, f \in R^N \qquad (1)$$

We consider non-symmetric matrices which are extremely sparse. Direct solution methods based on Gaussian elimination or LU factorization of A produce fill-in. Thus, the extra storage requirements and computational load are unacceptable.

Unlike direct methods, iterative solution methods do not suffer from fill-in because they never modify the A matrix. Only the product of the matrix by a vector is needed. For this reason, these methods also have a high degree of parallelism. The most popular iterative methods are Krylov subspace methods like GMRES(k) [Saad86], CGS [Son89], Bi-CGstab[Van92]. The convergence of this methods must be usually improved by the use of a proper preconditioner.

Incomplete factorizations are powerful general purpose preconditioners. However, they become the bottleneck in the parallelization of an iterative method, because they need to solve sparse triangular systems and this algorithm has a few degree of parallelism. Different orderings of the unknowns have been proposed [Duff89] in order to obtain a trade-off between the maximum degree of parallelism in the sparse triangular factors and the best approximation to A^{-1}.

In the last years a lot of attention has been paid to Domain Decomposition techniques. The main reason is that these methods give a natural way to extract a medium grain parallelism in the solution of PDEs. There are different Domain Decomposition approaches. Here we are interested in a non-overlapped Domain Decomposition method. This approach decomposes the domain into a set of non overlapped subdomains. Then the internal nodes of each subdomain are ordered the first and those on the interfaces the last, the matrix will have a 2x2 block form as follows.

$$\begin{bmatrix} A_{II} & A_{IB} \\ A_{BI} & A_{BB} \end{bmatrix} \begin{bmatrix} x_I \\ x_B \end{bmatrix} = \begin{bmatrix} f_I \\ f_B \end{bmatrix} \qquad (2)$$

where: x_I are the unknowns associated with the internal subdomain nodes and

x_B are the unknowns associated with the external subdomain nodes

If internal nodes of a subdomain are ordered consecutively before starting with the next subdomain, then A_{II} is a perfect block diagonal matrix, where each block is associated with one subdomain. Then the A matrix has an arrow structure. The system (2) can be reduced by block Gaussian elimination to:

$$Sx_B = \tilde{f}_B \qquad \text{with} \qquad \begin{aligned} S &= A_{BB} - A_{BI}A_{II}^{-1}A_{IB} = A_{BB} - \sum_{k=1}^{n} A_{Bk}A_{kI}^{-1}A_{kB} \\ \tilde{f}_B &= f_B - A_{BI}A_{II}^{-1}f_I = f_B - \sum_{k=1}^{n} A_{Bk}A_{Ik}^{-1}f_{Ik} \end{aligned} \qquad (3)$$

where n is the number of subdomains. S is called *Schur Complement Matrix*, and in general, this matrix is dense. We can solve the Schur system in order to obtain the unknowns associated with the interfaces, x_B, then the unknowns associated with the internal nodes, x_I, will be calculated by a parallel backward substitution.

An iterative method for solving $S\, x_B = \tilde{f}_B$ can be used because only matrix by vector multiplication involving S are required. Clearly, to multiply S by a vector only requires n solvings, one per subdomain, which can be done in parallel, if we use the implicit formula of S. The effectiveness of the iterative used method depends on keeping small the number of steps through a preconditioning, which is considerably less expensive to construct and apply than S^{-1} itself.

2. Some Krylov subspace methods

There are several iterative methods for solving (1), but the three main classes are:

(i) Methods based on the normal equations. For example LSQR [Saun88].

(ii) Orthogonalization methods. For example GMRES [Saad86]

(iii) Biorthogonalization methods. For example CGS [Sonn89] or BiCGstab [Van92]

In this paper we are focus in orthogonalization and biorthogonalization methods. Specifically we are interest in GMRES, GCS and BiCGstab.

GMRES build a sequence $\{x_n\}$ with

$$x_n \in x_0 + \{r_0, Ar_0, \ldots, A^{n-1}r_0\}$$

satisfying that the norm of the residual in the step n is minimum over all vector in $\{r_0, Ar_0, \ldots, A^{n-1}r_0\}$. This is accomplish forcing the orthogonality condition

$$r_n \perp \{Ar_0, \ldots, A^n r_0\}$$

In the original formulation of the method at each step of the method a new vector need to be stored. In the i-th step a orthogonalization involving i-th vectors is made. Of course this is a prohibitive amount of memory and computation. A practical implementation of the method restart after k steps for some k parameter. This implementation of the method is usually called GMRES(k).

CGS is an improvement of the Biconjugate Gradient (BiCG) algorithm [Flet75]. Again a sequence of vectors

$$x_n \in x_0 + \{r_0, Ar_0, \ldots, A^{n-1}r_0\}$$

is build, but now the orthogonality condition imposed is

where $\tilde{r}_0$ is a vector such as $\langle r_0, \tilde{r}_0 \rangle \neq 0$, usually $\tilde{r}_0 = r_0$. The main advantage of this method is that the two

$$r_n \perp \{\tilde{r}_0, A^t\tilde{r}_0, \ldots, A^{t^{n-1}}\tilde{r}_0\}$$

orthogonal secuences of vectors are generated by a set of simple three term recurrences like in the Conjugate Gradient. The main drawback is that the convergence is quite erratic, and the algorithms is susceptible to fail to converge.

Recently a new version of the BiCG called BiCGstab is presented as a stabilized version which improves the erratic convergence of the original method [Van92].

GMRES is known to be optimal in the sense that in the i-th step, if the starting vector x_0 is the same

$$\left\| r_n^{GMRES} \right\| \leq \left\| r_n^{GCS} \right\|$$

however in the i-th step for the CGS or the BiCGstab the computational cost is O(1) vector operations, in front of the O(N) operations of GMRES. Where N is the number of rows of the A matrix. For these reasons, in a particular problem is not clear a priori what method is optimum in computational time spent. Moreover the practical implementation of GMRES, GMRES(k), is not optimal in the same sense that the original method.

3. Preconditioners

All these methods can be applied to the whole system (1) or to the Schur complement system (3). However in both cases, the convergence of all these methods must be usually improved by a proper preconditioner.

When the whole system (1) is used there are two main kinds of general purpose preconditioner, incomplete factorizations [Meij77] and polynomial preconditioners [Saad85]. We are focus in incomplete factorizations because polynomial preconditioners need some accurate bounds of the spectrum of the matrix in order to be competitive in front of incomplete factorizations.

The main idea of incomplete LU factorizations (ILU) is to factorize the A matrix suppressing the fill in (or part of it) in the factorization phase. These preconditioners are called ILU(k), where the k parameter denotes the amount of fill-in allowed. The preconditioner are the two triangular matrices LU.

$$A = LU + R \quad \Rightarrow \quad M = LU$$

When k=0, i.e., if $a_{ij} = 0$ then l_{ij} or u_{ij} are forced to be zero, a ILU(0) preconditioner with the same sparsity pattern than A is obtained. An improvement of this preconditioner is to add to the diagonal terms of the ILU factors some amount to compensate the contribution of R. This is formulate in terms of a parameter that ranges from zero to one. An experimental value for this parameter need to be determined.

Other improvements has been proposed for ILU preconditioners, but in this paper we only consider the classical ILU(0) preconditioner.

Several preconditioners have been proposed in the literature for the Schur complement matrix [Dyja82], [Golub83], [Gropp92]. However, they usually need more information than the matrix A to be built. In this paper, we are only interested in general purpose preconditioners which can be built only with the information contained in the matrix A. A preconditioner of this kind for the Schur complement matrix was given by Chan and Resasco [Chan85]. The main idea is to approximate the Schur Complement matrix by a k-diagonal matrix, where k is a parameter selected by the user.

The main motivation behind this preconditioner is the observation that the for many elliptic problems magnitude of the elements in the Schur matrix decay rapidly away from the main diagonal. This is because the reduced interface operator has a strong spatial local coupling and a weak global coupling. The formulation of the MSC(k) preconditioner is the following:

$$MSC(k) = A_{BB} - E_k$$

where E_k is a banded matrix with 2k+1 diagonals, which satisfies as accurately as posible

$$E_k \cdot v_i = \left(\sum_{j=1}^{n} A_{Bj} A_{Ij}^{-1} f_{Ij} \right) \cdot v_i \qquad (4)$$

for a set of vectors v_i, i=1,2, . . , L

For non-symmetric matrices L = 2k+1 and the set of vectors used is:

k=0 v1 = [1,1,1,1,. . .]^t

k=1 v1 = [1,0,0,1,0,0,1, . . .]^t

v2 = [0,1,0,0,1,0,0, . . .]^t

v3 = [0,0,1,0,0,1,0, . . .]^t

For symmetric matrices L = k+1 and the set of vectors is reduced to

k=0 v1 = [1,1,1,1,. . .]^t

k=1 v1 = [1,0,1,0,1, . . .]^t

v2 = [0,1,0,1,0 . . .]^t

k=2 v1 = [1,0,0,1,0,0,1, . . .]^t

v2 = [0,1,0,0,1,0,0, . . .]^t

v3 = [0,0,1,0,0,1,0, . . .]^t

If the Schur matrix has N rows then in the non-symmetric case there are (2k+1)N-k(k+1) different elements in E_k, and (2k+1)N equations from **(4)**. There are (2k+1)N-k(k+1) equations which explicitly assigns a value to an individual element of E_k, and there are k(k+1) equations which are inconsistent. This equations are eliminated. Only for k=0 this overdetermination is avoid.

In the symmetric case the overdetermination is consistent. There are (k+1)N equations from **(4)** and E_k has (k+1)N - k(k+1)/2 elements. The diagonal elemnets of E_k are assigned directly from N equations and the remainding elements can be obtained in O(kN) operations.

4. Parallel aspects

If the whole system is used the length of the vectors involved is the number of rows of the A matrix. This can be exploited efficiently in vector multiprocessors for the required vector operations. Each of this operations is a single DO loop which is performed in our target machine with parallel/vector code, i.e. the loop is performed in pieces of the maximum vector length in a strip-mining loop. Then the strip-mining loop is parallelized in several threads over the processors.

Other basic operation is sparse matrix by vector multiplication. Its algorithm is completely dependent of the sparse format in which the matrix is stored. In this paper we restrict ourselves to the Row Wise format (RW). In this format the data structure are three arrays: A, JA, IA. The real array A containing all the no null entries of the matrix, stored row by row from the first row to the last. The integer array JA containing the column indices of the no null elements as stored in the array A. The integer array IA containing the pointers to the beginning of each row in the arrays A and JA. Thus IA(i) is the position in arrays A and JA where the i-th row starts. Then the algorithm to perform Y= A*X is:

```
DoAll i=1,Nrows
     p1 = IA(i)
     p2 = IA(i+1)-1
     t = 0.0
```

```
        Do j=p1,p2
             t = t + A(j) * X(JA(j))
        EndDo
        Y(i) = t
EndDo
```

The external loop is completely parallel and the internal loop can be vectorized by reduction. However there are two problems that can reduce its speed up in vector multiprocessors. First the access to the vector X is between a Gather operation, this means an increment (>30%) in the load time. The second problem is the vector length of the internal loop. It depends on the number of no null entries by row in the matrix A. If this number is little the internal loop will be performed better with scalar code than with vector code.

Finally we need to applied the preconditioner. This means to solve two triangular systems. This algorithm is quite secuential and is a bottleneck in the parallelization of the whole solver. However with the domain decomposition ordering we can solve in parallel each subdomain and add at the end the global coupling.

If the Schur complement system is used the length of the involved vectors is reduced to the boundary unknowns. Usually the number of unknowns associated with the interfaces is greater than the maximum vector length of the machine however for small problems this can reduced dramatically the performance of the code. Now, the other basic operation, matrix by vector multiplication, is performed using the implicit formula for S. This means that we have factorized each subdomain matrix and we used this factors in order to perform the needed solves by subdomain. This is a secuencial bottleneck in the code. The multiplications by the matrices A_{IB}, A_{BI}, are performed also using the RW format. Then in general if we want to obtain good performances with this solver we need to spoilt the medium grain parallelism give by the decomposition. In a vector multiprocessor this can be done performing the solutions on each subdomain in parallel tasks, and using only vector code for the solution algorithm of one subdomain. The overhead to start a set of parallel tasks use to be greater than the overhead of parallel threads, but this can be amortized if the problem is large enough.

5. References

[Chan85] T.F. Chan, D. Resasco, *A survey of preconditioners for domain decomposition*, Technical Report 414, Computer Science Dept., Yale University, 1985.

[Dryja82] M. Dryja, *A capacitance matrix method for Dirichlet problem on polygon region*, Numer. Math., 39, 1982.

[Golub83] G.H. Golub, D. Mayers, *The use of preconditioning over irregular regions*, Lectures of the Sixth International Conference on Computing Methods in Applied Sciences and Engineering, Versailles, 1983.

[Gropp92] W. Gropp, D. Keyes, *Domain Decomposition methods with local mesh refinement*, SIAM Jour. Sci. Stat. Comput., vol. 13, No. 4, 1992..

[Saad86] Y. Saad, M.H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM Jour. Sci. Stat. Comput., vol. 7, No. 3, pp 856-869, 1986.

[Saun88] M. A. Saunders, H. D. Simon and E. L. Yip, *Two conjugate-gradient-type methods for unsymmetric linear equations*, SIAM Jour. Num. Anal., 25, pp. 927-940, 1988.

[Sonn89] P. Sonneveld, *CGS, a fast Lancros-type solver for nonsymmetric linear systems*, SIAM Jour. Sci. Stat. Comput. , vol.10, No.1, pp 36-52, 1989.

[Van92] H. A. Van der Vorst, *BI-CGSTAB: A fast and smothly converging variant of the BI-CG for the resolution of nonsymmetric linear systems*, SIAM Jour. Sci. Stat. Compt., vol.13, No.2, 1992.

C-XSC, A C++ Class Library for Extended Scientific Computing

Ulrich Kulisch, Karlsruhe University

The programming environment C-XSC (C for eXtended Scientific Computing) is a powerful and easy to use programming tool, especially for scientific and engineering applications. C-XSC makes the computer more powerful arithmetically and significantly simplifies programming. C-XSC is implemented as a numerical class library in the programming language C++, a superset of the wellknown language C. C-XSC may be interpreted as an arithmetic module extending the properties of the language C or as a numerical extension of C.

Numerical mathematics has contrived algorithms which deliver highly accurate and automatically verified results by applying mathematical fixed-point theorems. This means that these computations carry their own accuracy control. However, their implementation requires suitable arithmetic support and powerful programming tools which were not previoulsy available. The development of C-XSC has aimed at providing these tools within C and C++. C-XSC is particularly suited for the development of numerical algorithms that deliver highly accurate and automatically verified results, which are essential, for example, in simulation runs where the user has to distinguish between computational artifacts and genuine reactions of the model. Problem-solving functions with automatic result verification have been developed in C-XSC for many standard problems of numerical analysis.

C-XSC consists of a run time system written in C including an optimal dot product and many predefined data types for elements of the most commonly used vector spaces such as real and complex numbers, vectors, and matrices as well as for real and complex intervals, real and complex interval vectors and matrices. Operators for elements of these types are predefined and can be called by their usual operator symbols. Thus, arithmetic expressions and numerical algorithms are expressed in a notation that is very close to the usual mathematical notation. Using C-XSC, many programs can be read like a technical report. All predefined numerical operators are of highest accuracy. That is, the computed result differs from the correct result by at most one rounding.

C-XSC provides 24 mathematical standard functions for real and complex arguments and for real and complex interval arguments. The computed values are accurate to within one or two bits in the last place. While the emphasis in computing is traditionally on speed, in C-XSC, the emphasis is more on accuracay and reliability of results. The total time for

solving a problem is the sum of the programming effort, the processing time, and the time for the interpretation of results. C-XSC reduces this sum considerably.

C programmers should be able to use and write programs in C-XSC immediately. C-XSC simplifies programming by providing many predefinded data types and arithmetic operators. Programs are much easier to read, to write, and to debug. The programming environment C-XSC is easily portable to any computer that supports a standard-conforming C++ compiler.

The most important features of C-XSC are:

- Real, complex, interval and complex interval arithmetic
- Standard functions of high accuracy
- Predefined arithmetic operators with maximum accuracy for real, complex, interval and complex interval vectors and matrices
- Dynamic vectors and matrices
- Subarrays of vectors and matrices
- Multiple- precision arithmetic and standard functions
- Dotprecision data types
- Rounding control for I/O data
- Library of problem solving routines with automatic result verification

The reference manual and language description of C-XSC has been published by Springer-Verlag (ISBN 0-387-55137-9). Software for C-XSC as well as the language description are available from "Numerik Software GmbH", Rettigstraße 6, D-7570 Baden-Baden, Germany, FAX-Number: +49-721-69 44 18, Phone: +49-721-69 42 17.

The software package contains problem solving routines with automatic result verification for linear systems of equations, matrix inversion, eigenvalues of symmetric matrices, Fast Fourier Transform, nonlinear systems of equations, ordinary differential equations and others.

SUPEREXCITED MOLECULES: LARGE MEMORY TO THE RESCUE

James R Henderson (1,2) and Jonathan Tennyson (1)

(1) Department of Physics and Astronomy
University College London
Gower Street, London WC1E 6BT, UK

(2) University of London Computer Centre (ULCC)
20 Guilford Street, London WC1N 1DZ, UK

1 Introduction

At the beginning of the 20th century science was to witness a major change through man's understanding of the fundamental description of matter at the very small scale. This change heralded the start of 'the new physics' of our time, and has had remarkable consequences for all areas of science. The discoveries from this change were to go forth and underpin the whole of modern technology.

The first few decades of the century saw scientists such as Einstein, Planck, Bohr, Schroedinger, Dirac and Heisenberg laying the foundations of *Quantum Theory.* The theory of quantum physics was the new description of molecular, atomic and subatomic systems and processes. The conceptual basis was at first shocking, as it detracted enormously from the more 'agreeable' or 'intuitive' description of nature loosely referred to as Newtonian Mechanics or Classical Mechanics – a set of laws which describe the behaviour of every day objects particularly well.

One of the huge *caveats* of the exciting new quantum theory was that it produced equations which were really very difficult to solve. When reviewing the status of quantum theory in the late 1920s the Nobel Physicist Paul Dirac wrote:

'The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are .... completely known, and the difficulty is only that the exact application of these laws leads to equations much too complicated to be solved.'

Since then we have seen the introduction of electronic computers and witnessed enormous strides in their capabilities. In parallel scientists have developed very powerful and sophisticated theoretical techniques and approximation methods. Using today's powerful supercomputers areas such as physics and quantum chemistry are now able to perform very large and accurate calculations for quantum systems and processes. Such features as powerful vector processing, fast I/O capabilities, large disk storage, very large memory and parallel processing are behind highly successful computer modelling and simulation, and are of such vital importance for today's scientific advances.

In this paper we review our success in using the Convex C3860 at ULCC in solving a fully quantum mechanical description of the fascinating hydrogen molecular ion, H_3^+. We describe how certain features of this simplest of triatomic molecules have eluded understanding until very recently when we were able to exploit the very fast vector and parallel processing capabilities of the C3860, and in particular the large 4 Gigabyte memory.

2 The Problem and its History

The simplest stable triatomic molecule is the H_3^+ molecular ion, consisting of three nuclei and just two electrons. Despite being an apparently straightforward system, quantum mechanically it still poses enormous challenges to theoretical investigation. When a molecule begins to rotate and when its constituent atoms begin to vibrate erratically against each other, the resulting motions can be extremely complex; it is the calculation and understanding of these motions that we discuss in this paper. We explain how we are using supercomputers to try to understand the very highly excited motions of molecules which have so much rotational and vibrational energy that they are about to break apart – at the so-called *dissociation limit.*

Studying the rotation-vibration behaviours of small molecules is of fundamental importance in many areas of science. It can help to increase our understanding in topics such as the structure of matter, the dynamics of chemical reactions and scattering, the atomic and molecular conditions and processes in space, the very exciting area of chaos, and such topical and important areas as atmospheric chemistry. For example it is exactly the excitation of vibrational and rotational motion of gases such as carbon dioxide which lead to the trapping of heat in the earth's atmosphere and hence global warming via the so-called greenhouse effect.

About ten years ago Alan Carrington and co-workers [1] at the University of Southampton made some remarkable observations of H_3^+ in their laboratory laser experiment. Their experiment was to allow a carefully tuned laser to interact with a beam of H_3^+ molecular ions and to pictorially record the results in the form of a *spectrum.* One of the major differences between quantum and classical descriptions of matter is that quantum mechanics can only allow certain discrete energy levels in a particular system – corresponding to particular *quanta* of energy. When a molecule experiences a decrease in energy then its energy will jump from a higher quantized level to a lower one. When this jump occurs the surplus energy is emitted and then can be recorded as a *line* in a spectrum. The reverse process happens when a molecule's energy is increased, and an absorption spectrum is recorded. When the molecule's energy moves from one discrete level to another, we say that a *transition* has taken place.

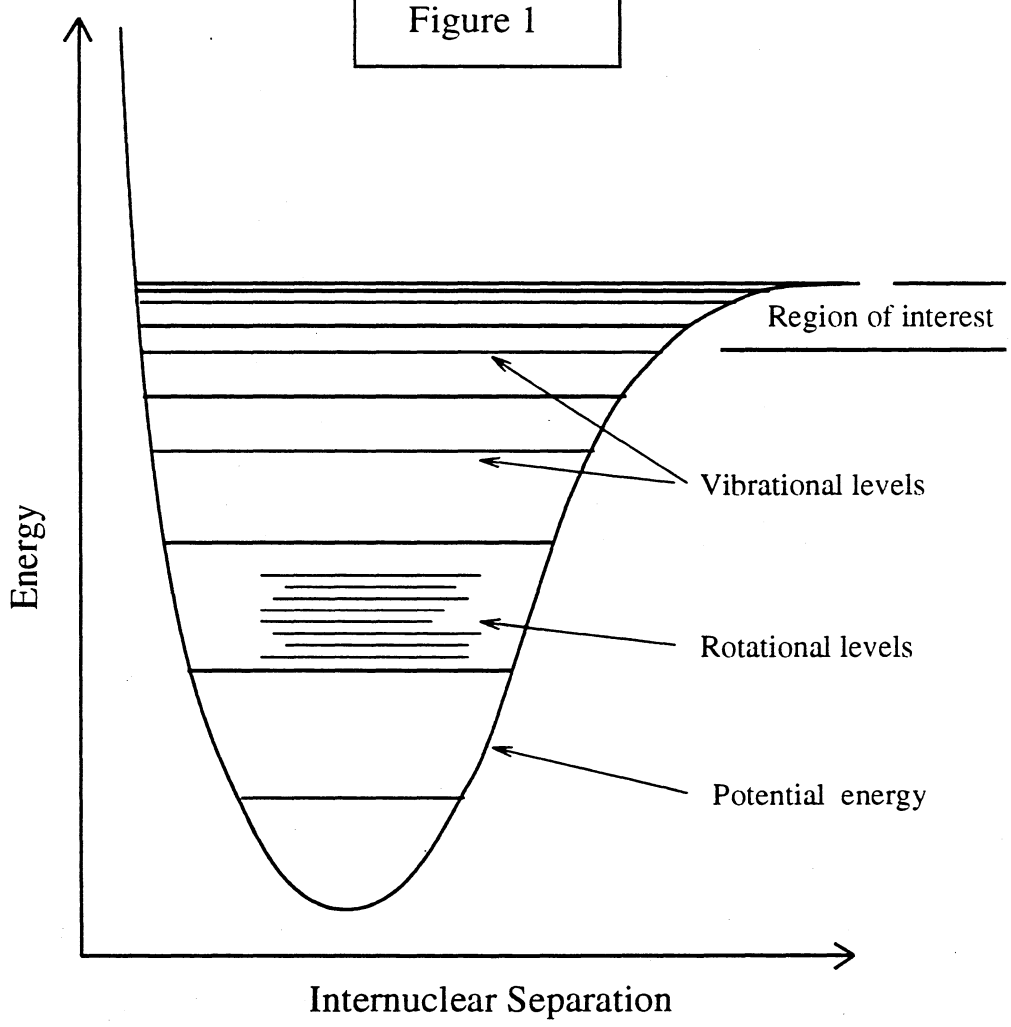
It is the study of the magnitude and structure of these lines that can yield a large amount of information about quantum systems such as atoms or molecules. In many cases, usually at low energy, these lines are well understood and serve as 'fingerprints' for particular molecular species or processes. High energy spectra are generally much more challenging in terms of their interpretation.

One of the first interesting features concerning the Carrington spectrum was the number of lines; even though the spectrum was recorded for a relatively small range in the energy change, a remarkable 27,000 lines were recorded. Carrington's group actually recorded this spectrum by watching the molecule break-up (dissociate) as they shone light on it. As their laser only gave each ion a small amount of energy, the molecules must have started in the dissociation region. Thus the observed transitions link many states, all lying very close to each other at the dissociation limit. The density of quantized energy levels, or *states*, is very high here; the density of states for a quantum system can increase dramatically as energy increases. So the 27,000 lines in the narrow window of the Carrington spectrum are due to ro-vibrational transitions between states in the very dense sea right at the top of the so-called *potential energy well.* Figure 1 is a simplified sketch of a one-dimensional potential energy well, showing a few quantum energy levels and the increase in their density as energy increases. Note that the potential well (the outer curve in figure 1) is obtained by solving for the motion of the electrons. Molecular rotation-vibration descriptions usually make use of the approximate decoupling of the electronic motions from the nuclear motions. In this respect we interpret a transition as a change in the nuclear motion 'on' the potential energy surface. The nuclear motion has two components: vibrational motion (the

coarse set of levels in Figure 1) and rotational motion. Each vibrational level has an associated set of rotational levels, one such set is given schematically in Figure 1. Rotational and vibrational motion are coupled through *Coriolis forces*.

Figure 1 illustrates a potential energy curve in one dimension, or co-ordinate. A description of the H_3^+ system needs three internal co-ordinates to define the relative positions of the nuclei. In our work we use R_1, R_2 and θ, which are shown in Figure 2. It is conventional to describe molecular vibrations in terms of coordinates representing stretching and bending motions. However at the energies of interest to us, these motions become very strongly coupled and impossible to identify.

Figure 1



The major puzzle in the spectrum became apparent when Carrington and his co-workers attempted to understand the structure of the spectrum and its lines. Although the lines were extremely well defined and resolved, they appeared to hold no explainable structure. There were no recognized 'patterns' or any means of identifying exactly what physical and chemical processes might be taking place. One idea they had was to convolve the spectrum and recreate it on a broader scale – equivalent to recording the spectrum at a much lower resolution, thus emphasizing any coarse-grained regularity that may be present. Amazingly the spectrum then reduced to four equally spaced clumps; it appeared that this was an example of underlying structure in the

'chaos'[1] of quantum mechanics – something which scientists knew very little about.

The scientific importance of this difficult problem became more apparent. Its further study would help to explain the poorly understood physics and chemistry of the reaction and dissociation dynamics of chemically bound systems. Scientists acknowledged the need for theoriticians to perform the best possible fully quantum mechanical calculations for H_3^+.

A massive theoretical and computational challenge lay ahead.

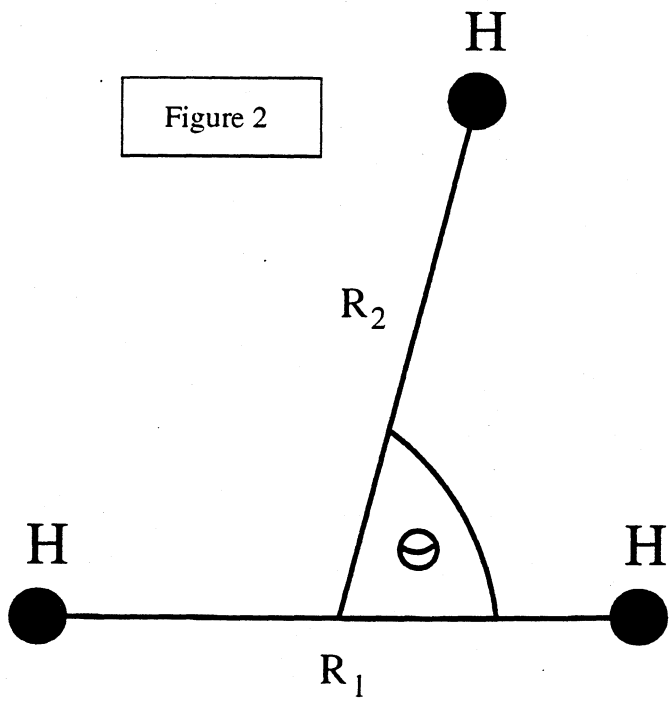


Figure 2

3 Solution Algorithms

When a molecule is in a state of low energy then its atoms tend to be slowly vibrating in a regular manner not too far from its equilibrium, or minimum energy, configuration. The motions can often be understood in terms of approximations based on harmonic oscillator (or 'ball-and-spring' type) descriptions. This kind of behaviour has been fairly well understood for several decades, and is very useful for many purposes.

For solving the nuclear motion problem of small molecular systems, an approach frequently employed in recent years is the so-called Finite Basis Representation (FBR) [2]. It is based upon the idea of choosing suitable (polynomial) *basis functions* and expressing the quantum mechanical solutions, or *wavefunctions*, as expansions in them. A matrix representation is then formulated and the diagonalization of that matrix yields the values of the discrete energy levels (the eigenvalues)

[1] Chaos should not strictly be used in the quantum mechanical sense, as it is a feature of classical mechanics. The study of classically chaotic problems using quantum mechanics is, however, a very active one and is often called *quantum chaology*.

and the wavefunctions (eigenvectors). The wavefunction corresponding to a particular quantum energy level describes the motion and physical characteristics of that state. After a final FBR calculation one often 'post-processes' using the wavefunctions to produce spectra, for example.

As a molecule's energy increases and the potential energy surface becomes far removed from a harmonic approximation, then the motions become less easily describable. The wavefunctions tend to be no longer localized and the system can access widely differing parts of the potential energy surface, often via very large amplitude and strongly coupled motions. The FBR method can become inappropriate and intractable here: 'good' basis functions are very difficult to find, and the final matrix, the so-called *Hamiltonian matrix*, becomes unmanageably large – even for today's supercomputers. In our recent experience we abandoned the FBR method with the Na_3 molecule [3] – this system has such a complicated potential energy surface that we found it very difficult to accurately predict even the low energy properties.

We decided to study highly excited molecules using different techniques. In collaboration with Brian Sutcliffe from the University of York, we have spent the last five years mastering a novel finite element technique called the Discrete Variable Representation, or DVR [4]. This method deals with a grid formalism, rather than the function representation of the FBR. The DVR method is based upon the transformation from a multidimensional basis function FBR description to a discretised grid representation and results in a DVR Hamiltonian matrix which can be very sparse.

The solution procedure is then to take the sparse DVR Hamiltonian matrix and perform a hierarchical succession of diagonalizations and truncations of reduced dimensionality matrices for each of the co-ordinates, on each of the grid points. This leads to a final, fully coupled matrix of much reduced size and very high information content. This matrix is now dense and symmetric. Its diagonalization yields the required solutions. Our experience has shown that if one diagonalizes an FBR matrix of order N then one can expect to obtain about $\frac{N}{20}$ accurate solutions; the equivalent fraction for the new DVR matrix can be as much as $\frac{N}{3}$. Let us remember the importance of such savings; apart from being able to circumvent memory restrictions, *huge* CPU requirements need to be avoided (matrix diagonalization times are proportional to N^3).

Using the powerful new DVR method we are now able to create matrices that will produce substantially more levels than before. The very large memory of the Convex C3860 is also allowing us to store extremely large matrices in core. The calculations we are now able to perform are about to lead to a major breakthrough in the Carrington story, and substantially increase our understanding of the highly excited dynamics of small molecules.

4 The Introduction of Large Memory at ULCC

Our research group has been concentrating on this problem for almost ten years. Throughout the 1980s, before the use of the DVR, it was the FBR method which the group mastered and developed substantially. We were always restricted in the amount of memory available to store the final matrix to be diagonalized – efficient solution of a dense matrix requires it all to be stored in core. During that period we were able to use all of the available memory on UK supercomputers, and still require more. Figure 3 shows the various machines housed at ULCC along with the memory that was available. Before the arrival of the Convex machine it can be seen that 8 Mwords was the largest memory, which actually meant about 7 Mwords being the maximum available on special arrangement – ie. the whole of a national facility dedicated to one user, for a weekend for example. Using 64-bit words this means that a matrix of order N=2600 could be stored and solved.

In 1991 Convex won the contract to replace the Cray and provide ULCC's main supercomputer –

a C3840 (Neptune) with 1 Gigabyte of memory, front-ended by a C220 (Pluto) with $\frac{1}{4}$ Gigabyte of memory. This was a significant leap in the amount of memory available in UK supercomputing. The Neptune configuration is now even larger – 6 processors and 4 Gigabytes of memory. Clearly we can now hold very much larger matrices in core and use the scheduled batch service at ULCC to solve these – jobs requiring more than a Gigabyte of storage can be routinely run alongside other large memory batch and interactive requests.

Figure 3

CDC 7600	0.5 million words	1972 - 84
Cray 1S COS1M	1 million words	1983 - 89
COS2M	2 million words	
Cray X-MP/48	8 million words	1989 - 91
Convex C3860	375 million words	1991 -

5 The H_3^+ Calculations

The first requirement in attempting to reproduce the Carrington spectrum is to accurately compute *all* of the rotation-vibration quantum wavefunctions and energies for H_3^+. Although this task has been impossible for many years, we are now close to achieving it.

Firstly we have managed to calculate all of the vibrational wavefunctions for a non-rotating system [5], using our program DVR3D [6]. About 1000 energies were calculated to within an accuracy of 1 part in 35,000. That study involved the construction and diagonalization of a double precision real symmetric matrix of size 10,000 × 10,000. The largest of these runs used almost 50 hours execution time on Neptune, using a diagonalizing routine from the NAG library. The efficiency of these large calculations was improved somewhat by the NAG library making calls to highly optimized BLAS routines from the Convex Veclib library. These routines become particularly useful if Neptune experiences periods of light load, when they are able to execute in parallel mode. The wavefunction vectors obtained from this calculation, which occupied several

hundred megabytes, were most conveniently stored on the ULCC StorageTek 4400 Automated Cartridge Store archive facility for future processing.

It is useful to note that without the diagonalization-truncation scheme of the DVR the resulting FBR matrices for the above calculation would have been 23,000 × 23,000, corresponding to 36, 40 and 16 grid points in the R_1, R_2 and θ co-ordinates. This would require over 4 Gigabytes of storage and about 650 hours CPU time to solve – and this is just for the non-rotating molecule!

The results are already making very exciting contributions to the Carrington problem. We were able to confirm that the quantum analogue of a periodic orbit predicted by classical studies on H_3^+ actually existed in a full quantum description. Our calculations confirmed that this motion was stable all the way up the potential energy well to dissociation. The orbit has been termed the 'horseshoe'. Figures 4 and 5 show contour plots of computed quantum wavefunctions; they are for R_1 (vertical axis) against R_2 (horizontal axis) in atomic units, with θ frozen at 90^o. Figure 4 is quantum state number 244, and depicts a 'horseshoe' state – the horseshoe shape is obtained by reflecting the figure about the $R_2 = 0$ axis. Notice that the motion is localized in a particular curved region of space, and that it is 'regular'. At about one third of the way up the potential energy well the H_3^+ molecule is capable of becoming linear, with one H atom capable of passing between the other two. The rapid movement of one between the other two is the horseshoe mode, and is interpreted as a highly excited bending motion of a quasi-linear molecule. Linearity is represented by a *saddle point* on the potential energy surface; often when a system's energy goes above features such as barriers and saddle points wide amplitude and irregular motions can occur. It is very interesting that amongst the highly irregular states we have calculated all the way to dissociation, we see these structured and *localized* horseshoe states.

Figure 4

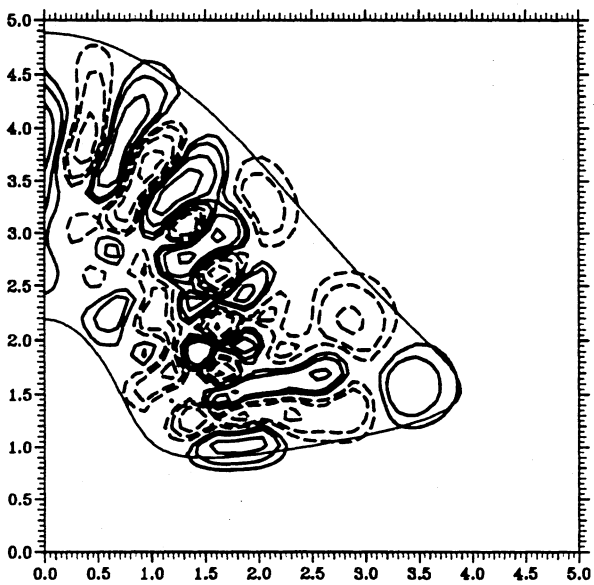


Figure 5

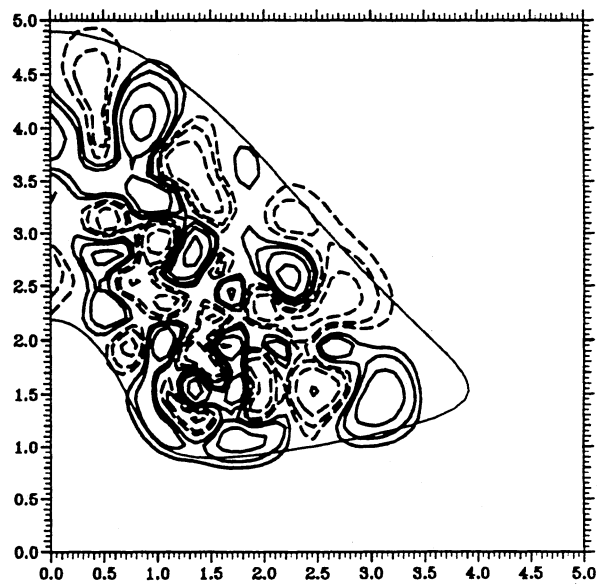


Figure 5 (quantum state number 249) shows an equivalent plot for a state which is not a horseshoe, but one of the many *delocalized*, irregular states, which corresponds to classical chaos. Notice how the 'chaotic' state fills all of the available space, which is restricted by the classical turning point – shown as the outer contour in the plots. Note that the classical turning point is simply a cut through the potential energy well at the energy of the quantum state in question. They were

made using the Uniras graphics package which is installed on both Pluto and Neptune.

It has been proposed for some time that the interesting coarse-grained regularity in the Carrington spectrum may be attributed to the horseshoe states. It is thought that the equally spaced 'clumps' are explainable in terms of rotating horseshoes states; again, our calculations have been able to support this from a fully coupled quantum mechanical view. Everything we have discovered so far has not been able to rule out the horseshoe as playing a major part in the spectrum. We have been able however to rule out another candidate periodic orbit however, as our calculations found it to be unstable (ie. it disappeared) with increasing energy.

So far in this section we have described our calculations for the nonrotating H_3^+ molecule. Although mastering this problem is necessary, and indeed very difficult, the Carrington story requires treating problems with significant rotational motion. In quantal systems, the total rotational energy of a system is given by a quantum number generally denoted J. Rotational calculations lead to even larger matrices and densities of states $(2J + 1)$ higher than the nonrotating, $J = 0$ problem. The rotation-vibration levels tend to pile up finely on top of the pure vibrational levels (see Figure 1), and usually make spectra extremely complicated. This is particularly so when the ro-vibrational levels corresponding to different vibrational *manifolds* overlap. This happens more for increasing energy, which is one of the reasons why high energy spectra are difficult to calculate and understand.

More recently we have been able to develop the DVR method to use a neat two-step rotational approach and use a further-contracted basis technique to solve the coupled rotational and vibrational problem. We now have a program called ROTLEV3 [7] which takes the DVR3D output and computes the second step in the rotation-vibration calculations. So far we have performed some very accurate H_3^+ calculations for $J = 1$ and $J = 2$; each of these solved several 10,000 $\times$ 10,000 matrices, as for the $J = 0$ case. Rotational excitation calculations are performed in two steps, each relying on matrix diagonalization. The first of these steps corresponds closely to the $J = 0$ problem; the second step results in a symmetric, sparse matrix with a predictable structure. Usually iterative diagonalization techniques have been used for this matrix as retaining only the non-zero matrix elements gave considerable savings in memory. However the present calculations required so many solutions that iterative diagonalization proved very inefficient. We were able to utilize the large memory of the Convex to retain the entire matrix and diagonalize it directly, and much more efficiently.

The $J = 1$ calculations involved two jobs of over 100 hours each, using almost 1 Gigabyte of memory each. $J = 2$ required two 1 Gigabyte jobs of 150 hours each. One of the 150 hour jobs made particularly efficient use of the parallel capabilities of the Veclib routines as it ran on a very lightly loaded machine over Easter. At times the CPU utilization was seen to be over 400% from a possible 500% (Neptune was configured with only 5 processors at the time). The density of states again showed dramatic increases with J; we calculated approximately 6000 bound states for just $J = 2$, although the accuracy of these was proving difficult to stabilize at the very top of the potential energy surface. Output vectors from these jobs were up to 1.4 Gigabytes in size, and needed secure storage for future processing. The ACS archive was able to store these files most conveniently, and allows their online retrieval directly to Neptune at any time.

6 Fault Handling

When running very long jobs on any computer there is always a risk of some failure which causes termination of the processes and hence the loss of data. A common way of preparing for this is to periodically save data or a full image of the process to disc. For our purposes we consider this

saving of information, commonly called checkpointing, to be vital.

One of the most obvious methods is for the user to write his or her program such that the large and important data are periodically written to dump files so that they can be picked up later in the event of a failure. The advantage here is that this can be reasonably portable across architectures, whilst the disadvantage is that the user will probably need to know the intricacies of the code, and may well need to influence its performance. Our DVR3D and ROTLEV3 codes are capable of spending most of their time in the third party library routines and hence we have found it necessary to actually modify a copy of the Fortran source of some of the NAG routines. This project is in progress at ULCC, and will result in a NAG-derived diagonalizer which is fragmentable into many, isolated, small batch requests.

Neptune's operating system, ConvexOS 10.2, is capable of instructing its own checkpointing to batch requests. One option is the *qchkpnt* command which can be issued by the user to save an image of the batch process(es) to a system-specified filesystem at a specified period, in terms of hours, minutes etc. When the operating system restarts the failed process then the checkpoint file is picked up and the job will resume from the last checkpoint time. When the job terminates successfully the checkpoint files are removed by the system. The equivalent command can actually be placed in the user's source so that the program checkpoints itself – the command operates similarly and is called *chkpnt*.

The most useful method of checkpointing for our own purposes was to use a switch on the *qsub* command – the command for submitting batch requests. Simply by using *qsub -cp m hours jobname* we were able to ensure automatic checkpointing of the batch request *jobname* every m hours. This precaution has proved its worth several times, and has potentially saved us many hours of computing.

7 Discussion and Conclusions

We are using a very large memory national facility Convex C3860 at ULCC to study the very highly excited quantum states of small, chemically bound molecules at their dissociation limit. Understanding the mysterious H_3^+ spectrum of Carrington and co-workers is the main goal in this research, though this topic in general is of great interest in many areas of chemistry, physics and astronomy. The large memory has allowed us to hold very large matrices in core, and hence, along with the powerful DVR method, we are now able to accurately compute *all* the rotation-vibration bound states of a triatomic molecule for the first time ever. Interestingly, the very large memory has also led us to change our solution algorithm in the second step of the rotation-vibration calculations.

So far we have calculated $J = 0, 1$ and 2 states of H_3^+. Almost all states were of the 'chaotic' type, although the family of horseshoe states appeared stable throughout $J = 0$. Elsewhere [8] we have shown these orbits to be capable of 'leaking' intensity to nearby states. Interestingly, all of the levels computed so far have shown remarkable agreement with classical predictions made by Berblinger *et al* [9].

8 The Future

8.1 Science

We now have a new program, DIPOLE3, working which can compute complete spectra using output wavefunctions from DVR3D and ROTLEV3. Our initial tests have reproduced observed spectra for transitions between low lying levels. We are just about to compute the spectra for *all* the rotation and vibration transitions between $J = 0, 1$ and 2. This alone will produce many thousands of lines – it is possible that we may see some structure in the region of interest. It is strongly suspected however, that all the ro-vibrational levels up to at least $J = 10$ will be needed to reproduce the Carrington spectrum. This is an even more formidable task which will require thousands of compute hours. We are confident that these calculations are within our reach now, and we expect to be making further progress very shortly, after we have fully analyzed the vast amount of data we have so far.

There have been relatively few accurate studies on systems larger than triatomics. The number of internal co-ordinates in just going from a three to a four-atom molecule *doubles*, and hence the matrices become unmanageably large. The theoretical techniques are far from straightforward, and require further development. We have now developed a prototype DVR method for tetratomic systems, which we are currently translating into a Fortran program. We expect soon to be producing accurate new data for these molecules and hope to discover new science and unravel a wealth of experimental and astronomical data. State of the art supercomputing facilities are going to be vital for this project's success.

8.2 Computing

Theoretical atomic and molecular physics has been identified [10] in a supercomputing study by the UK's major scientific and engineering funding body (SERC) as one area which is capable of using huge supercomputing resources, both now and in the future. We have certainly demonstrated that to be the case in this article. Atomic and molecular physics will continue to demand further compute power, memory and disc storage as many challenging problems, such as electron molecule collisions and the accurate study of tetra- and pentatomic molecules, slowly become doable.

Whilst researching the project described in this paper we have developed views on what kind of computational restrictions we currently face, and which current and future developments will help to circumvent them. One of the main problems now appears to be CPU compute power; memory is no longer a problem for the time being, but being able to use powerful enough processors to process huge amounts of data is now limiting what is practical in terms of a calculation. Massively and scalable parallel processing appear to hold the answer. Here it is more apparent that memory and processing power are both allowed to advance at more consistent rates. We have recently ported the DVR3D code to the Thinking Machines CM-200 at Edinburgh Parallel Computing Centre (EPCC). The available memory on that machine (0.5 Gigabyte) was rather too small for state of the art calculations such as those described here – we look forward to porting our codes to larger parallel facilities.

It is well known that software and applications for parallel architectures have been rather underdeveloped in the last five years – indeed, huge amounts of effort are now being expended to rapidly reverse the situation. We believe a very important requirement for the efficient study of atomic and molecular physics, as well as the very demanding area of quantum chemistry, is the provision of very highly optimized BLAS libraries. We also think it is crucial that an operating system has the ability to automatically checkpoint batch processes, as well as using sensible procedures in dealing with single node failures in a multi-node environment.

Acknowledgements

We thank Brian Sutcliffe for his continued involvement throughout this work. We are also grateful to Nic Fulton, Ruth Le Sueur and Steven Miller for their help and comments.

References

[1] A. Carrington, J. Buttenshaw and R.A. Kennedy, *Mol. Phys.*, 45, 753 (1982).
[2] J. Tennyson, S. Miller and J.R. Henderson, in *Methods in Computational Chemistry*, 4, S. Wilson (Ed.), 91, (Plenum, New York, 1992).
[3] J.R. Henderson, S. Miller and J. Tennyson, *J. Chem. Soc., Faraday Trans.*, 86, 1963 (1990).
[4] Z. Bacic and J.C. Light, *Ann. Rev. Phys. Chem.*, 40, 469 (1989).
[5] J.R. Henderson, J. Tennyson and B.T. Sutcliffe, *J. Chem. Phys.*, 97, 7191 (1993).
[6] J.R. Henderson, C. Ruth Le Sueur and J. Tennyson, *Comp. Phys. Comms.*, 75, 379 (1993).
[7] J. Tennyson *J. Chem. Phys.*, 98, 9658 (1993).
[8] C.R. Le Sueur, J.R. Henderson and J. Tennyson, *Chem. Phys. Letts.*, 206, 429 (1993).
[9] M. Berblinger, C. Schlier, J. Tennyson and S. Miller, *J. Chem. Phys.*, 96, 6842 (1992).
[10] *Research Requirements for High Performance Computing*, Science and Engineering Research Council, September 1992, ISBN 1 8706699 33 9.

METACOMPUTING: FROM THE DESKTOP TO THE TERAFLOP

Mike Tighe
Convex Computer Corporation, Tx, USA

This presentation will describe Convex's solutions for technical users. Among these will be the methods by which desktop workstations, C-Series, meta-series, and SPP machines are integrated into the meta-environment. It will describe the role each plays, as well as what future technologies users can expect to see Convex incorporating into the meta-environment.

POSTER SESSION:

This poster session will provide a real world example of using the meta-series to develop SPP applications. Comparison to other systems, parallel speedups, parallel algorithms, will be discussed.

NUMERICAL SIMULATION OF IMPACT PROBLEMS

J. Martí[1,2], F. Beltrán[1], F. Martínez[1], F. Mayoral[2]
[1] Principia S.A., Velázquez 94, 28006 Madrid, Spain
[2] School of Mines, Ríos Rosas 21, 28003 Madrid, Spain

1. INTRODUCTION

Impact problems are among the more demanding ones in terms of computer time. In comparison with other mechanical or structural problems, the high demands in computer time are due to a number of characteristics which generally accompany impact problems, namely: they are dynamic, non-linear and, very often, require a full three-dimensional treatment.

Their dynamic character implies that the inertia terms of the equations of motion cannot be neglected in the analysis; furthermore, not only rigid body inertias (which can be represented as body forces), but also local inertias (as activated in wave propagation phenomena) must be duly accounted for.

Impact problems are non-linear because of a variety of reasons. First of all, the contact conditions between the impacting bodies are inherently non-linear. Furthermore, the levels of deformations induced in one or both of the bodies are usually sufficient to develop non-linear mechanical behaviour in the materials (constitutive non-linearity) and even to require the use of non-linear measures for the deformation process (geometric non-linearity).

Occasionally, the symmetries of the problem allow the use of idealizations with less than three space dimensions, chiefly two-dimensional axisymmetric models. However, this is applicable only in normal impacts between aligned bodies with axially symmetric geometries and in the absence of transverse forces. Although one of the three cases presented here falls in this category, the majority of cases require a full three-dimensional analysis in order to be reasonably representative of the real problem under consideration.

2. BASIC FORMULATION

Two basic sets of equations govern the mechanical behaviour of deforming bodies: the equations of motion and the constitutive laws.

The local formulation of the equations of motion is as follows:

$$\sigma_{ij,i} + \rho\,(f_j - \ddot{u}_j) = 0 \qquad (1)$$

where σ_{ij} is the Cauchy stress tensor

ρ is the density

f_j is the body force

$\ddot{u}_j$ is the local acceleration

In their numerical implementation, these equations are referred to a grid of small elements spanning the bodies under consideration. In a Lagrangian

grid, which deforms with the body, the equations of motion can be expressed for each nodal point in the grid as:

$$\ddot{u}_j = (\int_s \sigma_{ij}\, n_i\, ds + f_j) / m \qquad (2)$$

where the integral extends to the surface surrounding the volume of material whose mass m is considered to be concentrated at the node. Since grid moves with the material, the mass associated to each node does not change with time.

The constitutive laws express the relationship between the states of stress and strain at each point in the body. Due to the lack of objectivity of the traditional time derivatives of Eulerian tensors, the formulation of the constitutive laws is not trivial, even for simple material behaviours, whenever large strains and rotations may be involved. The more common procedure is based on rate-type formulations:

$$\hat{\sigma}_{ij} = F(D_{kl}, \ldots) \qquad (3)$$

where $\hat{\sigma}_{ij}$ is an objective rate (Jaumann, Truesdell, Lie, etc.) of the Cauchy stress tensor

$D_{kl} = (\dot{u}_{k,l} + \dot{u}_{l,k})/2$ is the rate of deformation tensor

F is a scalar or tensorial function describing the relationship between the stress rate, the deformation rate and possibly other variables such as the temperature, accumulated plastic strain, etc.

In a similar role to the constitutive laws, impact problems generally require the use of contact laws. Just like the constitutive laws relate relative motions inside a body to its internal state of stress, contact laws relate relative motions across a contact interface to the forces of interaction developed at that interface.

For impact problems in which relative particle velocities are well below the speed of sound of the material, the most convenient formulation is afforded by explicit Lagrangian procedures. The scheme of the calculations is depicted in Fig. 1.

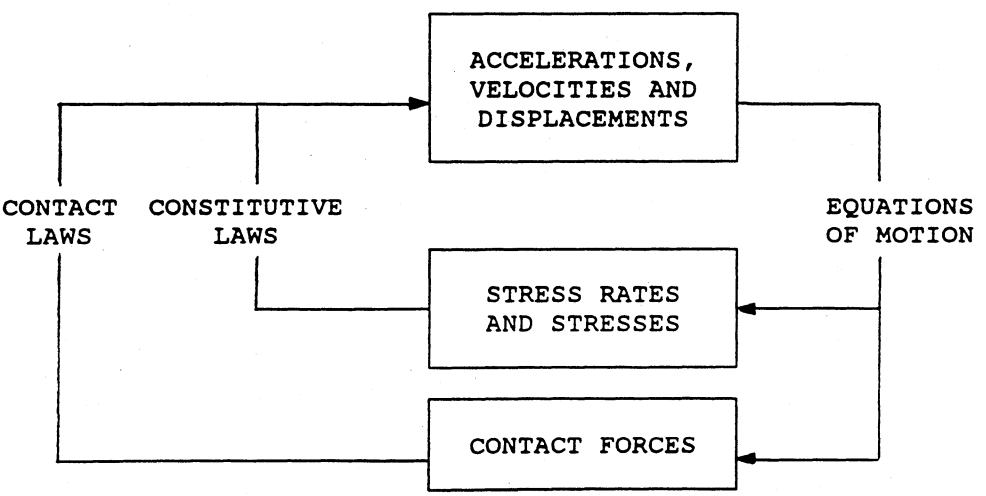


Fig. 1 Computational cycle

The integration cycle shown in the figure must be repeated as many times as necessary to span the time interval of interest. The basic limitation imposed on the size of the time step for integration is given by the Courant condition; this condition essentially requires that the time step be so short that no two nodes have time to communicate during one time step, which effectively uncouples their equations of motion. In most cases, this limitation is also sufficient to ensure adequate accuracy of the integration process.

3. COMPUTATIONAL CONSEQUENCES

The previous scheme presents great advantages for its numerical implementation. Among them:

- all equations of motion are uncoupled
- no matrices need to be inverted; no systems of equations have to be simultaneously solved
- the constitutive laws can be used directly in an explicit form; no special treatment is required for material softening and similar behaviours, which inevitably lead to ill-conditioning of matrices in other formulations
- storage and computer time requirements increase only linearly with the size of the problem (number of elements in the grid and number of time steps in the time interval of interest)
- most of the operations are easily amenable to natural implementations in vector and parallel processors; the main exception relates to the contact algorithms, which by necessity involve searches to find out which element is contacting which at each time.

There are certain limitations, however, in this procedure. The main ones are listed below:

- the conditional stability of the explicit integration process may require large numbers of time steps for solving certain problems, but this is seldom of great concern in impact cases, which are typically short-lived events
- contact searches are often time consuming and, as mentioned earlier, awkward to vectorize or parallelize
- special measures are required when relative particle velocities approach the velocity of sound and, generally, for the control of shock waves in the material

Overall, it can be concluded that explicit Lagrangian procedures are very well suited to deal with impact problems within the conditions mentioned before. This is corroborated by the examples discussed in the following Sections.

4. LOW-LEVEL WASTE CONTAINER

The example presented here has been taken from a series of analyses performed in order to assess the feasibility of achieving a safe and economically attractive transport for a given waste package. The motivation for the analyses was that certain low-level nuclear waste contained in glass bottles had to be transported. An attractive alternative was to surround the glass bottle with a shock absorbing material and to enclose

this assembly within a steel drum.

The investigations conducted were designed to assess the feasibility of the proposed concept and to optimize the properties of the shock-absorbing material. The design impacts for the assembly were the 9m drops onto an unyielding target originally proposed in 1973 by the International Atomic Energy Agency [3].

The program selected for the calculations was DYNA3D [1]. Shell elements were used to model the steel drum and the glass bottle. Solid elements were employed for the shock absorbing material as well as for the waste inside the bottle. The target was represented with a rigid stonewall.

Although different models were created for studying impacts with different symmetries, the model shown in Fig. 2, composed 576 shell elements and 1350 hexahedral solid elements, makes use of only one plane of symmetry. For scale, the height and diameter of the steel drum are 0.84m and 0.57m, respectively.

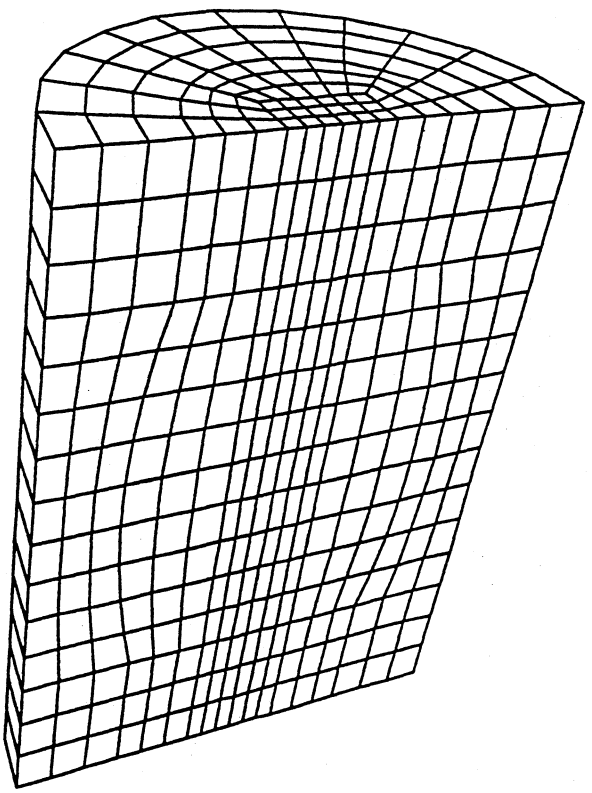


Fig.2 Mesh used to model the skel drum, shock-absrbing material, glass bottled and radioactive contents

As for the behaviour of the various materials, the contents of the bottle were assigned water properties. Glass was assumed to behave as an elastic-brittle material. The steel drum was represented with an elastic ideally plastic material with a von Mises yield surface. A similar type of constitutive behaviour, albeit with different properties, was assigned to the shock-absorbing material. In this latter case, preliminary hand calculations were used to provide initial estimates for the desirable properties, later followed by more detailed numerical analyses.

Figures 3 and 4 show some results corresponding to one of the more

demanding cases: a corner impact following a free drop from 9m, in pure translation and with the centre of gravity directly above the impact point. The problem is made harder, even in physical terms, because of the very large deformations which necessarily ensue an energetic corner impact. Numerically, additional demands arise from the fairly large duration of the problem (much longer than a base or side impact); the long duration is associated to the relative softness of an impact in which the contact area is growing slowly from an initial zero value.

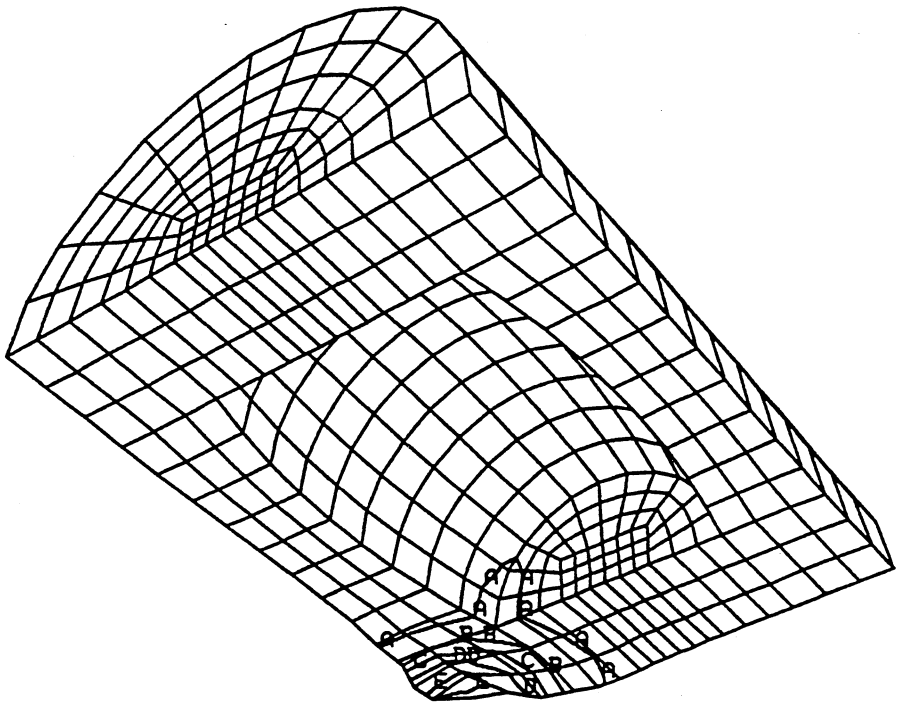


Fig. 3 Contours of equal plastic strain (A=0.1, B=0.2, C=0.3, D=0.4, E=0.5)

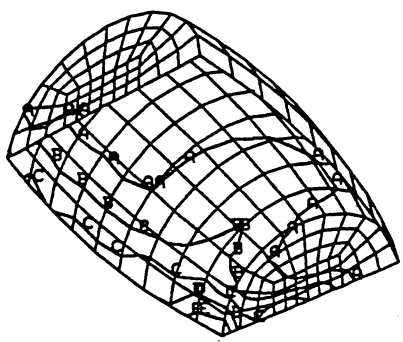


Fig. 4 Contours of equal effective stress in the glass bottle at peak values (A=10MPa, B=20MPa, C=30MPa, D=40MPa, E=50MPa)

Fig. 3 shows values of plastic strains occurring in the shock absorbing material (the steel drum, the glass bottle and its contents have been removed in this plot) at the end of the impact event. Fig. 4 presents contours of effective stresses in the glass bottle at the time when maximum values are reached.

The number of time steps used for integration in this analysis was on the order of 5,000. The Convex C-220 utilized under 15 CPU minutes in

completing the calculations.

The analyses conducted finally allowed demonstrating the feasibility of the proposed concept and determining the properties of a material which would ensure physical separation between bottle and drum while maintaining the bottle stresses below the glass strength.

5. SPENT FUEL FLASK

After leaving the reactor, nuclear fuel is sent to the spent fuel pool. Some time before the latter is filled the fuel must be transported, either to a reprocessing plant or to a storage facility, be that temporary or permanent. All these activities require the use of various types of storage and transport casks.

A spent fuel cask must fulfil a number of conditions. From a mechanical point of view, the design requirements include a satisfactory performance during and after a series of postulated accidents. This performance refers, among other considerations, to preserving the structural integrity of the cask, maintaining the isolation of its contents and ensuring that the fuel does not sustain unacceptable damage.

In the present Section, an example is presented of a cylindrical cask undergoing a postulated storage accident. The cask falls unprotected on a rigid plane from a height of 37cm (15in). The impact occurs on the edge of its base and with the centre of gravity of the cask directly above the impact point.

The cask is primarily made of stainless steel, although it also contains lead as gamma ray shielding. The cask is about 5.0m long, 2.2m in diameter and has an overall weight slightly in excess of 1MN. Two bolted lids are used to close the cask. For reasons of space, it is obviously impossible to provide here details of the structure of the cask, fuel basket, etc.

Two codes were used in the dynamic analyses of the cask: DYNA3D [1] and ABAQUS/Explicit [2]. The plots presented here correspond to the DYNA3D calculations.

Fig. 5 presents the mesh used to model the cask, the two lids and the bolts. The mass of the contents was smeared along the inner surface of the cask. The model, which has 4116 elements and 5385 nodes, includes frictional contact surfaces between the lids and the walls of the cask. A 0.1 friction coefficient was assumed to be adequate for those contacts.

Elastic plastic models with hardening were generally used for representing the materials involved (various types of stainless steel, lead, neutron shielding, etc.). The impact calculations were preceded by thermal analyses in order to provide the existing temperature distribution at the time of the impact.

The impact duration is long, again due to the negligibly small initial contact area; it lasts about 23msec. Two vertical velocity histories are shown in Fig. 6, corresponding to the lowest and highest points in the cask. The gradual deceleration process of the top of the cask is evident in this figure, eventually leading to its rebound when the lowest node acquires a positive velocity. Fig. 7 presents two close-up views of the

contours of accumulated plastic strain which, as can be seen, may approach 25% very locally at the impacted edge. No problems are caused in relation to the cask integrity, since about half of the wall thickness remains below the 0.2% strain level even at the impact location.

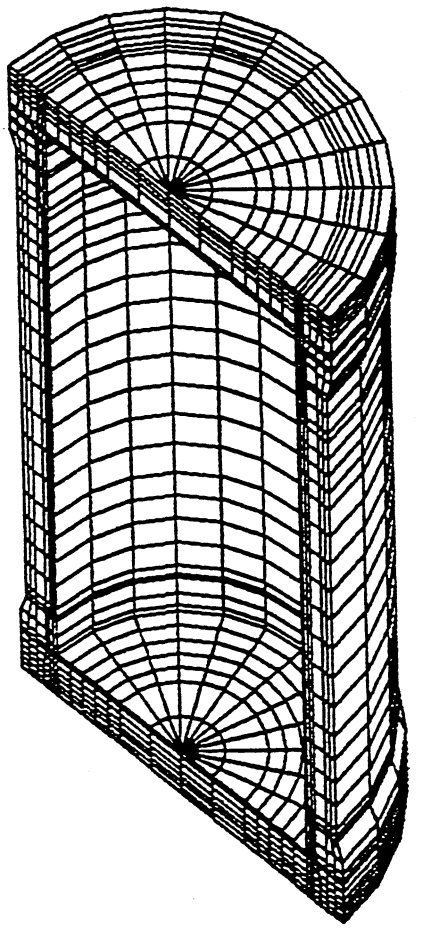


Fig.5 Mesh used to model the complete cask with bolts and lids

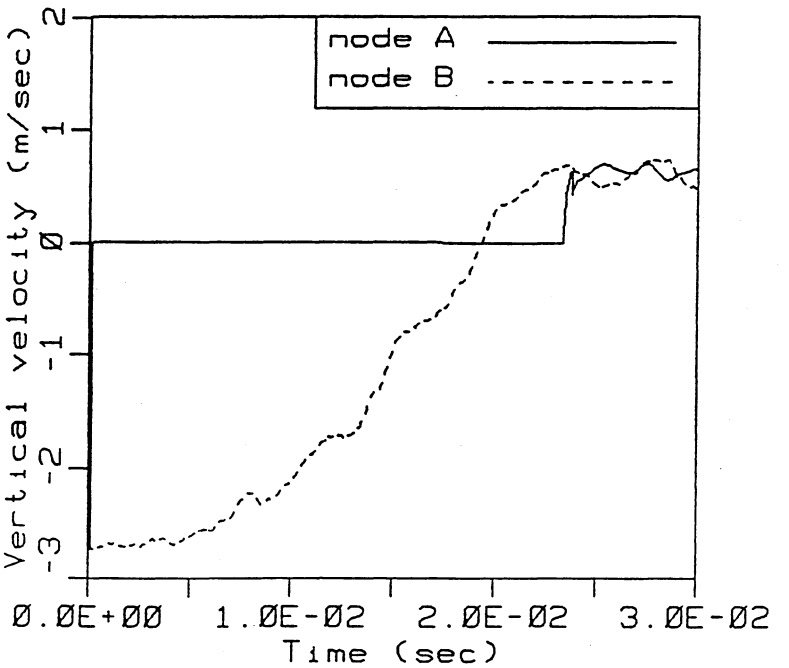


Fig. 6 Velocity histories of the lowest (A) and highest (B) nodes in the mesh

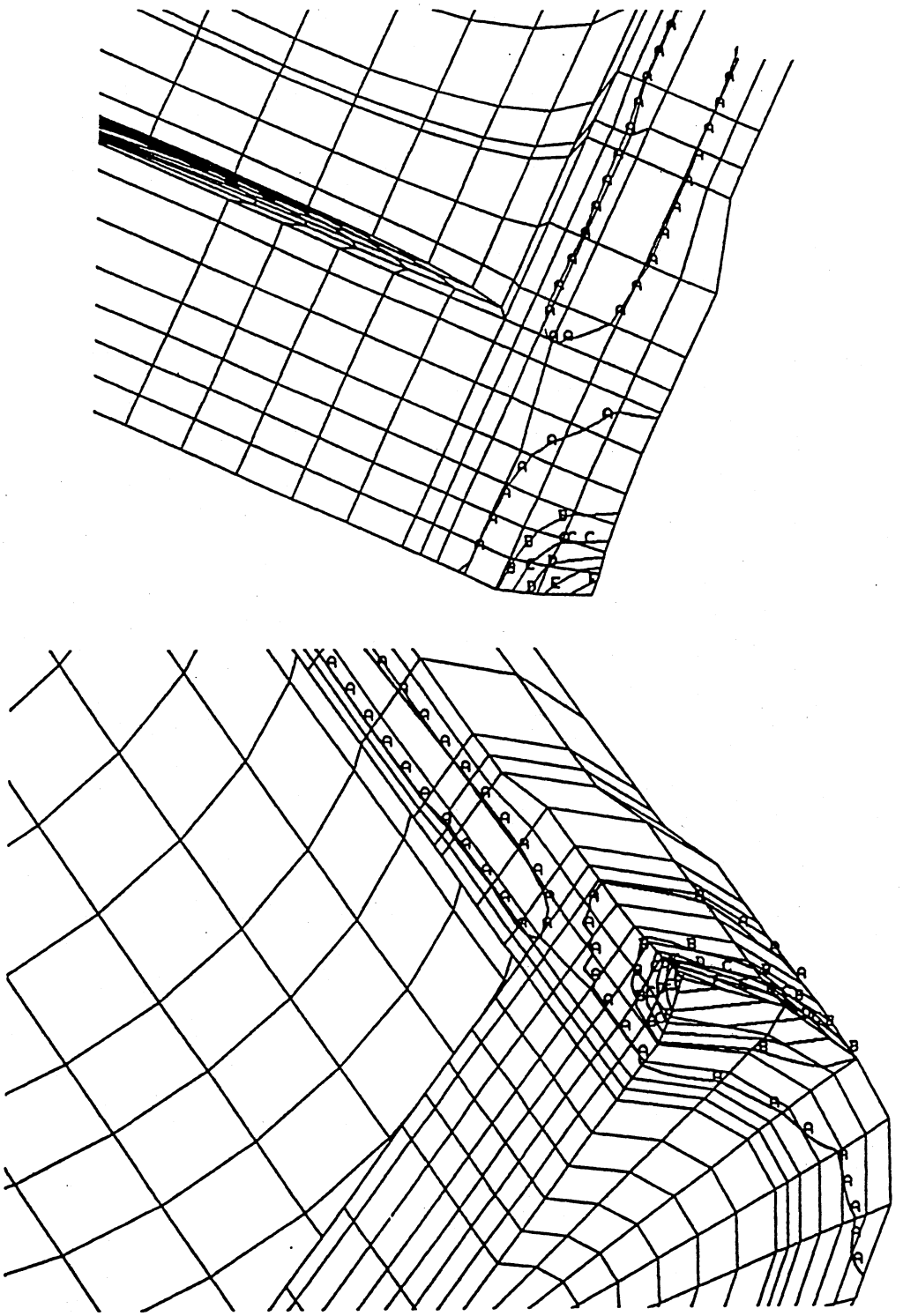


Fig. 7 Two close-up views of plastic strain contours (A=0.002, B=0.05, C=0.10, D=0.15, E=0.20, F=0.25)

While this type of corner impacts are rather demanding on the cask structure, the corner crushing acts as a shock-absorbing mechanism. Thus, peak rigid body decelerations for the cask are only about 25g and the bolts give no reason for concern in this particular impact.

Overall, slightly over 20,000 integration time steps were required for the analysis of this problem, consuming about 3 CPU hours in a Convex C-220.

6. RIGID MISSILE PENETRATION INTO CONCRETE

Various types of accidents, such as the failure of a turbine rotor, are able to produce highly energetic, relatively rigid missiles. These missiles then interact with structures and it is of obvious interest to evaluate the potential consequences.

Impacts by rigid missiles, when they result in considerable penetration of the target, are particularly difficult from a numerical standpoint. This is primarily due to the very large deformations caused to the target in the area directly affected by the projectile. These very large deformations eventually lead to tangling of the mesh, negative volumes of elements and, in short, the impossibility to pursue the calculations. To deal with this problem, the main techniques currently available are rezoning and erosion algorithms. In the first case, new regularized meshes are constructed with varying degrees of participation from the analyst; then, all variables are mapped from the old to the new mesh and the calculations proceed based on the new mesh until rezoning is again necessary. In the second case, elements which have distorted beyond some user-specified limit are simply removed from the mesh. Each procedure has a number of advantages and at least as many disadvantages. The calculations presented here employ a rezoning algorithm.

The problem considered is the normal penetration by an inert military projectile on a reinforced concrete target [4]. The projectile has a mass of 547kg, length of 2.0m and diameter of 0.4m; the nose is ogive shaped and the impact takes place at a velocity of 250m/sec. The reinforced concrete target has a thickness of 2.0m and an unconfined compressive strength of 45MPa.

Since the problem has axial symmetry, only a two-dimensional analysis is required in this particular case. The program selected for the calculations was DYNA2D [5].

The mesh used to simulate the problem is shown in Fig. 8. Solid elements have been used to represent all materials, including the inert explosive inside the projectile. Only one element across the steel shell is necessary, since it is known from the experiments that the deformations in the steel are very small and without any tendency to initiate buckling or other instabilities. The mesh shown in the figure actually has twice the number of elements of that used in the calculations since a reflection has been added to improve the visualization. The real mesh is composed of 556 nodes and 483 quadrilateral elements.

Although a number of different analyses were conducted using different constitutive assumptions for the reinforced concrete, only one is presented here for brevity. The concrete model provided in DYNA2D has been used together with a mixture rule for deriving stiffness and strength properties for the composite concrete-reinforcement material. A 7% reinforcement was assumed to be present in the outer 16cm of the concrete target.

As mentioned earlier, it is necessary to rezone the mesh from time to time because of its large distortions and their consequences on both accuracy and the size of a stable time step. Two rezones of the mesh were typically

sufficient in each of the analyses conducted.

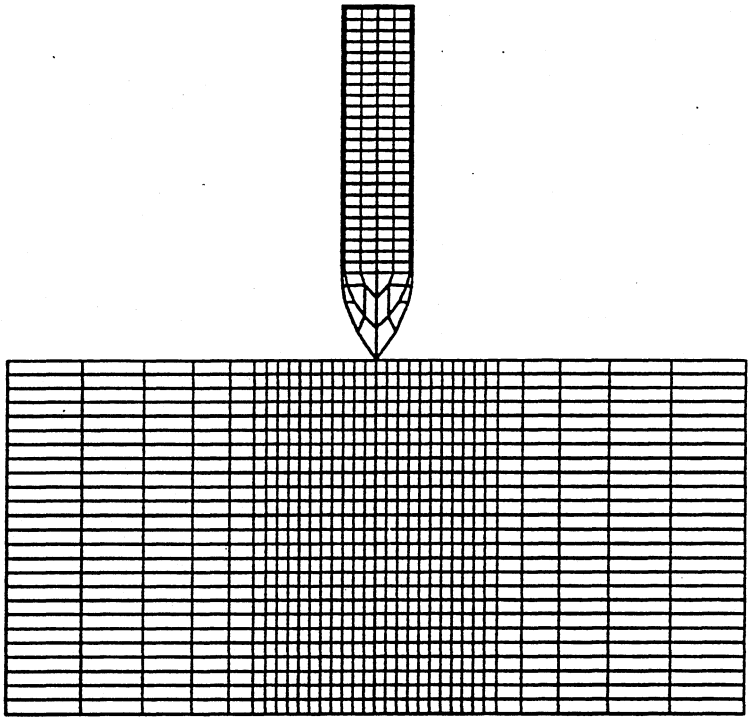


Fig. 8 Mesh used to model the projectile and the target

The global evolution of the problem can be followed by inspection of Fig. 9, which presents the histories of vertical velocities at four points, two

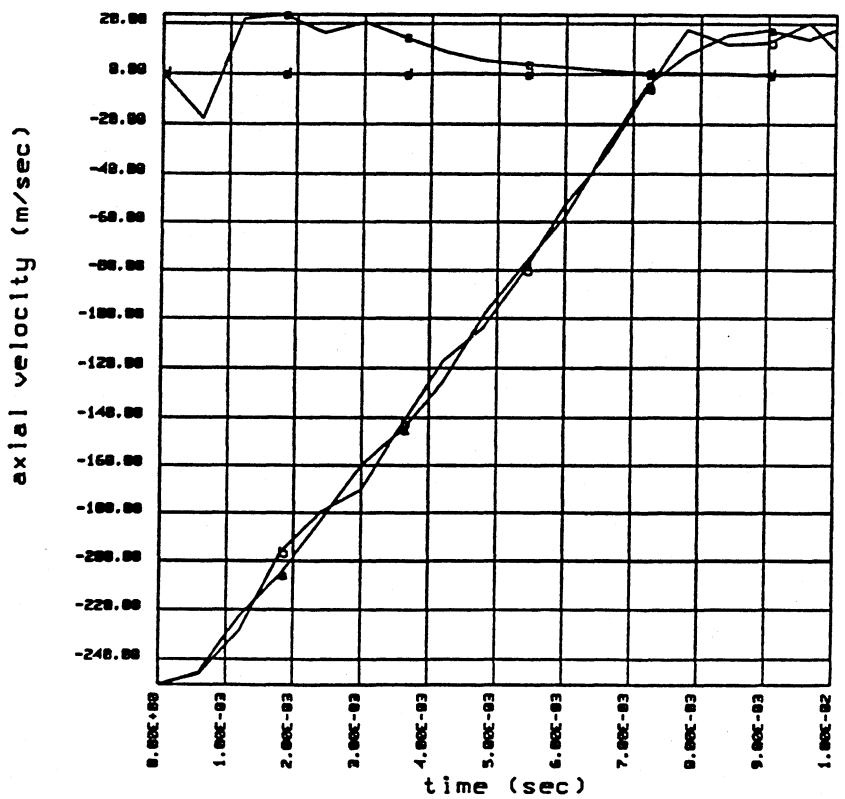


Fig. 9 Velocity histories of two points in the projectile (A, B) and two in the target (C, D)

in the projectile and two in the target. As can be seen, the projectile deceleration is almost linear, which indicates that the force developed in the penetration process does not change very much with the depth of penetration. The history culminates with a slight rebound of the projectile, after about 1m penetration, some 7.4msec after the initial contact was developed. These numbers compare well with the experimental observations.

Additional details are shown in Figs. 10 and 11, where contours have been plotted for plastic strains and pressures, respectively, after 4.8msec; at this time, the penetration process is still underway. Fig. 10 clearly indicates the narrowness of the large strain region caused by the penetration. The typical bulb of high pressures ahead of the projectile is depicted in Fig. 11.

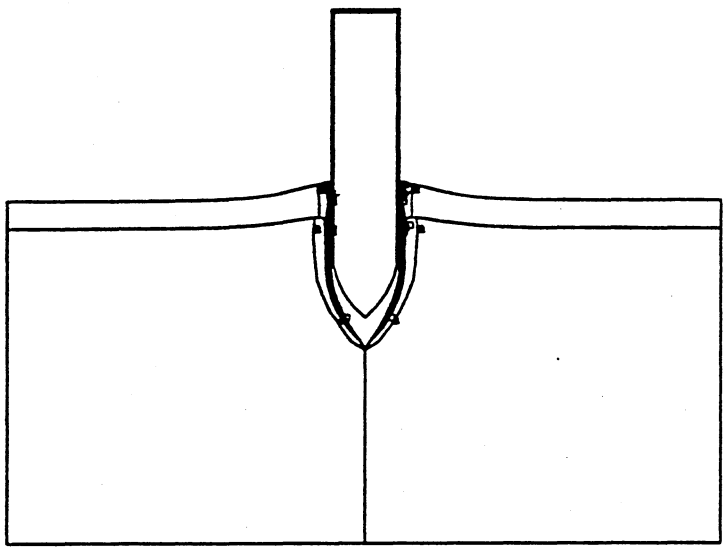


Fig. 10 Contours of equal plastic strain (A=0.5, B=1.0, C=1.5, D=2.0)

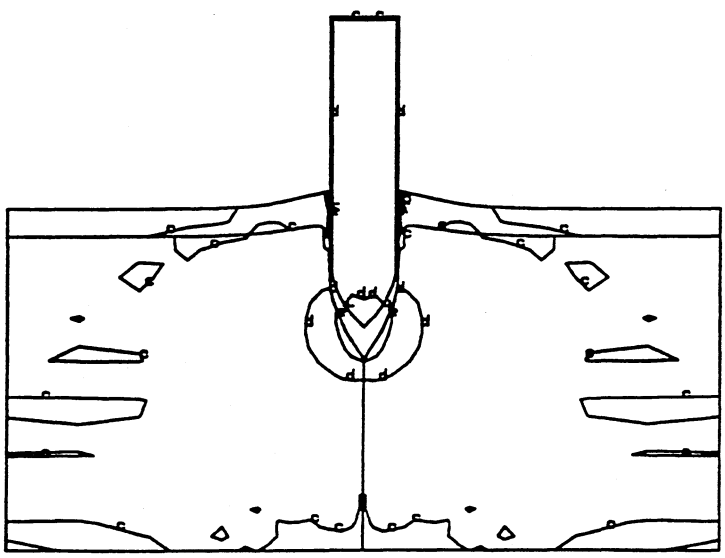


Fig. 11 Contours of equal pressure (A=-200MPa, B=-100MPa, C=0MPa, D=100MPa, E=200MPa)

The number of integration time steps needed to span the duration of the problem was in this case close to 15,000. However, being only a two-dimensional problem, about 8 CPU minutes of Convex C-220 were sufficient to complete the calculations.

7. CONCLUSIONS

In the present paper, several aspects of impact problems have been discussed. The speed, flexibility and simplicity of Lagrangian methods with explicit integration make them the obvious choice for numerical simulation of such phenomena.

Three different problems have been mentioned, with impact velocities ranging from a few to hundreds of metres per second. Also, a variety of computer programs has been used (DYNA2D, DYNA3D and ABAQUS/Explicit), albeit all of them within the explicit Lagrangian family.

The results were in all cases very satisfactory, both in the performance of the programs and in the capabilities of the Convex C-220 computer in which the analyses were conducted. With the exception of the contact logic, for which it is difficult to reap benefits from vector/parallel architectures, the rest of the operations involved in impact problems are implemented naturally and efficiently in this type of architectures.

8. REFERENCES

[1] Whirley, R.G. and Hallquist, J.O. (1991). "DYNA3D: A Non-linear, Explicit, Three-Dimensional and Finite Element Code for Solid and Structural Mechanics - User Manual". Lawrence Livermore National Laboratory, Report no. UCRL-MA-107254.

[2] Hibbitt, Karlsson and Sorensen (1992). "ABAQUS/Explicit User's Manual". Version 5.2.

[3] IAEA (1980). "Regulations for the Safe Transport of Radioactive Materials", International Atomic Energy Agency, Safety Series no. 6.

[4] Johnson, G.R., Stryk, R.A. and Nixon, M.E. (1988) "Two- and Three-Dimensional Computational Approaches for Steel Projectiles Impacting Concrete Targets" in "Impact: Effects of Fast Transient Loadings", by Ammann et al (eds.), Balkema, Rotterdam, pp. 233-249.

[5] Whirley, R.G. and Engelmann, B.E. (1992) "DYNA2D: A Nonlinear, Explicit, Two-Dimensional Finite Element Code for Solid Mechanics. User Manual". Lawrence Livermore National Laboratory, Report no. UCRL-MA-110630, April.

SHOPPING FOR THE MACHINE ROOM OF THE FUTURE

Jerry McAllister
Michigan State University, Mi, USA

Abstract

In response to the increasingly complex and rapidly changing computing marketplace the Michigan State University Computer Laboratory is working on creating a vision of the computing services it will provide and how to purchase and develop the resources it will need. This vision has come locally to be called "The Machine Room of the Future". This presentation will discuss creating the model, developing a plan and a bid for it and comment on the response from vendors.

It is similar to, but intentionally even more generic than, the now familiar NCSA meta computing model in that it proposes diverse resources from multiple vendors interoperating with users easily accessing whichever resource is optimum for a particular operation.

First a disclaimer

This paper discusses my observations of the process and experience of the bid process. Neither I nor this paper in any way speak for Michigan State University or the MSU Computer Laboratory or its advisory committees. Since the process discussed is still in progress many of the statements reflect plans and expectations for the future.

Overview of Michigan State University Computer Laboratory central site computing services

The Michigan State University Computer Laboratory is a separate University department charged with providing computing services to the general University community. It is overseen by several committees made up of representatives from other departments on campus.

The Computer Laboratory at Michigan State University provides and manages a wide variety of computing services for the general University community ranging from mainframe services on the Convex C240 and IBM 3090 to many microcomputer workrooms around campus that are open to the public. The Computer Lab also manages campus wide network services that link nearly every computer on campus as well as communicating with the outside world. There are more than 5200 on-campus IP hosts connected to the network. They are a wide variety of mainframes, minicomputers, workstations, PCs and Macintoshes. Of these about 2600 are owned by the Computer Lab directly. The rest are owned by other departments and individuals.

Many of the on-campus systems are workstations in offices and research laboratories. Users have been encouraged to use Computer Lab resources particularly the Convex C240 for such things as communication, CPU intensive computing as well as backup of their PCs and workstations.

The Michigan State University Computer Laboratory central site where the Convex is installed houses several different and dissimilar systems including IBM, DEC, Convex, BBN, Workstations, PCs and various network management services. These are all monitored around the clock by operators who are trained in operation procedures but who generally do not have programming or system management training or experience.

There are charges for all Computer Lab mainframe services. They are divided into CPU use, memory use, connect time, disk use and tape mounts. The charges are adjusted so that the "same" job will incur approximately the same charge if it is run on any of the systems. The charging scheme has two purposes; to generate revenue that is used to replace and upgrade resources and to encourage efficient use and discourage waste and misuse. This charging scheme is oft discussed and debated at MSU because some factions are concerned that there are projects which are worthwhile but poorly funded and cannot afford the needed computer resources at the central site and therefore are shut out of access to computing services.

The most active users of the MSU Convex C240 are from the Physics department who run large FORTRAN coded CPU intensive applications. They provide an infinite job queue and tend to need a large amount of disk space. Other major users are from the Engineering and Chemistry departments. There are few general purpose users on the MSU Convex. Although it is not restricted on the Convex, general purpose usage is steered toward the IBM, VAX and general purpose workstations.

Discovering the problem

It is not a new idea to say that in today's computing marketplace no single vendor is able to provide all computing resources that might be desired nor is any single vendor line always the best in any particular field. Both the computer marketplace and the needs of users are changing and growing so rapidly it is hard to even stay informed of the current state of the art. And, of course, everyone's budgets are constantly being tightened making it even more critical to make optimum choices when shopping for computing resources.

The Michigan State University Computer Laboratory is wrestling with this situation just as are most other computer users. At the MSU Computer Lab it may be even more confusing than at most sites because of the wide variety of services that are provided and the even greater diversity of its users.

The old way - Vendor dependence

In the past the lab has selected a vendor for a major system and that system and vendor determined the resources that could be made available to the users and also the path for upgrades.

- Bad flexibility

If the vendor did not offer a resource, generally that meant not being able to offer it to Computer Lab users or in some cases creating and supporting the resource locally. Large vendors are not usually willing to commit their money to developing a service unless they see a significant market for it. Our small installation does not carry enough weight to cause vendor development. In addition, it was often difficult of impossible to integrate third party resources into the system.

- Difficult to switch vendors

When it became obvious that it was necessary to switch vendors in order to meet the needs of the Computer Lab's users, everything from the previous vendor was replaced including peripherals, operating system, operations procedures and the computing environment for users, even things that were still useful had to be replaced. New procedures for operations had to be created and users were forced to port their applications to the new system.

- High cost

Being bound to one vendor meant buying peripherals and software only from that vendor. Large vendors had no incentive to keep prices low and we were usually unable to take advantage of lower cost third party peripherals. This kept costs to the Computer Lab high and necessitated higher rates for users. High cost also meant that the amount of services that the Computer Lab could afford to offer was significantly reduced.

- Redundancy

Still later, when the Computer Lab found it necessary to support more than one vendor system, everything had to be duplicated including peripherals, and operations procedures for each system.

This redundancy and waste of resources, money and both staff and users' time had to be eliminated.

The new way - Vendor Independence

To get out of this trap it is necessary to become vendor independent. Although there is a significant amount of initial work, in the long run the MSU Computer Lab expects to experience several benefits from its effort to become more vendor independent.

- More flexible

The Computer Lab will be able to provide a variety of services from a variety of vendors. When shopping for new resources, it will not be necessary to be bound by currently supported vendors or face major difficulty in switching over to different systems.

- More efficient

The Computer Lab will be able to provide services of different types from which ever vendor is currently the best provider without having to duplicate common resources such as file servers. When moving to a different vendor, it will not be necessary to "throw away" resources such as common peripherals that are still providing good service.

- More consistent user environment

By specifying a single operating environment and minimum services from the operating system (UNIX) across all systems the users can be assured that their applications will move easily from one compute resource to the next without a lot of wasted time and effort. Users will feel free to use any of the resources available depending on which runs their application optimally and which has the highest availability at a given time.

- Easier to stay current

Because the Computer Lab will be free from any particular vendor's development schedule and because it will be possible to upgrade or add only what is needed, it will be easier to track the latest changes in the marketplace.

- Lower prices

It will be possible to buy on the open market and look for the best price/performance available, especially for peripherals from any major or third party vendor that supplies the resource. This also is expected to make the major vendors bid somewhat more aggressively.

- Better interoperability

By specifying minimum standards of operation and protocols for communication for each resource that each vendor must support and eliminating any products which fail to provide the support it will be possible to ensure that all resources in the machine room can work optimally together. It has a side effect of defining the range of services and standards which are supported by the Computer Lab. This will also help users make better informed decisions about their purchases on equipment that must communicate with Computer Lab resources.

The task was upgrading our process of purchasing and developing computing resources for the MSU Computer Laboratory was undertaken by the Computer Lab oversight committees in the course of creating a bid for future computing resources. Since the MSU Computer Lab is already managing several dissimilar systems which interoperate only in very limited ways and since MSU has had significant experience with systems development the risks seemed well under control.

The big picture

The first step was to create an overview of services and resources. Review what is being done and what is needed and where it is going. Do this with an awareness of what is available in the marketplace.

- Mission and policy

It is the policy of the MSU Computer Lab to exercise the least amount of control possible on the work that is done on the computer systems. In general, if the user is able and willing to pay for use of the resource, no restrictions on amount of use are imposed. Because of this, the range of services that are demanded by users is very wide, especially in relation to the small size of the Lab's operating budget.

- Types of work

The MSU Computer Lab supports a very heterogenous set of resources that are all interconnected via networks and are running a diverse set of applications many of which are "own code" written in mostly FORTRAN with a mixture of C. The applications range from highly vectorizable and parallelizable to highly scalar. Some use large amounts of disk but most are highly compute intensive with only small amounts of I/O.

- Which resources are needed

As a result of the "self examination" and evaluation the needs of computer Lab users were seen as falling into eight areas; General purpose computing, Scalar processing (including scalar clusters), Disk server, Archive server, Vector processing, Parallel processing, High speed interconnect(backbone) and an Interface to the campus network.

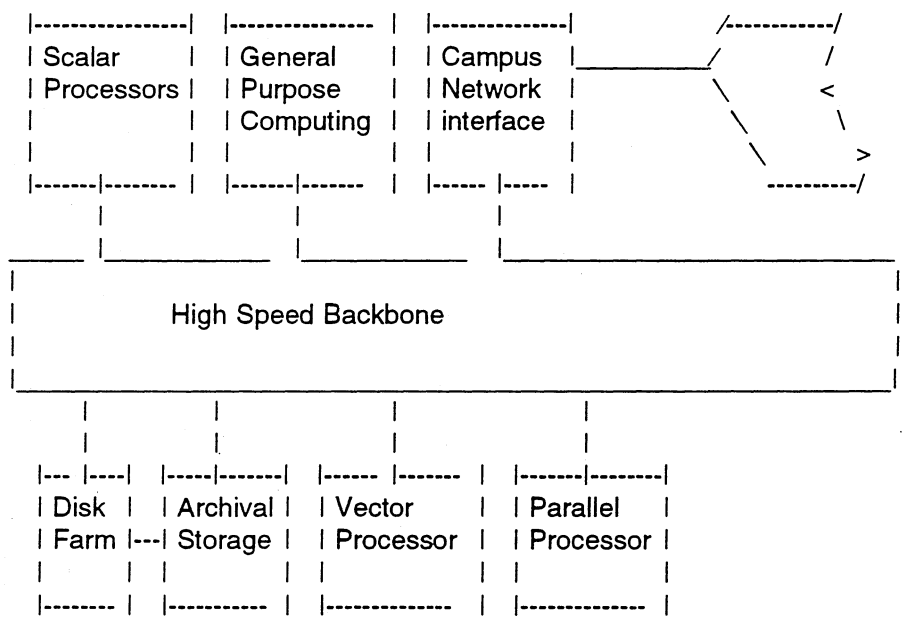
- Operating environment

For all but the general purpose computing, the only operating system that would provide enough interoperability and portability is UNIX.

- The machine room of the future

The reference model that was created which has come to be called "the machine room of the future" is blends the concepts of a multiplicity of services, interoperation of all systems and vendor independence. It is similar to, but intentionally even more generic than, the now familiar NCSA meta computing model in that it proposes diverse resources from multiple vendors that interoperate seamlessly so that users can easily access whichever resource is optimum for a particular operation.

The sketch drawn to illustrate the model has come to be called the big picture. It merely contains generic boxes for each category of resource with no mention of a particular vendor.



Define each resource and service

Create a specification of the minimum capacity and functionality for each resource.

Keep the specifications within the realm of the possible.

Members of the oversight committees each adopted a particular area to explore and become familiar with what was currently available in the marketplace including what was being announced at trade shows such as Supercomputing. This information was shared and compiled along with estimates of actual need into a set of minimum specifications and questions about each type of resource.

Since these will change and grow with the market and since each site will have its own unique needs, it would not be helpful to detail the specifications here but some general examples are given. Note that these are minimum values for consideration in the current bid and that these are only a small sample of the list of minimum requirements for each category. The bid also included recommended options for each category.

- Scalar processing (including scalar clusters):
Minimum published peak performance of 50 SPECint92 and 50 MFLOPS per CPU. Minimum capacity of 2 Gigabytes of virtual address space per processor and configuration of 128 Megabytes real memory.

- Vector processing:
Minimum published peak of 50 SPECint92 and 125 MFLOPS per CPU in 64 bit dyadic performance. Minimum configuration of 512 Megabytes of real memory. Minimum software includes C and FORTRAN compilers, support for files larger than 4 Gigabytes and support for parallel processing if a multiple CPU system is bid.

- Parallel processing:
Minimum of 16 CPUs, 20 CPECint92 and 20 MFLOPS (64 bits) per CPU. Base architecture expandable to a minimum of 1 Gigaflop overall performance. Minimum of 40 Megabytes per second bandwidth between two adjacent processors.

- Disk server:
A minimum processor speed of 60 SPECint92 scalar performance and 256 Megabytes of real memory. The ability to easily add and support third party disk. NFS is required and AFS is highly desired (because most vendors had not announced their plans for AFS at the time of writing the bid document it was not made an absolute requirement. It should now considered to be required.) High speed connectivity and RAID support are examined in detail as well.

- Archive server:
Possible tape technologies are DAT, Exabyte(8mm), 3480, VHS, D1, D2 and even 1600/6250 nine-track. The nine-track tape is included to allow transfers to other sites. Questions are included to examine host interface requirements and robotic tape handler support.

- High speed interconnect(backbone)

Eligible technologies are FCS, ATM and HIPPI. There is an expectation that FDDI will be used for some period as an interim until the technology picture clears a little more.

- Interface to the campus wide network

TCP/IP and NFS are required protocols.

- General purpose computing

Because it is not possible to change everything at once or buy everything that is desired, general purpose computing has for the present been kept on the IBM 3090. There are still several applications being run that depend on that architecture and it is not possible to remove that system in the near future so the general purpose work will help keep it busy for now. At some time in the future, general purpose use is expected to move to one or more of the scalar UNIX systems.

What we don't want

The committees also spent some time discussing what we do not want to hear from the vendors.

- Packages binding us to one vendor: Although vendors are welcome to offer special package discounts or enhancements if they choose, we are not interested in any "deals" that would tend to bind us to future purchases from that vendor or that would require inclusion of things we don't want or need.

- Vaporware: Although we allowed nondisclosure discussions of future product plans, we will not commit to purchase of any resource until after our own benchmarks are run on the system under consideration.

- Marketing fluff: Every vendor loves to spend hours extolling the virtues of the company; how the customer is their top priority; how they are the market and/or technology leader, etc. Very often their graphics and text for this area are far more sophisticated than for the far more important technical information. More than about 15 minutes of this is generally a waste of time and we counselled every vendor account representative in this regard - with widely varying degrees of success.

Create and execute the bid

- A timeline was created for the process including writing each section of the bid document, publishing the bid, required responses and benchmark results from vendors and evaluation periods. Except for the dates for vendor required responses all target dates were kept internal to the committee.

- Write the bid document RFP/RFQ with separate and distinct sections for each category of resource. Separate minimum standards and sets of questions were written for each category of resource.

- Vendors were encouraged to write separate responses for each section and to price each resource separately. If they wanted to assemble an _additional_ package deal with price consideration, that would be on their own initiative.

- Create benchmarks and evaluation standards and procedure.

The MSU Computer Lab has assembled a suite of benchmarks that represent actual applications that our users are currently running and intend to continue to use. The suite also includes several benchmarks intended to discriminate between various aspects of performance. Although we look and published numbers such as Linpack and SpecINT and SpecFP our final and most important performance evaluation is based on our suite of real applications. Since every installation will have different applications and since our benchmarks create relative comparisons and not Megaflops or some other seemingly objective number it is inappropriate to include details on specific results here. The important issue is that we have created an evaluation procedure that is consistent across all systems of any category and that represents real work that the system will have to do.

To improve accuracy and clarity a standard form was designed for reporting benchmark results. The form indicates which resource category relating to which section of the bid document and which benchmark it covers. A separate copy of the form was required for each benchmark for each system bid by each vendor. Vendors were also required to return a tape with all of the benchmark data including source as run.

Hold vendors (and ourselves) to it

After the bid went out, we spent time with the sales representative from each vendor who indicated an interest in responding carefully explaining our newly created philosophy. We made sure the benchmark procedures were clear and tried to make it clear what we don't want to hear as well as what we are looking for.

We insist that each vendor provide a single person as their point of contact and designate such a person from the Computer Lab. This person is welcome to refer questions to someone else, of course but they must always be aware of every communication.

Because of the fast moving marketplace nowadays, we did something this time (with mixed results as far as it being truly useful is concerned) that has never been allowed before. We allowed each vendor to include a non-disclosure section presentation to their formal presentations. The formal bid was, as always due to our legal requirements, read in public and made available in the Computer Library for the public to review. When each vendor came to make their presentation, the formal bid and whatever else the vendor wished was discussed in the public portion of the presentation and then the committee moved to a closed session for any non-disclosure information that the vendor chose to discuss.

Since the budget will not allow every resource to be purchased at one time, the non-disclosure discussions did help choose which decisions to make first and in some cases there was much better discussion of technical issues of architecture and design that helped us understand the performance results we were seeing (although in some cases some vendors' nondisclosure presentations seemed to actually give less information than their public discussions - they seemed to want to throw in that marketing fluff that they took out of the public session because of our advance warning).

General observations and impressions, both positive and negative

Our main observation has been that although there is a lot of talk about standards and vendor independence in the industry, there is still much reluctance to actually work that way. Most vendors seemed confused by the bid document and asked questions like how we wanted them to package their products to cover all of the categories. It took a lot of discussion to convince them that it was truly expected that they respond to each section separately. Even when vendors did write separate sections correctly, they tended to try and price things in packages, sometimes with unwanted additions and not necessarily even giving any additional price advantage or improved service for the package. The committees had to break apart some of the bid responses and try to guess what some of the resources would cost in a stand alone sale (which, of course, was only used for internal discussions. An actual price quote would be required before any decision was made).

Although vendor independence / standardization is talked about a lot in trade shows like Supercomputing, and in spite of all the standards groups in which every major vendor participates, in real life it is not a popular concept with vendors. Vendors are, not surprisingly, reluctant to do anything to give the competition a break.

Actively pursuing vendor independence does not always seem to be popular with computer users either because it requires them to actively take charge of the system, select products and integrate them which is a lot of work.

There is an interesting risk in pursuing the vendor independent plug and play philosophy. It is possible for the market to get into a rut similar to that in the PC marketplace where vendors have catered to the lowest common denominator and have then found themselves forced to continue obsolete defacto standards that make upgrading their products difficult or awkward. That is showing up in areas such as the 2 Gigabyte file system and file size limit and in the number of UIDs that can be created on a system. Both of these issues and others are important at MSU. So far, each vendor is going its own way for solutions still claiming to be the most standard.

Recommendations to users and to Convex

- Take charge of your own installation resources.
Do your homework and learn what is available in the marketplace, what your site needs and how different resources need to communicate.

- Write vendor independence and interoperability into bids. You cannot depend on vendors to do it for you.

- Discipline yourself to let marketing fluff go in one ear and out the other.

- Insist that vendors respond the way you want.

X TERMINALS AT CSN NETWORK

Maria Luisa de la Cruz
Consejo de Seguridad Nuclear

ABSTRACT

The CSN Network has 4 UNIX* hosts with different applications running in them. There are 42 X terminals in the network the network description and X terminals configuration are described.

THE NETWORK

The CSN has an Ethernet network installed. It is used as a LAN with 5 servers for graphic, electronic mail, word proccesing,... etc. The servers are personal computer with operating system OS/2. The hardware and software come from 3COM* Company (whith the exception os some servers from HP*, COMPAQ*). A TCP/IP network coexits for acceding to the differents computers and Data Base servers.
The network has 8 different sub-networks which are linked by 2 CISCO* routers. The hosts sub-network is the main one. The CONVEX* C-220, DATA GENERAL AviioN* 4625, DEC* 5500, SUN* IPX and DECpc* 450st are installed in it. These routers are used for routing the data from one sub-network to another one, mainly from the personal computers and X terminals to the hosts sub-network.

The network adresses are type "C". The CSN network has 250 personal computers and 41 X terminals installed in the 8 sub-networks.

The personal computers have a 3COM* board in order to connect them to the network. The software installed, on the personal computers, is also from the 3COM* Company. The software is TCP/IP, TELNET(with a vt100 emulation) and FTP.

THE X TERMINALS

This is not the situation that existed in the CSN 3 years ago. There was only one TCP/IP network. There was only one TCP/IP network, the mainframes were non-UNIX* machines and there was not X terminals installed.

In December 1990 it was installed the CONVEX* C-220 and 4 X terminals(one of them as the console). In that moment the users demanded personal computers for connecting to the CONVEX(*). The X terminals technology was installed whitout any problem.

After the first monts of working there was the first orders of X terminals. In 1991 16 X terminals were installed.

In 1992 14 X terminals were installed and during 1993 11 has been installed.

The models have have changed from NCD17c* to the latest MCX* from Network Computing Devices, INC.. It has tried to look for the best solution for the users in each moment acordig of the state of the art.

The memory configuration has changed from 4MB for the first models to 6MB or 10MB configurations depending on the graphics applications.

At first the X terminals were loaded from the CONVEX* C-220, the boot and configurations file were on /usr/tftpboot directory. It was used the 2.0 NCDware* software version.
The sofware was upgrade to 2.1, 2.2, 2.3 and 2.4 NCDware* software versions when each version was released. Each new software version had new utilities and solved the problems of the older version.

After the SUN IPX* workstation installation the latest NCDware software was installed on it.The 3.0 NCDware software version has a lot on new features. The MCX* X terminals and some of X terminal from Data Center Departament booting from the SUN* IPX.
The reason of this configuration was to test if it is possible to have redundancy between the clients (CONVEX, SUN, DATA GENERAL, DEC). Soon,it was discovered that the migration between NCD 2.4 software version and 3.0 was not so trivial because it was very hard to change the configuration files from one version to another. After some tests it was decided that the CONVEX users would keep up their boot and configuration files in the 2.4 version in the CONVEX. The Data Base and SUN users would have their files in the SUN* IPX, they will use the 3.0 version.
After 3 months of testing the new version it will be use for all X terminals.

It was defined as public utilities : xclock, xload, xpostage, xcalendar, xcalc, xman, xedit. There is another utilities set such X11/unsupported that are not considered interesting as being public utilities and it has been deleted.

There are several programs that use X interface such as : DISSPLA*, Simone*, Isovu*. Several applications developed by the own CSN and several programs developed for the CSN by CONVEX SUPERCOMPUTER S.A.E. Company.

There another applications : CASE tools, SAS* that are not running in the CONVEX. These applications are in another computers on the network. How can the users running these applications? Executing an xterm call to the required computer and setting the enviroment variable DISPLAY. From this moment it is possible to run this aplications with the X interface. With this simple mecanish is possible to run all standard X applications such xclock, xload ... at diferents hosts. By example it is possible to run xload at CONVEX, DATA GENERAL and DEC and knows the load of all the hosts.

CONCLUSION

The X terminals in the UNIX* environment provides a simple interface to the operating system and many applications.

The users are delighted of working with such simple interface and they are not worried about the need of making backups everyday. The system administrator tasks such as configuration, testing,... are easier with X terminals than with another type of equipment. The system security is increased using the 3.0 NCDware* software version because it has new security enhacements.

* 3COM, HP, COMPAQ, DATA GENERAL, DEC, SUN, NCD, CONVEX, Unix, CISCO, SAS, DISSPLA, Simone, Isovu, or their products : are trademarks.

NET-WIDE USER DATA ADMINISTRATION SYSTEM

Elisabeth Dregger-Cappel, Detlef Lannert, Norbert Krickl

Universitätsrechenzentrum der Heinrich-Heine-Universität Düsseldorf

1 Introduction

The Heinrich-Heine-Universität Düsseldorf is a comparatively young state university which evolved from a former medical academy. The university was founded in 1965. It consists of the four faculties of natural sciences, philosophy, medicine, and economics. Since 1993 it has also been possible to study Law. In the winter semester of 1993 there were 17.832 students enrolled.

The computer centre is a central institution of the university. Among its responsibilities are the management of the central computing resources, operational support of the university network, user counselling, planning support for the university government, standardization, and general co-ordination in questions of university-wide data processing.

At present the computer centre subnet consists of a Siemens 7.580-S (BS2000), a Convex C220, a Convex C120, several workstations and servers such as HP817s, IBM RS/6000, Silicon Graphics, and SNI RM400, a pool of 25 Sun Sparcclassic workstations and a Sparcserver 1000 as well as a pool of about 50 PCs in two Novell networks. Among the services provided are compute and file services and general network services like printing, news, gopher, and e-mail. Additional computers – PCs, workstations, and special servers – can be found in almost every area of the university, most of them connected to the computer centre via an Internet LAN.

The usage of the computer centre resources is governed by a legal regulation called 'Benutzungsordnung' whereas institutes and working groups may register new users of their systems on a more informal basis. Somebody who wants to become a user of a computer belonging to the computer centre has to fill in an application form which has to be signed by several different persons depending on the status (student, scientific employee etc.) of the requestor and the kind of the requested resource. Before disk space can be granted, for instance, it has to be checked whether it is actually available. After passing all these steps, the user office will store the request in a database. The actual allocation of login id's and other resources of the administered systems is performed by batch procedures which are started regularly or on demand. Eventually the user is informed in written form.

User data created and registered by the computer centre may be used by other system managers of the university in order to offer net-wide services like NFS, NQS, and NCS. Vice versa it is possible that users first have a user account on a local computer and later on need an appropriate login id on a central computer. As the system administration of local computers is independent of the computer centre, it is necessary to make arrangements and to take technical precautions. The procedure to register new users, however, differs very much between the institutes. Some large institutes with several hundred users have their own formal regulations, whereas small working groups may enter their few users without a great deal of red tape.

We decided to create a net-wide user data administration system which takes into account the described situation in a decentralized environment.

This system should fulfil the following requirements:

- We regard the university network as a set of *clusters* which may or may not be identical to Internet subnets. A cluster is characterized by its own user data management. A special cluster (more equal than others :-)!) is the computer centre.
- We distinguish between natural *persons* and *projects* managed by these persons. A person can be responsible for several projects.

  In the context of this paper a project is the entity that may be allocated the rights to use computers or network services (like NFS, NQS, NCS). Therefore its characteristic data (login name, login id, group id, etc.) should be equal or at least compatible on all systems involved.
- The system should implement a client/server structure where only the data relevant for internetworking (e.g. user and project data) are kept in a central database. Data specific to a cluster or the computer centre are maintained locally.
- Central and local servers should use a uniform database format in order to guarantee consistency.
- Users and projects have to be identified unambiguously, taking into account situations like duplicate names and changes of address. Therefore unique identifiers for users and projects have to be defined.
- The participation of a cluster in the net-wide user data administration is voluntary. A user of a computer in a local cluster, however, can only benefit from certain network services if the local user data is consistent with the net-wide information structure. This is an incentive for operators of remote systems to join the global administration system.
- As all user data is security sensitive, it must be protected against system errors, corruption by mistake, and wanton attacks. It must be transmitted with suitable data encryption, authentication, and checksums.
- Apart from certain special-purpose clients, all software should be easily portable between various flavours of Unix and proprietary systems.

2 System Model Overview

The heart of the user data administration system is the central database to which only a set of special servers have direct access. The database contains personal data of users (as far as it is needed for user administration purposes), project data, numerical user id's, information about clusters and their administrators (if they take part in the administration system), and some auxiliary data like addresses of institutes etc.

Local clusters may maintain their own database which should be compatible with the central data, as well as their own servers which communicate with the central servers. As mentioned above, the computer centre forms a special cluster of its own where server and client software implement the 'Benutzungsordnung' and the derived procedures.

Access to all local and central databases is accomplished by a set of clients which take into account the individual requirements of user office staff, heads of departments, system administrators, and end users.

3 Central Database

3.1 Central User Database

Every user has to be identified and registered within the user administration system before rights (and responsibilities) are assigned to them. Their data such as personal name, name of institute or other organizational unit within the university, and postal address have to be entered into the central user database. The user must be identified uniquely so that no unauthorized person may take advantage of this user's access rights and resources; nor should the same user be registered several times with different 'identities'.

The registration itself may only be performed by specially authorized staff in the university's computer centre. It may become effective immediately since no further approval by other authorities is required.

When a user has been entered into the database, an identity card can be issued together with the user's personal password which is necessary for on-line access to the administration system. The user has to specify this password whenever he or she wants to request further resources, modify their administrative data (e.g. address or telephone number) or obtain status information about previous requests.

These transactions are not regarded as utilization of computing resources in the terms of the usage regulation and can therefore be performed as soon as the user is known to the system.

When a user is registered for the first time it is checked whether an entry for the same person already exists (even if he or she misses to state that fact). In cases where personal data like academic titles or addresses have changed it might be necessary to call back the particular institute to clarify the user's identity.

3.2 Central Cluster Database

The central cluster database is the place where information about clusters participating in the user administration system is stored. Besides the cluster name, being the key to this database, it contains the user name, postal and e-mail address of the cluster administrator, the Internet address of the machine that runs the cluster-specific database, and various limits for global resources like the maximum number of user or group id's that may be reserved in advance by the cluster administrator.

3.3 Central Project Database

The central project database is indexed by project names. Basically there are two different kinds of project names: net-wide names (*'network projects'*) and names local to a cluster (*'local projects'*).

A project name is first created locally: it is defined by a cluster administrator and entered into the central project database. The administrator may query the database and check whether the name already exists as a net-wide or a local project name.

A local name registered in the central project database is stored in a field specific to the cluster in question; in addition the responsible user is associated with the project. This is achieved by a link to an entry in the central user database where the user's identity is stored. As long as a local project name has no link to a (registered) user, the cluster administrator is held responsible for that project, and the project name remains restricted to local use.

A user who identifies himself or herself as being registered in the user database and who is responsible for a local project, may cause this project to become a 'network project' under the following conditions:

- the project name must be registered in the central project database and reserved by a specified cluster;
- the user must be the responsible person for that project name;
- the project name must not already be used as a 'network project' name by some other person;
- the project name must not be reserved by another cluster – except if the same user is registered as the responsible person there as well. (Otherwise the person or cluster administrator responsible for that project must agree and renounce their option to ever extend their local project to a 'network project'.)

With the creation of such a 'network project', the respective user becomes responsible for its net-wide attributes as well. (The user entry in the central user database contains a maximum number of net-wide project names linked to this user.)

When a local project becomes a 'network project', the cluster-specific entry in the project database is marked as 'synchronized' with the net-wide project. This implies, for instance, that the local administrator must not change the associated user and group id's as this would cause havoc when that project makes use of certain network services.

Cluster administrators may also adopt an existing net-wide project name as a local one if they accept the associated user as the responsible person for the local project.

As a result, we may summarize the following transactions performed on the central project database:

1. determine whether a 'network project' name is registered
2. list all or selected 'network project' entries
3. determine whether a 'local project' name is registered
4. list all or selected 'local project' entries
5. enter a 'local project' name, optionally specifying the responsible user and possibly requiring that this name be not yet registered as a 'network project' or a 'local project' name by any other cluster
6. modify a 'local project' entry:
   - delete it
   - enter (or modify) the id of the responsible user
   - enter (or modify) an expiration date
   - waive the right to convert this project into a 'network project' (thereby allowing a user at a different cluster to obtain a 'network project' of the same name)
7. extend a 'local project' to a 'network project'
8. adopt a 'network project' as a 'local project' for another cluster
9. modify the attributes of a 'network project'

Appropriate authentication is required for all these transactions: the requestor has to be the administrator of the particular cluster (1-6, 8, 9) or the responsible user (7), respectively.

The computer centre will only assign project names that have not yet been registered by any other cluster, and these will be entered as 'network projects' right from the beginning (which is always possible as conflicts with other clusters' local names are precluded).

3.4 Administration of Numerical Userid's

In the Unix operating system, the numerical userid is essential to identify a project also for the purpose of remote data access in a network (e.g. NFS). Therefore the system offers the feature of net-wide coordination of numerical userid's.

For that purpose there is a userid management file containing all numerical userid's that are assigned or reserved by the system. A cluster administrator can request a range of numerical userid's in order to use them for local projects and to prevent the same number from being assigned to some other project in another cluster of the network. This strategy helps to avoid problems that otherwise might occur when a user wants to extend a local project to a 'network project'.

The userid management file relates the numerical userid's to the name of the cluster for which these id's are reserved. There is a maximum number stored in the cluster database that prevents a cluster administrator from requiring too many numerical userid's by mistake.

When users extend their local project name to a net-wide one, it is checked whether the numerical userid belonging to the project name is reserved for that cluster. If this is true, the project name instead of the cluster name is stored in the userid management file as the 'owner' of this numerical userid, and the numerical userid is assigned to the net-wide project in the central project database.

This procedure implies that the cluster administrator may not change the numerical userid locally without giving up the synchronization between the local and the net-wide project name.

If a user requiring a net-wide project name already owns several identical local project names, it is sufficient that one of the responsible cluster administrators has properly performed the reservation of the numerical userid and that the other local project names are not associated with differing numerical userid's. (For projects not being bound to Unix accounts, the corresponding entry need not contain a numerical userid.)

When a net-wide project name is created without previous reservation of a numerical userid but the chosen number is marked as unused in the userid management file, it is assigned to the project name and reserved automatically.

A cluster administrator who creates a project name that is already being used by the same user in a different cluster, is strongly advised to use the same numerical userid for the local account. Otherwise the user might suffer from functional restrictions later on, for instance be unable to use NFS from or to a machine within that cluster. Therefore it is recommended that a user should *first* extend a local project name and *then* have the resultant 'network project' simply adopted by the administrators of other clusters.

Concluding this section, we summarize the transactions to be performed on the central userid management file:

1. reserve a specific numerical userid which had been assigned locally (before the cluster joined the central administration system)
2. reserve a range of numerical userid's for later use
3. release a numerical userid that had been reserved before but has not been used for other than local purposes
4. assign a previously reserved numerical userid to a (net-wide) project name (provided the number is still free or is assigned locally to this same project name by the cluster administrator who reserved this number)

Appropriate authentication is required for all these transactions: the requestor has to be the administrator of the particular cluster (1-4) or the responsible user (4), respectively.

3.5 E-Mail Related Data in the User Database

With the increasing importance of communication via electronic mail especially in the academic and scientific area, the e-mail address becomes as important as the phone number. In principle, all members of the university as well as all projects should be reachable via e-mail. A central mail server is planned as a default 'mail exchanger' for this purpose.

The e-mail address should be independent of the hostname of the computer where the user's mailbox resides. It should only consist of the user's first name, surname, and the university's address (`Firstname.Surname@uni-duesseldorf.de`); modifications are needed to distinguish between different persons who happen to bear the same name.

A similar construction of an e-mail address is required for a project (`project@uni-duesseldorf.de`) which may or may not be an alias for the responsible user in charge of this project.

The mail server performs the mapping of such a 'global' mail address to the mailbox address that is recorded in the user database or the project database, respectively, and which may be selected by the user at the time of registration and can be changed at any time later on. The mapping might, for instance, be changed temporarily in order to forward the user's mail to another mailbox in cases like a computer breakdown or during the user's holiday.

We have different modes of mail delivery in mind that are both required by our users and supported by the mail server's software (which is PP 6.0); these modes must also be supported by the user administration:

- The mail is forwarded to the user's favourite host where he or she has an account and logs in frequently.
- A group of hosts – probably for security reasons restricted to hosts that are administered by authorized staff of the computer centre – may directly share the mail server's mail spool directory via NFS. In this case, the mail address is mapped to a project name also valid on at least one of the participating hosts. The user can access the mail from any of these hosts where he or she possesses an account with the respective project name.
- The mail server also supports mail delivery via the Post Office Protocol (POP). This protocol authenticates the user through a password which is kept in the central databases.
- E-mail addresses of projects may also be mapped to a list of personal mail addresses.
- For users registered in the central user database who do not have a login id or access to a POP client, delivery of a reprint of the e-mail via inhouse mail or as a fax message is planned.

The following transactions have to be performed on the central e-mail address database:

1. determine the owner (user or project identification) of an e-mail address
2. enter an e-mail address together with the requested delivery mode, the owner identification and the actual mailbox address
3. change the delivery mode for an existing e-mail address
4. change the owner of a registered e-mail address
5. change the mailbox address for a registered (global) e-mail address

4 The Computing Centre Cluster

A person who wants to become a user of the computer centre has to take several steps. First of all his or her personal data have to be registered in the central user database. After that he or she can state a request for a resource like system dialogue or disk space on a particular machine. The request can be made in written form on a questionnaire or online via a special client. If the user prefers the written form, a member of the user office staff will enter the data into the system. All necessary transactions are supported by a dedicated 'user office client'.

Several authorities must approve the request before any rights can be granted. In most cases a faculty member as well as a senior member of the computer centre have to verify that the request is in accordance with the regulations (the 'Benutzungsordnung'), and the administrator of the system concerned might want to check whether the resources are available.

The request data is kept in such a form that these authorities can be notified electronically; they can approve or reject the request and obtain status information about pending requests and resource utilization on-line by means of special 'approval clients'. The user office staff, however, can enter all approvals on behalf of these authorities as well because some of the persons involved might not yet be all too familiar with computerized systems and instead prefer to give their consent in the traditional form by signing a written document.

Once a request has been accepted and becomes valid, it is automatically entered into a quota database for the particular machine. Such a database exists for each machine that is controlled by the user administration system. Individual clients are implemented on all these machines which routinely or on demand compare the quota database with the actual resource allocation and invoke operating system commands in order to adjust the allocation values.

If a new project has been allocated to a Convex, for instance, the client will receive all necessary information about this project such as login name, numerical userid, disk quotas etc. and consequently run 'nu' and *...* commands. As soon as this is successfully done it informs the server which in turn marks the request as fulfilled and causes the printout of an approval letter which will be sent to the user, informing them that the account has been established and is ready for use.

If an error occurs, the client will also notify the server and thus pass the return information and error codes to the user and to the system administrator who will investigate the problem.

The request database also contains expiration dates for all valid requests so that the user can receive timely warnings before their accounts expire or resources are withdrawn. If the user does not enter a new request, the account will be deleted or the resource allocation cancelled without manual intervention.

To avoid major changes to the software of the user administration system in case the set of administered computer centre resources is extended or modified, the software has been designed in a very flexible way. Each administrator of a system or a particular resource maintains a 'resource description database' which contains a formalized description of each resource item, consisting of

- the name of the individual item,
- the data format of the possible resource values,
- restrictions and dependencies.

These data are also used by the user office and end user clients to generate the screen layouts and user interfaces dynamically so that requests will always conform with the current set of resource definitions. Thus the resource description database acts as a meta-language to describe the structure of the request and the quota databases.

We support system administrators in programming their resource allocation clients by providing library functions for the dialogue with the corresponding servers and exemplary clients which we created for some selected systems.

5 Clients

Several different types of clients are needed for human as well as for automated access to the user administration databases. Some of these can also be used by interested system administrators (not belonging to the computer centre) to obtain a uniform resource administration within their local clusters.

User Office Client: This client is the most comprehensive client because it is able to perform some functions of other, more specialized clients. It also administers auxiliary databases, e.g. address lists. It communicates with the central (user, project, userid) databases and all (request and resource) databases of the computer centre cluster. We chose an implementation based on Turbo Vision because of the availability of MS-DOS PCs which are already in use for various user office tasks.

The user office client offers functions to

- register and maintain user-specific data
- allocate new projects
- modify project-specific data
- enter requests for resources
- approve requests on behalf of the authorities
- print user identity cards and request approvals

Global Management Client: It operates on central databases and allows, for instance, to modify the cluster database.

Cluster Administrator Client: This is the tool for local cluster administrators to communicate with the central databases.

User Client: Every registered user should have access to this client. Therefore it has to be implemented on MS-DOS PCs and on the major part of the workstations that are connected to the system. This client enables end users to

- enter requests for computing resources
- obtain information about their personal data and their resource requests
- modify their personal data (e.g. home address)
- modify global administrative information (e.g. mailbox address)
- change the status of projects for which they are responsible (extend a local project to a 'network project')

The user client contacts both the central database and the local cluster database.

Approval Client: This client is used by the authorities to check, accept or reject resource requests for which they are responsible. Depending on the function of the particular person, the client may give access to status information (e.g. resource usage) which is relevant to decide on the fate of the request. It contacts the computer centre database as well as local resource databases (if necessary).

Resource Allocation Client: This client contacts the associated resource quota database and must be programmed to control the local resources on a particular system.

6 Security and Privacy

The client-server protocols are designed very carefully in order to guarantee security and privacy of the data. German Data Protection Law requires that personal data should be transmitted on the Local Area Network in encrypted form, and data concerning the allocation and administration of resources must be carefully protected as it might be a prominent target for crackers.

On the highest protocol layer we have a logical protocol for each special type of client. The underlying layer provides services like connect, disconnect, send, and receive.

The connect service includes authentication of the requestor using the RSA public key algorithm and Diffie-Hellman exponential key exchange. The identity of a person starting a user administration client is verified by means of a passphrase and, where needed, by a secret keyring stored on a floppy disk.

All data transports are secured by MD5 cryptographic checksums and an IDEA conventional encryption where needed, using random keys which are negotiated during the connection phase. The TCP/IP protocols are used for the networking interface.

AVS: NEW MODULES AND OPTIONS

Markus Pleier
Convex Computer GmbH

I. Introduction

The application visualisation system (AVS) has grown to a more and more standard for technical and industrial graphics on an applications level. Due the movement from hardware manufactor to software vendor AVS Inc. (former Stardent) has opened this kind of software for more platforms than any other equivalent tool. AVS has gone from a single workstation support tool for graphics to a more and more complete picture of image processing facilities and 3D graphics options. In combination with a supercomputer like the Convex C-Series AVS gets the possibility to give the user very fast results. So, if there are some mistakes in the algorithm, the user doesn't consume and waste expensive CPU cycles. This leads to a 56% install base on all Convex 3800 systems.

The mechanism AVS is developed, tested and proved works on a gear wheel relationship between Convex Computer Corporation and AVS Inc. Due the high performance enhancements and the much complex operating system in a supercomputer than a workstation, Convex needs a longer and more complex test period for a new release then workstation vendors. This is the reason for the "late" appearance of the ConvexAVS 5.0 version.

However, the current 3.9 version has 95% of the functionality of the 4.0 release from AVS Inc. So ConvexAVS V5.0 will complete the transition to the AVS 5.0 standard since it will supplant AVS 4.0. It will also provide additional modules that depend upon the advanced capabilities present in the newest AVS kernel. The key feature of the development is the implementation of Convex-specific modules which enhance distributed visualisation between Convex C-Series supercomputers an Hewlett-Packard workstations.

This new release of ConvexAVS was based on the following demands:

- complete transition to the AVS V5.0 standard
- incorporate Convex - specific enhancements into the new software base
- release a complete documentation set that incorporates V3.0 and V3.9 documents with new features found in ConvexAVS 5.0
- resolve all outstanding customer related bugs

II. New functionalities

a. Kernel

The AVS 4.0 kernel from AVS, Inc. offered major new functionalities such as hierarchical networks or "macro modules", enhanced flow executive and a software renderer.

- Hierarchical networks
  Hierarchical networks refer to a method of encapsulating whole networks with simplified inputs and outputs in such a way that the network appears to be a module. This allows networks to be used in a true "building block" fashion and enhances the ability of a user to build complex data visualisation networks.
- enhanced flow executive
  The enhanced flow executive provides support for three new features: direct-module-to-module communication, parallel module execution, and XDR bypass. Direct module-to-module communication is a feature that has a major impact on remote module execution with distributed visualisation. ConvexAVS supported remote module execution starting with V3.0, but with the limitation that remote modules in separate executables could only communicate by sending data through the local kernel. The new feature allows remote modules to communicate directly between each other, eliminating the need to send the data back and forth over the network connection. Parallel module execution optimizes data flow in a module network and allows independent modules in separate executables to run simultaneously. XDR bypass allows modules running on the same machine to communicate in the native machine data formats rather than going through an intermediate XDR translation step.

- New software renderer
  The new software renderer in AVS 5.0 provides support for two capabilities previously available only on selected hardware platforms, texture mapping and integrated volume rendering. AVS 5.0 also makes it possible to select the renderer type on a per-camera basis, thus allowing multiple renders on the same display screen.

b. Module

Over 66 new modules, including a set of 45 image processing modules, four new volume rendering modules, and 10 new UCD modules are attended to the new release. Following is a list of the new modules in ConvexAVS 5.0.

Colorize molecule

The Colorize molecule module based on atomic radius, formal charge or atomic weight.

Copy molecule

The Copy molecule module copies the input molecule to two output molecules; one is the actual input molecule and the other is its copy.

Data Viewer

The Data Viewer module invokes the ConvexAVS Data Viewer Application.

Display molecule

The Display molecule module allows the user to display the structure(s) contained in the input molecule in a variety of ways: colored sticks, ball and sticks, and solid spheres.

Mol list editor

The Mol list editor module takes the input molecule and performs the list editing function specified by the current parameters.

Molecular area

The Molecular area module computes the molecular surface area, along with the percent exposed surface area for each atom in a molecule.

Molecular volume

The Molecular volume module calculates the volume of a given molecule.

Monopole elesta

The Monopole elesta module calculates the molecular electrostatic potential surrounding a molecule, using the monopole approximation.

Print molecule

The Print molecule module generates a hierarchical display of the entire contents of an input molecule.

Read structure file

The Read structure file module reads a molecular structure file. This file (suffix *.con*) contains the title and number of atoms in the molecule along with the location, type and connectivity of each atom in the molecule.

Select molecule

The Select molecule module allows a user to select individual structures from a molecule.

Symmetric float

The Symmetric float module sends a positive and negative floating point number (same absolute value) to one or more floating point parameter ports on one or more AVS modules.

Translate molecule

The Translate molecule module permits the user to center the structures contained in the input molecule at their center of mass, charge or coordinates or translate all of the structure´s x, y, or z axes.

Write structure file

The Write structure file module writes a molecular structure file. This file (suffix *.con*) contains the title and number of atoms in the molecule along with the location, type and connectivity of each atom in the molecule.

3D bar chart

The 3D bar chart module converts a 2D floating point filed into a group of 3D blocks or planes for viewing in the Geometry Viewer.

average down

The average down module reduces the size of a 2D scalar uniform byte field in any combination of dimensions by computing the mean of each block of values.

blend colormaps

The blend colormaps module takes two ConvexAVS colormaps as input and interpolating between them to generate an output colormap. This functionality allows the AVS Animator to animate colormaps.

brick

The brick module uses 3D texture mapping techniques in the ConvexAVS software renderer to perform slicing of 3D volume data.

byte image to float

The byte image to float module converts an image contained in a byte field to one contained in a floating point field using a method which avoids a hue shift.

calc warp coeffs

The calc warp coeffs module calculates the warp coefficients required by the warp module from a given tiepoint list using matrix inversion.

clip geom

The dip geom module interacts with the ConvexAVS Geometry Viewer to allow the specification of arbitrary clipping planes which may be applied to geometric objects.

colorize geom

The colorize geom module takes an input field and a colormap and assigns colors, transparency values and texture mapping coordinates to the vertices of geometric objects.

combine geom

The combine geom module is a Convex AVS built-in module that accepts multiple geometry edit list input connections and constructs a single geometry edit list output that can be passed to an external module, such as render viewer.

compute shade

The compute shade module combines the functionality of the colorizer, compute gradient and gradient shade modules as a single memory-efficient module.

create geom

The create geom allows the user to interactively create geometric objects such as Points, Polylines, Arcs, Circles, Surfaces, Revolutions, and Extrusions. It also provides a set of operations to modify created objects.

cube

The cube module ray-traces uniform volumes in four modes: texture mapping, maximum value, transparent surfaces based on a substance table, and a surface creation mode that stores intermediate results for rapid computation for surface opacities and colors.

dialog box

The dialog box module is a coroutine that uses a dialog box to input long string parameters that can be output to other modules.

draw grid

The draw grid module draws a set of grid lines superimposed on top of an image.

edit substances

The edit substances module creates the substance table required by the cube modules *ray-cast* and *create-surface* rendering modules. It communicates with cube via a user defined data structure called *substances*..

excavate

The excavate module removes a selected octant from a 3D scalar byte field, revealing the interior data.

excavate brick

The excavate brick module provides yet another way of visualizing 3D uniform volume data. The volume is displayed using multiple orthogonal slice texture mapped slice planes.

field to short

The field to short module takes any field as input and converts it into an output field or date type short.

float image to byte

The float image to byte module converts an input image contained in a floating point field to one contained in a byte field.

generate grid

The generate grid module creates a geometry representation of the coordinate planes XY, XZ and YZ in the form of a grid. The user can control the size of the grids, number of grid lines, and initial position for each plane.

geometry viewer

The geometry viewer module provides access within an AVS network to the complete Geometry Viewer subsystem.

image measure

The image measure module measures the distance between two pixels of an image. The result is reported in pixel units.

image probe

The image probe module reports the data values present at a pixel location selected in the image viewer module's window.

image to cgm

The image to cgm module converts its input to the Computer Graphics Metafile (CGM) format and stores it in a file.

ip absolute

The ip absolute module calculates the absolute value of all the data elements in the input field and places the result in the output field.

ip arithmetic

The ip arithmetic performs arithmetic between two uniform fields or between one uniform field and a constant value.

ip blend

The ip blend module performs a pixel-by-pixel composition of two fields, using an alpha mask field for the blending.

ip compare

The ip compare module compares two fields (usually images), looking at such information as sizes (extents), vector lengths, data types and pixel values.

ip contour

The ip contour module derives iso-level contours from the source field and draws the contours into the destination field.

ip convolve

The ip convolve module convolves an input field with a specified floating point kernel.

ip dilate

The ip dilate module performs dilation or "region growing" morphological operations on fields based on an arbitrary structuring element.

ip edge

The ip edge module performs an edge enhancement operation, using one of several specified algorithms.

ip erode

The ip erode module performs erosion or "region shrinking" morphological operations on fields based on an arbitrary structuring element.

ip extrema

The ip extrema module finds the minimum and maximum data values in one channel of a field.

ip fft

The ip fft module computes the Fourier Transform of a uniform field (non-complex) and places the packed complex result in a uniform float output field.

ip fft display

The ip fft display module converts the packed conjugate symmetric FFT representation written by ip fft to a displayable form by calculating the magnitude and/or phase of the packed input field.

ip fft pack

The ip fft pack module folds real and imaginary input fields into a single output field appropriate for inverse transform using ip ifft.

ip fft unpack

The ip fft unpack module unfolds the conjugate symmetric FFT representation written by ip fft into two fields that represent the real and imaginary components. The destination fields will be conjugate symmetric.

ip float math

The ip float math module perfoms floating-point operations on the input

ip histogram

The ip histogram module takes the non-normalised histogram of the source field.

ip ifft

The ip ifft module performs an inverse Fourier Transform of a conjugate symmetric field to produce a real (non-complex) field.

ip lincomb

The ip lincomp module operates on the vector elements ("bands") of a source field, combining the vector elements of each input pixel to produce pixels in the output field.

ip linremap

The ip linremap module linearly remaps a field (generally an image) by first adding a constant to the input pixels, and then multiplying by a multiplier value.

ip logical

The ip logical module does bitwise logical operations on each pixel of the two input fields and places the result in the output field.

ip lookup

The ip lookup module passes a byte or short field through an integer lookup table. The original number is replaced by the number found at that index in the lookup table.

ip median

The ip median finds the median value in a local collection of pixels using a structuring element.

ip merge

The ip merge module merges two fields (usually images) on a pixel-by-pixel basis, using a region of interest (ROI) filed to specify which source field a given pixel in the output field comes from.

ip morph

The ip morph module performs various morphological operations on an input "logical" field and places the result in the output field.

ip read kernel

The ip read kernel module reads a convolution kernel from a file into a 2D uniform float scalar field. This kernel is usually used as an input to the ip convolve module.

ip read line

The ip read line module reads a line of pixel data between two pixels of an image, outputting the data as a 1D n-vector field.

ip read mtable

The ip read mtable module reads a morphology table from a file into a 1D uniform byte scalar field. These tables are used as inputs to the ip morph module.

ip read sel

The ip read sel module reads a structuring element from a file into a 2D uniform scalar field. This structuring element is used as an input to the ip dilate, ip erode and ip median modules.

ip read vff

The ip read vff module converts SunVision *.vff*-format image files into a 2D uniform AVS field with dimensions equal to size = *xsize ysize,* vector length equal to bands = *n*, and a data type that corresponds most closely to bits = *n*.

ip reflect

The ip reflect module reflects a field (usually an image) in one of seven different directions.

ip register

The ip register module performs a sequential search correlation match of a field with a template.

iop rescale

The ip rescale module rescales fields (usually images) by linearly remapping the pixel values between src min and src max in the input field to the output field in the range dst min to dst max.

ip rotate

The ip rotate module rotates a field about its center.

ip statistics

The ip statistics module finds the number of pixels, mean and variance of one channel of a field. These outputs are displayed as floating point values in an output widget on the modules control panel.

ip threshold

The ip threshold module computes the threshold of a field against a floating point value, producing a bi-valued (logical) byte field as a result.

ip translate

The ip translate module copies the input field to the output field with the translation relative to the input specified by dx and dy.

ip twarp

The ip twarp module performs an arbitrary warp using a warp table to designate which pixel in the input field corresponds to each pixel in the output field.

ip warp

The ip warp module applies a geometric transformation to a field (usually an image).The transform is defined as a polynomial mapping from an output pixel position to an input pixel position.

ip write vff

The ip write vff module converts an AVS field in image format into a SunVision binary *vff*-format image and writes it to disk.

ip zoom

The ip zoom module zooms a field using one of four interpolation methods. The zooming can be done with floating point offsets, which enables you to offset a zoomed image by fractional pixels.

label

The label module creates a label text string in GEOM_LABEL format for input to the geometry viewer module.

minimax

The minimax module modifies the minimum and maximum values of a selected vector element (channel) of an n-vector AVS field. The output field is identical to the input field, except for the new vector minimum and maximum values.

render viewer

The render viewer module provides the capabilities of the geometry viewer module in a form separate from the AVS kernel.

ribbons

The ribbons module generates a set of geometric ribbons by taking the polyline output of the stream lines module and replacing them with finite width, colored, and textured polytriangle ribbons.

scatter to ucd

The scatter to ucd module converts a scatter field to a single UCD structure of tetrahedral cells using a Delauney tessellation algorithm. The scatter data points become the nodes of the tetrahedral ICD cells. Each vector element becomes a node data component in the output structure.

sketch roi

The sketch roi module creates the region of interest (ROI) field that modules such as ip edge, ip twarp, and ip convolve use to restrict their operations to a subset of their input image.

string int

The string int module generates an output string based on string prefix, embedded integer field, and string suffix parameters.

time sampler

The time sampler module extracts two sequential 3D fields from a 4D field and interpolates between their computational data values by one of three techniques, producing a single 3D field as output.

track ball

The track ball module generates a 4x4 transformation matrix, producing the same transformation output as euler transformations. The difference is that you specify the transformation with a direct manipulation of a trackball widget, rather than with dial values.

trigger int

The trigger int module accepts a field as input and outputs a new integer value as each new field arrives. Parameters exist to control the initial integer value and subsequent increments.

ucd cell color

The ucd cell color module is used to color a UCD structure based upon either the *cell* data values, or the data values of the structure's material ids.

ucd curl

The ucd curl module accepts a UCD structure with vector node data components as input and computes the curl of that structure as output.

ucd div

The ucd div module accepts a UCD structure containing 3-vector mode data components as input and computes the divergence of the vector component as output.

ucd grad

The ucd grad module accepts a UCD structure containing scalar node data components and computes the gradient (3-vector float data) at each node of the output.

ucd math

The ucd math module performs unary and binary operations upon ICD structures, working with both node and cell data.

ucd minmax

The ucd minmax module modifies the minimum and maximum values of a selected scalar node data component in a UCD structure. The output UCD structure is identical to the input structure, except for the new component minimum and maximum values.

ucd plot

The ucd plot module samples the node data along a line through a ICD structure, producing a 2D field that is used as input to the graph viewer modules's rightmost (XY plot) port.

ucd reverse cell

The ucd reverse cell module corrects ordering mistakes in the cell topology of a UCD structure. It will either traverse the entire structure, reordering all incorrect cells to match standard ordering, or it will reverse the normals on all cells of a particular type.

ucd rubber sheet

The ucd rubber sheet module maps node data component values as a 3D surface.

ucd vol integral

The ucd vol integral module performs two functions. One is calculating the total volume enclosed by a UCD structure's cells. The other is computing the volume integral of any one of the scalar node data components.

c. Miscellaneous

The ConvexAVS 5.0 has also a variety of new features which are necessary to provide more compex data. Also is the handling with some modules a improved. The list of additional features contains:

- Interface and API support for interactive mouse-driven image drawing and picking operations in the Image Viewer.
- 8-bit scalar images in the Image Viewer
- Full support for a short (16-bit) field data type.
- Geometry Viewer support for sending scene windows to a different display, turning off image output to improve performance, non-transformable objects and new texture-mapping *libgeom.a* calls.
- Support for ANSI-C and C++ modules.
- Unmapped module control panels.
- The ability to write "in place" into shared memory input data
- A new, reorganised list of default module libraries.
- AVS can be installed anywhere in a directory hierarchy, without the need for links from */usr*.

III. Conclusion

With the implementation of the 5.0 release of ConvexAVS on the C-Series, the complete AVS package grows to a more stabilized software. Lots of bugs have been reported to AVS Inc. during the

port of the software package to Convex Supercomputers. This leads to a more stabilized environment. Now the product has increased functionality, which will be the best for Meta-Series or complex environments. However, AVS is a software which will be developed further. Every application, every module and every user who will use this package will increase the availability and functionality which is the main advantage of AVS.

Bibliography

1. Smith Wade, *ConvexAVS V5.0 Project Plan*, Convex Computer Corporation 1993
2. Convex, *ConvexAVS X5.0.0.1 Release Notice*, Convex Computer Corporation 1993
3. Smith Wade, *Distributed Computing in AVS*, AVS User Group Conference 1993

CRITICAL ASPECTS FOR SECURITY IN LAN'S.

Prof. Dra. Mª Teresa Areitio and Prof. Dr. Javier Areitio
University of Deusto
Faculty of Computer Science
SPAIN

ABSTRACT

Today's, LANs, micro-to-host links, gateways, routers, bridges, and other data communications equipment and facilities provide microcomputer users with access to an abundance of corporate data, and this access will intensify as a result of the growing trend toward distributed computing. This paper discusses the problem of maintaining network security and elaborates on possible solutions.

INITIAL ASPECTS

Two major trends are shaping the future of the network security field. The first trend, smart cards also called integrated circuit or chip cards, will become more prevalent as personal identification data carriers. These cards, equipped with a microprocessor and solid-state memory, perform on-line encryption, maintain a time-on and time-off log in memory, and handle biometric identification as well as passwords. The smart card can also carry a LCD display and miniature keyboard, allowing users to interact with the network. Smart cards provide a high level of security because each log-on exchange is unique to the time, the workstation, and the user. The second trend is the development and adoption of industrywide security standards by important Organizations.

This paper identify sources of LAN security violations and remedies. In this paper we analyze solutions mainly in the area of access control and protection in LAN environments.

LANs themselves are becoming an increasingly popular method for micro-to-mainframe connectivity. Whereas LANs provide true peer-to-peer networking, some businesses choose to or are contrained to use them in a traditional host-to-terminal architecture. The use of gateways represents a relatively new innovation in micro-to-mainframe communications. In this scenario, a microcomputer designed as the gateway is equipped with appropriate hardware and software to enable it to communicate with the mainframe. When another microcomputer on the LAN requires access to the mainframe, the request is addressed to the gateway, through which access is provided. The gateway perform the neccessary conversions that let the microcomputer communicate as a terminal with the mainframe. In distributed processing environments, microcomputers, minicomputers, and mainframes each handle a porcion of an application. The entire application and data appear to reside locally to the user;

however, each application component and individual user has access to (and, sometimes, control over) the collection of corporate data residing on various hosts and storage devices. Such openness invites misuse and requires some degree of administration to enforce new security procedures.

SPECIFIC PROBLEMS IN LANS

In communication security, local area networks were initially conceived as systems that would facilitate communication between a wide variety of users and workstations in a limited geographic area with, possibly, gateway connections to wide area networks. They have, therefore, the potential vulnerabilities of computer systems and wide area networks.

It may be argued that the nature of LANs is such that they present no particular security problems because:

1) The equipment is installed in secure premises.
2) The transmission paths are short and traffic is localized to a specific area.
3) Access protection is guaranteed by passwords.
4) Network failures can be tolerated by reversion to backup procedures.
5) Fiber optics cabling can be used to prevent line taps.

In fact the security of LAN can present quite specific problems, namely :

a) The network was designed to facilite the connection of additional workstations and to interface with a variety of manufacturers' equipment.

b) The network may operate in a broadcast mode so that every message is transmitted to every workstation; a listener at any location is thus guaranteed access to every message.

c) A variety of signal types may be transmitted through the network, thus unauthorized users and messages may be difficult to detect.

d) Radiofrequency emissions both by the network and at point of interconnection, unless special precautions are taken, or fiber optic cabling is employed, may occur.

e) A high proportion of computer frauds are perpetrated by employees; the flexibility and ease of access to network facilities increases the range of options available to a technically competent, fraudulently inclined employee.

Deliberate misuse of LANs can arise both from illicit taps and subversion of users or workstations. Illicit taps can be facilitated by the provision of spare interface points incorporated to allow for a future expansion of the network. Use of such illicit taps is also simplified by the well-publicized, user friendly protocols of LANs. An illicit workstation connected to a LAN may be used to listen in to sensitive information, or

to inject illegal messages. Such messages could be used to disrupt the network or to trigger the release of security-relevant information (e.g. the workstation may masquerade as a host to persuade another workstation user to reveal passwords).

As illicit workstation may be prevented or discovered by sensible physical access controls, but the attacker may well be a legitimate user of a workstation. In this case the subversion may involve the attacker using a terminal in one area to gain access to sensitive data from another office. The attacker may also subvert the workstation itself by modifying the hardware or software. This opens up the possibility that the workstation would continually monitor and store network data, for later use by the attacker, who would be relatively safe from detection.

LANs are vulnerable not only to deliberate attack, but also to network malfunctions which can have serious effects upon the organization's operating efficiency. The problems can arise from the factors listed below :

1) Network failures or inadequate performance arising from cable installation performance by inexperienced staff or with inadequate attention given to network specifications.

2) Inadequate protection against interference from power surges and current spikes.

3) In general there is limited experience in LAN operation and inadequate diagnostic tools are available.

4) Ring networks are particulary susceptible to workstation failure; unserviceable nodes can be bypassed, but such failures may cause temporary corruption of network data and degraded network performance.

The important security features of a LAN are listed below:

a) Confidentiality of messages, if necessary employing encryption.

b) Integrity of transmission (i.e. all completely and correctly valid messages are received and invalid messages from unauthorized users or modifications of genuine messages and replays, are inhibited).

c) Access control for data and resources (i.e. no unauthorized use of workstations). This is particulary important if the LAN has a gateway to a wide area network.

d) Service availability when the attack is directed towards incapacitating the network or reducing its performance by message flooding.

e) Protection against covert use of the network (e.g. by coded signals).

The protection of LANs can be tackled, at one level, as a management problem. The level of security applied must be appropriate to the estimated losses from an attack or

system failure. LANs were primarily designed to facilitate information interchange among professional and office workers. The impact of security precations upon users should be such that they recognize the purposes and importance of such measures. Overrestrictive user procedures may simply discourage use of the network or lead to circumvention of security measures; there are common reports of elaborate passwords written on the side of terminals to serve as a help to the users. Sensible access controls to workstation, protection of cables against illegal taps, monitoring of network traffic, etc. can, in many cases, provide an adequate level of security. The network topology itself can be designed to enhance security. For example, if the network is effectively divided into subnets then the traffic to and from sensitive areas can be supervised by controlled gateways.

In those applications where security is a major factor, the design of the network itself, or at least the interface to it, must be considered from a security point of view. A LAN may be designed as a distributed secure system if it is important to minimize the information flow from one security compartment to another. Other proposed security procedures for LANs are based upon the concept of unprotected and protected LANs.

The security of the unprotected LAN is thus based upon the assumption that any user, legal or illicit, can both receive and inject network messages. Sensitive data is protected by encryption, and there must be tamper-resistant modules at the host and workstation to protect cryptographic keys. The security of unprotected LAN is based upon the assumption that the network is untrusted and unprotected, and the workstation itself may be subject to subversion; therefore security is a host function responsability. A trusted unit, termed a special terminal interface unit, is interposed between the network interface and the workstation. The host will have a corresponding special host interface unit connected between the computer and the LAN interface. Sensitive messages will be encrypted on an end-to-end basis but the network control data must be sent in clear.

The host is assumed to be a trusted unit; the special terminal interface unit and the special host interface unit establish each others identities using installed cryptographic keys. The host then established the user's identity by requesting the user to input account number, and password, directly into the special terminal interface unit to obviate a subversive attack on the workstation. The account information will be sent in clear, but the validation of passwords may involve messages encrypted by session keys, which are therefore only available to the host and user.

The protected LAN, on the other hand, is one in which the LAN provides a service to its hosts that guarantees authentication, mutual secrecy and message delivery. The protected LAN distrusts all hosts, workstations and users. Trustworthy network interface units, which are tamper-resistant and unforgeable, are built into the network. Encryption may be employed for network communication, but it is not essential if the physical installation is such that it renders the network impervious to illicit taps. Control of communication over the LAN is the responsibility of a secure authentication server. In the quiescent state the trustworthy network interface unit can only communicate with the secure authentication server. The user enters an account number, password and

requests for resources, communication to other workstation, etc. The trustworthy network interface unit transmits this information to the secure authentication server, which checks the user's identity, password, authorized privileges and requests. If these are valid the secure authentication server sets up the requested connections and places the trustworthy network interface unit in a state that allows communications over these connections. When the user logs off the trustworthy network interface unit reverts to the quiescent state.

The trustworthy network interface unit is connected between the workstation, or server, denoted as end system and insecure communication network. The function of the trustworthy network interface unit is to ensure that data can only be communicated between end systems sharing a common security compartment. Data is only passed from the network to the end system if it has the appropriate security compartment label. Normally the trustworthy network interface unit encrypts and decrypts data flowing from, and to, the insecure network, and the compartment label provides an identifier to the appropriate cryptographic key. If the end system is allotted only a single security compartment, then the workstation software has no security function, and a conventional operating system may be employed.

On the other hand, a workstation dealing with a number of security compartments (e.g. a file server) will require secure software to label outgoing data correctly, check labels on incoming data and apply security policies to flows of data between security compartments in the end system. An access control center sets up the trustworthy network interface units for specific workstation sessions. A workstation user logs on, passing name, password and desired security compartment, via the trustworthy network interface unit, to the access control center. The access control center checks the password of the user and ensures that the requested priviliges have been authorized for that user. If the access control center checks are successful, then it loads the trustworthy network interface unit with the appropriate compartment label, and any associated cryptographic key.

At the end of the session the trustworthy network interface unit purges all secure information from the workstation, informs the access control center of the log out and reverts to its initial state. If dumb terminals are employed, instead of workstations, then the trustworthy network interface unit is replaced with a trustworthy terminal interface unit. In this case only one secure compartment is made available to the terminal; the access control center can provide a specific label to the trustworthy terminal interface unit, and the trustworthy network interface unit of the host workstation communicates with the dumb terminal. Thus there can be separation between dumb terminals even if they are communicating data from the same security compartment.

SECURITY RECOMMENDATIONS

With security issues becoming increasingly important among businesses of all types and sizes, its a good idea to have a security plan in place. The plan's extensiveness

is directly proportional to the network's complexity. At a minimum, the plan should address the following :

1) Physical installation and access guidelines, including approved add-in security protection.
2) Access levels and definitions.
3) Rules for access, such as password policy and volume access.
4) Workgroups and how they fit into the network's access rules.
5) Internetwork connections and access requirements.
6) Levels of administration and accountability.
7) Accountability guidelines.
8) Periodic and random auditing policy; software upgrade, auditing, and installation policies.

The security plan should have the power of a corporate policy, signed and authorized by the management's highest level. Department heads should administer the plan, and every user should be requiered to read and sign it. Security plans should be reviewed and revised at least annually; perhaps more frequently if employee turnover is a problem. The ideal network security plan must protect against theft, alteration, and interception of network information. It should also deter theft or vandalism of equipment, the duplication or resale of programs, and unauthorized use of computer time by wiretapping or dialing in. Companies should first implement procedures to deter unauthorized users who may be tempted to breach the system. When properly implemented, such deterrents bolster confidence in the integrity of the organization, ensuring management, staff, and customers that the system is safe and dependable.

When deterrence fails and an intruder attempts to breach the system, preventive techniques, including access controls, data encryption, network-intrusion detection, and interface barriers guard against further damage. Companies should choose a security method that offers password control, data encryption, alarms to notify administrators of a security breach, and audits trails to locate the source of the breach. Companies should design a security plan that can expand to accommodate network growth. Because no security system can function effectively without the cooperation and support of end users, network support personnel, and upper level management, companies should evaluate the effects additional security measures will have on authorized users.

Network users usually resist improvements in network security because security devices and controls can make a complex system even more difficult to operate and maintain. Trusted employees often resent having to use passwords to access the network. For these reasons, some security measures may cause delays and reduce productivity. Companies should expect resistance to added rules and regulations and counter this by providing security awareness training. The data-processing manager, with the aid of the human resources department, must sell the long-term employee on the idea of additional security by implementing bulletin-board campaigns and training sessions. At a minimum, security awareness training should include :

a) An explanation of password management, including password selection, access privileges, routine password changes, and the need for secrecy.

b) A description of tangible security measures, such as locks, keys, card systems, and badges.

c) A policy and procedure for implementing employee suggestions for improving security.

d) Procedures for reporting lost or stolen equipment.

e) Definitions of sensitive data and procedures for keeping data confidential.

Companies should also develop security demostrations, possibly in the form of a game in which one employee is assigned to penetrate the system while others try to prevent or detect the attack. Games demostrate how well the system works and involve staff members with the security system. Whatever the result of the game, the company wins. If the attempt fails, the security system is performing effectively. If a security breach occurs, the company has identified a weakness and can take appropriate corrective measures. Once an acceptable level of security awareness exists throughout the organization, the company should schedule periodic retraining sessions to maintain awareness. All aspects of a security system (i.e. hardware, software, premises, facilities, and personnel) must work in unison and at a consistently high level to safeguard against threats to security.

Using multilevel password protection to govern disk drives and ports prevents individuals without full knowledge of the password library from accesing privileged information. Password-controlled access can extend to selected ports, drives and specific operating system commands for PCs, compatibles, etc. Callback modems can be equipped to provide complete security or partial protection. Some modems, for example, combine callback security, password protection, and data encryption, thereby providing one of the highest levels of dial-up access security currently available. The next step is to implement security procedures before the modem handshaking sequence, rather than after it. This effectively elimites unauthorized access opportunities. In operation, the user gives a password to the modem before making the call. When the remote modem answers, it solicits the password from the calling modem before the units complete their handshake. Voice-recognition technology validates spoken passwords, ensuring that only authorized personnel can access corporate systems and networks. These units compare the caller's voice to voice prints kept on file. Speaker-password validation takes only a few seconds. All remote access attempts are logged, including an actual recording of the call to help investigate suspected toll fraud.

LAN SECURITY COMPONENTS

Enterprisewide network security is a combination of LAN security measures and host-based data encryption schemes. In turn, LAN security is a combination of software and physical restrictions.

(A) LAN security schemes typically permit a network administrator to allow or deny individuals or groups of users the ability to read, write, create, delete, search, or modify files. LANs use one or more of three types of software based security:

1) The first type gives each shared resource on the network a network name; a single name can designate an entire shared drive, a subdirectory, or even a file. A password can be associated with a network name and limit the read, write, and create capabilities affiliated with that password. The network name architecture makes it easy to protect the shared resources, but one user might have to keep track of several passwords. Security is easily compromised when password management becomes overly burdensome. to make matters worse, passwords are transmitted over the LAN in the clear and therefore vulnerable unless encryption is used.

2) The second security type assigns specific rights to a user and provides a password for the identification of the user. This works fine in small organizations, but customizing the rights of each user quickly becomes an administrative headache in larger networks.

3) The third security method uses the concept of domains or groups. The domain format arranges individual servers and users into groups, each with specific access rights. Novell's NetWare, for example, makes each user responsible only for his or her password and lets the LAN administrator easily move people into different groups as they change jobs and responsabilities or leave the organization.

(B) The problem of physical security is acute in DOS-based networks where many users have hands-on access to the server. By using the server's keyboard, intruders can access files on its hard-disk. Physical protection needs to be extended to any PC acting as a file server. That could mean removing the keyboard or locking the machine in a room. Of course, both options mean that users will not be able to use the machine as a workstation. Some LAN security devices provide a better option by locking the keyboard or disk drives on unattended servers.

BUILDING A SECURITY FIRE-WALL

Once an attacker has violated network security, it is relatively simple to jump from network to network. One way to reduce malicious damage and further intrusions is to build 'fire-walls'. A fire-wall consists of two parts: a gate and choke. A choke filters or blocks all packets from the outside network destined for the inside network, unless they are destined for the gate. Likewise, all packets from the inside are blocked from the outside unless they originated at the gate. The choke and gate can be the same

computer, although LAN interconnect devices are well suited to the choke task (e.g . a router). The gate has a stripped-down version of the LAN operating system and virus-protection software.

LAN SECURITY MECHANISMS/SERVICES

A distinction between security services and security mechanisms is that security services specify "what" controls are required; the security mechanisms specify "how" the controls are to be implemented. The selection of the appropriate mechanism is part of architectural considerations.

The following basic security services are :

(1) Access Control.
The access control service provides protection against unauthorized use of resources accesible via the network. This protection service is applied to various privileges of access to a resource (e.g. the reading, the writing, or the deletion of an information file or record in the file, or the execution of a program).

(2) Data Confidentiality.
These services provide for the protection of data from unauthorized disclosure. Four subtypes of these service are :

a) Connection confidentiality, provides for the confidentiality of all user data on a connection.

b) Connectionless confidentiality, provides for the confidentiality of all user data in a single connectionless message exchange.

c) Selective field confidentiality, provides for the confidentiality of selected fields (records) within the user data.

d) Traffic flow confidentiality, provides for the protection of the information that might be derived from observation of traffic flows.

(3) Data Integrity.
This service provides for the integrity of all user data or of some selected fields over a connection or connectionless message exchange; it detects any modification, insertion, or deletion of data. Four subservices are:

a) Connection integrity with recovery.
b) Connection integrity without recovery.
c) Selective field connection integrity.
d) Connectionless integrity.

(4) Nonrepudiation.
This service can take one or both of two forms :

a) Nonrepudiation with proof of origin; in this case, the recipient of data is provided with the proof of the origin of the data; this proof will protect the recipient against any attempt by the sender to deny falsely the sending of the data or its contents.

b) Nonrepudiation with proof of delivery; in this case, the sender of data is provided with the proof of the delivery of the data; this proof will protect the sender from any attempt by the recipient to deny falsely the receiving of the data or its contents.

(5) Authentication.
These services provide for the verification of the identity of a remote communicating entity 2and the source of data. Authentication consists of two services:

a) The Peer Entity Authentication Service, is used at the establishment of a connection or at times during the data transfer phase to confirm the identities of one or more of the entities connected. This service provides confidence, at the time of usage only, that the corresponding entity is not attempting a masquerade or an unauthorized replay of a previous connection message.

b) The Data Orign Authentication Service, provides corroboration to an entity in a particular layer that the source of the data is really the claimed peer entity.

SECURITY MECHANISMS DESCRIPTION

The main security mechanisms can be classified as preventive, detective and recovery. These are :

(1) Authentication Exchange Mechanism.
This mechanism provides corroboration that a peer entity is the one claimed. Specific mechanisms include authentication information, such as passwords, supplied by a sending entity and checked by the receiving entity, and cryptographic means. The mechanism may be incorporated into a layer in order to provide peer-to-peer entity authentication. If the mechanism does not succeed in authenticating the entity, the connection is rejected and terminated. An entry in the security audit trail or a report to a security management center may also be generated. When cryptographic techniques are used, they may be combined with handshaking protocols to protect against replay. In many cases, authentication exchange mechanisms will be used in conjunction with nonrepudiation services by digital signature or notarization mechanisms.

(2) Traffic Padding.
This mechanism provides generation of spurious traffic or filling of protocol data units to achieve constant traffic rates or message length. Traffic padding mechanisms can be used to provide various levels of protection against traffic analysis. This mechanism is only effective if traffic padding is protected by a data confidentiality service.

(3) Routing Control.
This is a mechanism for the physical selection of alternative routes that have a level of security consistent with that of the message being transacted. Routing control mechanisms ensure that the routes used by the data across the network are those that have been specified. Routes can be chosen either dynamically or by prearrangement so as to use only physically secure subnetworks or transmission links. The initiator of a connection or the sender of a connectionless message may specify routing instructions that request particular subnetworks or links to be avoided. This mechanism can direct the network service provider to establish a connection via a different route if persistent attacks on the initial route are detected. Data carrying certain security labels may be forbidden by the security policy to travel through certain subnetworks not cleared at the appropriate level. Routing control is not easy to implement in practice.

(4) Notarization Mechanism.
This security mechanism provides the assurance that the properties about the data communicated between two or more entities, such as their integrity, origin, time, and destination, are what they are claimed to be. The assurance is provided by a third-party notary, which is trusted by the communicating entities and which holds the necessary information to provide the required assurance in a manner that can be tested. Each communication channel may use digital signature, encipherment , and integrity mechanisms as appropriate to interface with the service being provided by the notary. When the notarization mechanism is invoked, the data are exchanged between the communicating entities via the protected communication channels and the notary.

(5) Pervasive Security Mechanisms.
These types of mechanisms that are not specific to any particular service; some of these security mechanisms can be regarded as aspects of security management. The major three follow :

a) Security Recovery, handles the requests originating from mechanisms such as event detection handling and management functions, and performs recovery actions as the result of applying a set of rules. The recovery actions may be of three types: immediate, temporary, or long term.

b) Event Detection. Security-related event detection encompasses the detection of apparent violations of security and may also include detection of normal events, such as successful network logons. Examples of security-related events are a specific security violation; a specific network logon; a specific selected event. Security-related events can be detected by entities. The specification of what constitutes an event is maintainned by event handling management facilities. Detection of various security-related events may, depending on implementation, trigger one or more of the following actions : local reporting of the event; remote reporting of the event; logging of the event; recovery action for the event. For each event, the relevant information for event reporting and event logging, and the syntactic and semantic definitions to be used for the transmission of event reporting and event logging, are standardized.

c) Security Audit Trails. A security audit permits detection and investigation of breaches of security by permitting a subsequent security audit. A security audit is an independent review and examination of system records and activities in order to ensure compliance with established policies and operational procedures, to test for the adequacy of system controls, to aid in damage assessment, and possibly to recommend any indicated change in controls, policies, and procedures. A security audit requires the analysis, recording, and reporting of security-related information found in the security audit trails. The security audit trail is considered to be a security mechanism because the known existence of a security audit trail may serve as a deterrent to some potential sources of security attacks.

(6) Digital Signature.
This is an encryption method that provides data integrity, namely, the guarantee that data have not been altered or destroyed in an unauthorized fashion. The digital signature is data appended to (or a transformation of) a protocol data unit (frame, packet,...) that allows a recipient to prove the source and integrity of the data.

Digital signatures are typically created by taking a message's content and running it through a program called a hashing function, which produces a binary number (a digest) that represents the file. That number is then encrypted using the sender's private key, creating the digital signature that is appended to the file. Both are then transmitted over the network. At the other end of the exchange, the recipient separates the digital signature from the file and uses the system's public key to decode the signature and reveal the binary hashed output. The file is then run through the same hashing function and the output is compared with the binary number that was contained in the digital signature. If the two hashed outputs match, the file is authenticated.

This system works for several reasons.If an impostor tries to send a message in someone else's name but uses the wrong private key to encrypt the signature, the two hashed outputs will not match. Similarly, if someone intercepts a message and changes part of it, increasing the amount of money in a wire transfer, for instance, the hashed outputs again will not match.

The RSA (Ronald Rivest-Adi Shamir-Leonard Adleman) encryption technology offers three hashing programs, which are also called message digest (MD) functions : MD2, MD4 and MD5. MD2, MD4 and MD5 produce 128-bit hashed outputs, called digests. Of the three, MD5 is considered the most secure.

These security mechanisms comprise two different processes :

a) The signing process uses information that is private, i.e. confidential to the signer. This process involves either an encipherment of the data unit or the generation of a cryptographic check-value of the data unit, using the signer's private information as a private key.

b) The verification process involves using the public procedures and information to determine whether the signature was produced with the signer's private information. The basic characteristic of the signature mechanism is that the signature can only be produced using the signer's private information. When the signature is verified, it can subsequently be proven to any third party that only the unique holder of the private information could have produced the signature.

(7) Access Control Mechanisms.
These mechanisms use the authenticated identity of an entity, its capabilities, or its credentials to determine and enforce the access rights of that entity. These mechanisms ensure that only authorized users have access to the protected facilities. If the entity attempts to use an authorized resource in the proper manner, then access to the resource is granted. If the entity attempts to use an unauthorized resource, or an authorized resource with an improper type of access, then the access control function will reject the attempt and may additionally report the incident for the purpose of generating an alarm or recording it as part of a security audit trail. The access control mechanisms may be applied at either end of a connection or to a connectionless message exchange. Five basic access mechanisms involve the use of the following tools :

a) Access control lists.
Lists of entities together with access rights to a resource.

b) Passwords.
Confidential information used to authenticate the user of a resource.

c) Capabilities lists.
An unforgeable identifier conferring access rights to a given user for a given resource.

d) Credentials.
Data passed from one entity to another that is used to establish the access rights of the requester entity.

e) Labels.
Tokens possessed by a user that confer specified access rights.

The implementation of access control mechanisms requires the existence of one or more of the following types of information :

a) Authentication information, such as passwords, the possession and subsequent presentation of which is evidence of the accessing entity's authorization.

b) Access control database, in which the access rights of authorized peer entities to resources are maintained; this information may be maintained by authorization centers or by the entity being accessed. The information may be in the form of an access control list or matrix.

c) Security labels (secret, top secret, ...) which are used to grant or deny access according to a security policy.

d) Other information, such as time of the attempted access, route of the attempted access, duration of access, etc.

e) Capabilities, the possession and subsequent presentation of which is evidence of the right to access the entity, or the resource defined by the capability.

Access control is based by definition on authenticated identification. Typically, a user claims the identity of a person or process operating on behalf of the person. Then the identity must be proven; in manual systems, a common proof is a credential carrying the photograph and signature of the individual. Authentication information must be validated before the user identification is accepted through comparison of known and presentes information. Automated identification is based on two methods :

i) The nonencryption method, rely on :

- Information the user has, such as passwords.

- Information about a physical characteristic, such as a traditional signature.

- Instruments the user possesses, such as a physical key or card (typically for challenge-and-response authentication).

These last two approaches require an appropriate entry device. The host computer normally stores the comparison information in encrypted form; when the user presents the authentication information, it is encrypted and then compared with the store cipher text. Challenge-and-response authentication eliminates the majority of the stored information because it assumes that the user possesses secret information. After the challenge is issued by the computer, the user can use devices such as portable cards or other portable devices to formulate a response; the response is compared with the expected string and the access is granted or denied. Handwritten signatures entered via a pressure-acceleration device can also be employed.

ii) The encryption approach is encryption-based authentication for access. This method assures that the message came from the claimed source and was delivered unchanged. The functional similarity to a handwritten signature has caused message authentication to be known as digital signature.

Access controls should prevent unauthorized local access to the network and control remote access through dial-up ports. The three minimum levels of user access usually asigned are: public, private and shared access. Public access allows all users to have read-only access to file information. Private access gives specific users read and write file access, while shared access allows all users to read and write to files. Perhaps the most difficult application to secure is the shared database environment offered by LANs. When a company offers network access to one or more databases, it must

restrict and control all user query operations. Each database should have a protective key, or series of steps, known only to those individuals entitled to access the data.

To ensure that intruders cannot duplicate the data from the system files, users should first have to sign on with passwords and then prove that they are entitled to the data requested. Passwords should have a minimum of six or seven characters; less, and they are too easily broken by brute-force guessing. Worth noting is that plain-text passwords are especially vulnerable on LANs, since each guess increases the chance of unauthorized entry by a factor of 1xN, where N equals the number of passwords on the LAN. A user identifier should be suspended after a certain number of passwords have been entered, reducing the chance of a trial-and-error procedure accessing the operating system.

There are two passwords-protection systems with which users can maintain security: hierarchical or specific. With hierarchical passwords, users can employ a defined passwords to gain access to a designated security level as well as all lower levels. With specific passwords, on the other hand, users can access only the intended level and not the others above or below. Although specific-level passwords offer more security, they require that a senior, trusted employee have many passwords in order to work with the many databases and associated levels used throughout the day. Password levels, especially specific levels, also make the task of network management more complex.

Once an administrator implements a particular password security method, he should ensure that the connected workstation play an active role in supporting password usage. For example, as a user enters the characters of the password, the monitor screen should automatically blank out all key entries to minimize the risk of exposing the password to casual observers. Administrators can also install passwords routines that do not display any information on the screen or that sound an audible alarm while locking keyboard input after a specified number of failed entry attempts. In addition, administrators should survey the password master file, keep the file on disk, change any infrequently used passwords, and review risks periodically whenever a breach of security is suspected. Although passwords are the most frecuently used access control method, their effectiveness depends on how rigorously they are enforced by network administrators. For this reason, administrators should always combine passwords security with a local physical access control measure, such as a keyboard lock, a card reader, or a biometric device (fingerprint, handprint, voice quality, pattern of capillary blood vessels in the retina of the eye, etc.).

(8) Data Integrity.
These mechanisms ensure that the data have not been alterd or destroyed. These security mechanisms deal with two aspects of data integrity: the integrity of a single data unit or field, and the integrity of a stream of data units or fields. Different mechanisms are used to provide these two types of integrity services. Two specific tools include checksums (cryptographic remainder) and sequencing (which assures that the protocol data unit sequence is correct).

The determination of the integrity of a single data unit involves symmetric processes, one at the sending entity and the other at the receiving entity. The quantity may be supplementary information such as a block check code or a cryptographic check-value and may itself be enciphered. The receiving entity generates a corresponding quantity and compares it with the received quantity to determine whether the data have been modified in transit. This mechanism alone does not protect against the replay of a single data unit. Detection of manipulation may trigger a recovery action at that layer or at a higher layer of the architecture. For example, the recovery may be performance by retransmission or error correction. The determination of the integrity of a stream of data units or fields varies for connection-oriented and connectionless information exchange.

For connection-oriented data transmission, protecting the integrity of a sequence of data units, namely, protecting against misordering, losing, replaying, inserting, or modifying data, requires some additional explicit ordering mechanism such as sequence numbering, time stamping, or cryptographic chaining. For connectionless data transmission, time stamping may be used to provide a limited form of protection against replay of individual data units.

(9) Encryption.
Encryption is probably the best available technique for providing security in data communications. Encryption (also called encipherment) is a key security mechanism that can provide confidentiality of either data or traffic flow; often it is used in conjunction with other mechanisms. Cryptography is the discipline involving principles, means, and methods for the mathematical transformation of data in order to hide its information content, prevent alteration, disguise its presence, or prevent its unauthorized use. The transformation uses confidential variables, called keys, to map between the plain text and the cipher text. Message content release and traffic analysis can be minimized by using this tool, which also affords protection against repudiation and forgery. This mechanism is required for many security services. Cryptography can be used as part of encryption, data integrity, data origin authentication, password storage, and checking. Encipherment algorithms can be divided into two categories :

a) Symmetric encipherment.
In a symmetric encipherment algorithm, the encipherment key is secret and the knowledge of the encipherment key implies the knowledge of the decipherment key and vice versa. When symmetrical or private-key encipherment is used, the key must be kept secret.

b) Asymmetric encipherment.
In an asymmetric algorithm, the encipherment key is public and the knowledge of the encipherment key does not imply the knowledge of the decipherment key or vice versa. The two keys of such a system are referred to as the private key and the public key, respectively.

Encryption can take place at a number of layers. The two types of encryption are :

i) Link-by-link encryption.

Link encryption has limitations in that data are handled in plain text in the communication processors, and it is inappropriate for broadcast networks. Link encryption protects individual links. All information passed to the physical link is encrypted. A procedural advantage of link encryption is that it can be implemented independently of the end systems; the end systems can continue to operate as they did prior to the introduction of the service, while at the same time achieving communication link protection. One disadvantage is that the encryption process must be applied at each link; at the nodes where links join, the message must be retransformed into clear text, introducing liabilities. All hosts and switching nodes of the network must be trusted to provide the necessary protection. Also, the problem of managing (storing and distributing) all the required keys is present. Additionally, link encryption is inappropriate for broadcast networks.

ii) End-to-End Encryption.

It is more robust, but it requires increased resources, and is subject to Trojan horse attacks and covert channel infiltration. End-to-end encryption is considerably more complex architecturally than link encryption, particularly if in-band electronic key distribution is included in the algorithms.

MECHANISMS FOR CHANNEL SEPARATION

There are a number of different schemes for separating the security groups using a LAN. The most obvious and straight forward is to provide separate LANs for each security group. Simply providing multiple cable may not be sufficient; some mechanism may be required to make sure that an unauthorized network interface unit does not get plugged into the wrong LAN. Another way to achieve logically separate channels is employed encryption. Assuming for the moment that the security group encompasses the network interface unit and that a network interface unit belongs to one and only one security group, then encryption between networks interface units is analogous to link encryption in conventional communications systems. Only those network interface units sharing the same encryption key (or pairs of keys) would be able to communicate thereby creating separate logical channels for their security groups. A message may be encrypted more than once, if the encryption algorithms permit. This serial encryption is sometimes called super-encryption. Some algorithms require decryption to be performed in the inverse sequence as encryption; other algorithms are insensitive to sequence of operations.

Encryption can be used in a rather straight forward way to establish logical channels. Each logical channel is assigned in encryption key (or pair of keys). When an network interface unit is going to operate on a given logical channel, it need only load the appropriate key. A pair of keys thus establishes a logical channel between network interface units. If a network interface unit is to be trusted to operate on multiple logical channels, then it need apply a key for each of those channels to every message which

passes through it. This seems to be an unnecessarily complex procedure; it would not be unreasonable to restrict a network interface unit to operating on a single logical channel. If the attached person or process had access to multiple logical channels, then the encryption key could be changed appropriately under the control of that person or process.

Encryption between data terminal equipments would require another pair of keys. Messages between data terminal equipments would then be doubly encrypted. Without this second data terminal equipmemt-to-data terminal equipment key, any data terminal equipment attached to a network interface unit possessing the key for the logical channel could eavesdrop on all communication on that logical channel.

A key is used in modern encryption systems to control the encryption and decryption algorithms which transform between plaintext and cyphertext. All keys and passwords must have a finite lifetime. There are loss of a key, employee turnover, and potential key breaking by crypto-analysis. This last problem derives from the property that allows a skilled adverse party to analyze encrypted messages to deduce the key used to encrypt them. The more messages which are available, the better the analysis. Given that keys must have a finite lifetime, we have the problem of distributing keys and maintaining a historical data base. The historical data base is necessary so that it is possible to ascertain what a particular user's or user group's key, was at a given point in time. Various combinations of manual and automated methods exist for distribution of keys.

One common scheme limits the lifetime of a key for communication between pairs of parties to a single session; a session key is generated every time a connection is made. A special master key is used to encrypt the messages used to distribute the session key. The distribution of the master key is usually outside the automated system. It would seem that the operation of key distribution center requires a great deal of trust, and the computer implementing this function requires a significant degree of physical protection.

BIBLIOGRAPHY

[1] TILBORG, H.C.A. "An Introduction to Cryptology". Kluwer Academic. Boston. 1988.

[2] MULLER, N.J. and DAVIDSON, R.P. "LANs to WANs: Network Management in the 1990s". Artech House. Norwood, Mass. 1990.

[3] FITES, P., JOHNSTON, P. and KRATZ, M. "The Computer Virus Crisis". Van Nostrand Reinhold. New York. 1989.

[4] DENNING, P.J. "Computers under Attack. Intruders, Worms, and Viruses". Addison-Wesley. Berkshire. 1990.

[5] CONARD, J.W. "Handbook of Communications Systems Management". Aurbach. Boston. 1988.

[6] CHOU, W. "Computer Communications". Prentice-Hall. Englewood Cliffs. NJ. 1983.

[7] CARROLL, J. "Computer Security". Second Edition. Butterworths. Boston. 1987.

[8] CAILLE, W., LONGLEY, D. and SHAIN, M. "Information Security for Managers". Stockton Press. New York. 1989.

[9] AREITIO, J. and AREITIO, M.T. "Mecanismos de Protección en Telemática". Mundo Electrónico. Grupo CEP Communication. Cetisa-Boixareu Editores, S.A. No. 238. Pp. 24-29. Mayo 1993.

[10] AREITIO, J. and AREITIO, M.G. "Servicios de Seguridad para Sistemas Distribuidos Abiertos". Revista de la Seguridad Informática. Ediciones CODA, S.L. Pp. 24-26. No. 5. Marzo 1993.

EUROPEAN CONVEX USERS CONFERENCE

Thursday, 28 October, 1993

EFFICIENT IMPLEMENTATION OF BLOCK-ITERATIVE TECHNIQUES FOR NON-LINEAR COUPLED PROBLEMS

G. Bugeda, M. Cervera and M. Galindo
E. T. S. Ingenieros de Caminos, Canales y Puertos, Barcelona, Spain

SUMMARY

In this paper we outline a general methodology for the solution of the system of algebraic equations arising from the discretization of the field equations governing coupled problems. In the example, we shall consider that this discrete problem is obtained from the finite element discretization in space and the finite difference discretization in time. Our motivation is to preserve software modularity, to be able to use existing single field codes to solve more complex problems, and to exploit computer resources optimally, emulating parallel processing. To this end we deal with a well-known coupled problem of computational mechanics: the fluid-structure interaction problem. The possibility of coupling the block-iterative loop with the nonlinearity of the problems is demonstrated through numerical experiments, which suggest that even a mild nonlinearity drives the convergence rate of the complete iterative scheme, at least for the two problems considered here. The paper discusses the implementation of this alternative to the direct coupled solution, stating advantages and disadvantages. The need for on–line synchronized communication between the different codes used is also explained, together with the description of the master code who will control the overall algorithm.

1. INTRODUCTION

Coupled problems arise frequently in engineering applications. As defined in reference [1], *"coupled systems and formulations are those applicable to multiple domains and dependent variables which describe different physical phenomena and in which (a) neither domain can be solved while separated from the other; and (b) neither set of dependent variables can be eliminated at the differential equation level"*. It is also usual [1] to classify coupled systems in two categories: (I) those problems in which coupling occurs on domain interfaces via the boundary conditions imposed there, and (II) those problems in which the coupling comes from different physical phenomena which occur on (totally or partially) overlapping domains. The methodology to be described in the following has been applied to problems falling in both groups [7]. In this paper, an example of application to a group (I) problem is presented.

Numerical methods applied to these coupled problems lead to the solution of a set of nonlinear algebraic equations which necessarily involve the (nodal) variables corresponding to the various domains (for Class I) or to the various physical phenomena (for Class II). Thus, the alternatives to solve a coupled problem are two–fold:

(a) To treat all the domains simultaneously. This leads to a single set of algebraic equations involving all the relevant variables. In general, these variables will not be homogeneous, as they represent discretization of different domains and/or different physical phenomena.

(b) To treat the domains one at the time, considering the coupling terms as forcing terms on the right–hand side of the equations. This leads to several sets of algebraic equations (one per domain), each of them to be solved solely for the variables related to one domain, but with the right hand side depending on variables related to the other domains.

Strategy (a) necessarily requires the development of a special–purpose code, probably involving collaboration from different expertise areas. Standard engineering software developed for uncoupled problems may be of little help when writing such a program, due to its particular structure. The outcome of this *may well be* a complicated code, difficult to maintain, modify or upgrade, and even difficult to use. This program could only be parallelized at basic instruction level. Moreover, even though for a 'standard' coupled problem this alternative could make sense, the effort that it involves is hardly affordable for all

the coupled problems one must be ready to solve in engineering practice. For example, let us mention two other particular problems to which we have been faced recently. One of them arises when the deformation of the roll in the rolling process of a flat metal plate has to be taken into account in order to reproduce more precisely the final shape of the metal piece. Another coupled problem has been encountered in the thermal analysis of a mould filling simulation of a casting process. The boundary condition for the temperature inside the cavity of the mould depends on the temperatures of the mould itself via the classical Robin condition, and these temperatures are in turn determined by those of the fluid filling the cavity.

Strategy (b), on the other hand, allows each domain/problem to be tackled on its one. The codes used may be either new or existing programs, slightly modified to account for the coupling terms. Each of these codes may be developed by a different expert or team of experts on the particular field, using optimal (and different) strategies for each of them. The outcome of this *should be* a set of (relatively) simple programs, easy to maintain, modify or upgrade, one independently of the others. Note that this approach is parallel *by construction*, and at module level. On a multi–user non–parallel machine, this approach turns the coupled problem being run into several interconnected processes. So, the kernel of the operating–system is working, automatically and inadvertedly, as a pseudo–parallel processor emulator.

Our first concern in this paper will be to study the application of Strategy (b) to the non–linear coupled problems mentioned earlier and, in particular:

- To cast the fully discrete finite element equations of the problems in the general framework of block-iterative techniques.
- To consider the linearized forms of these equations and to justify a unique iterative loop to deal both with the coupling and the nonlinear terms.
- To study the numerical performance of the final scheme from the standpoint of cost–effectiveness and rate of convergence.
- To justify the use of block–iterative methods vs the direct solution of the original coupled systems.

Of course, Strategy (b) has some drawbacks, and for some specific coupled problems these may prove this alternative impractical. In any case, if the goal outlined above, namely, modularity of the software used for coupled problems, is to be achieved, a corner stone will be required: a *master code* that will perform, at least, three tasks:

- To communicate the different *slave codes/processes*, transfering information from one program/domain to another, if (and only if) required.
- To control the overall coupled algorithm, checking convergence, forcing the *slave codes/processes* to iterate if (and only if) required; and
- To synchronize the execution of the different *slave codes/processes*, ensuring that each process runs as efficiently as possible, and that all of them iterate and/or advance in time simultaneously.

Having these general objectives in mind, the paper is organized as follows. In the next two sections we describe in more detail options (a) and (b), viewing the latter as a block–iterative procedure which can be coupled with nonlinear iterative loops. In Section 5 this strategy is applied to the fluid–structure interaction problem showing some numerical results that illustrate our discussion. Once the use of block–iterative methods is justified, Section 4 is concerned with the description of the basic characteristics of a computer code whose goal is to interact single field analizers in order to solve coupled problems. The modifications to be introduced to the original codes are also indicated. It is shown that only a few coding lines have to be added, so that the solution of *any* coupled problem turns out to be an extremely simple task once the above mentioned master code is available.

2. ALGORITHMIC SOLUTION FOR STRATEGY (a)

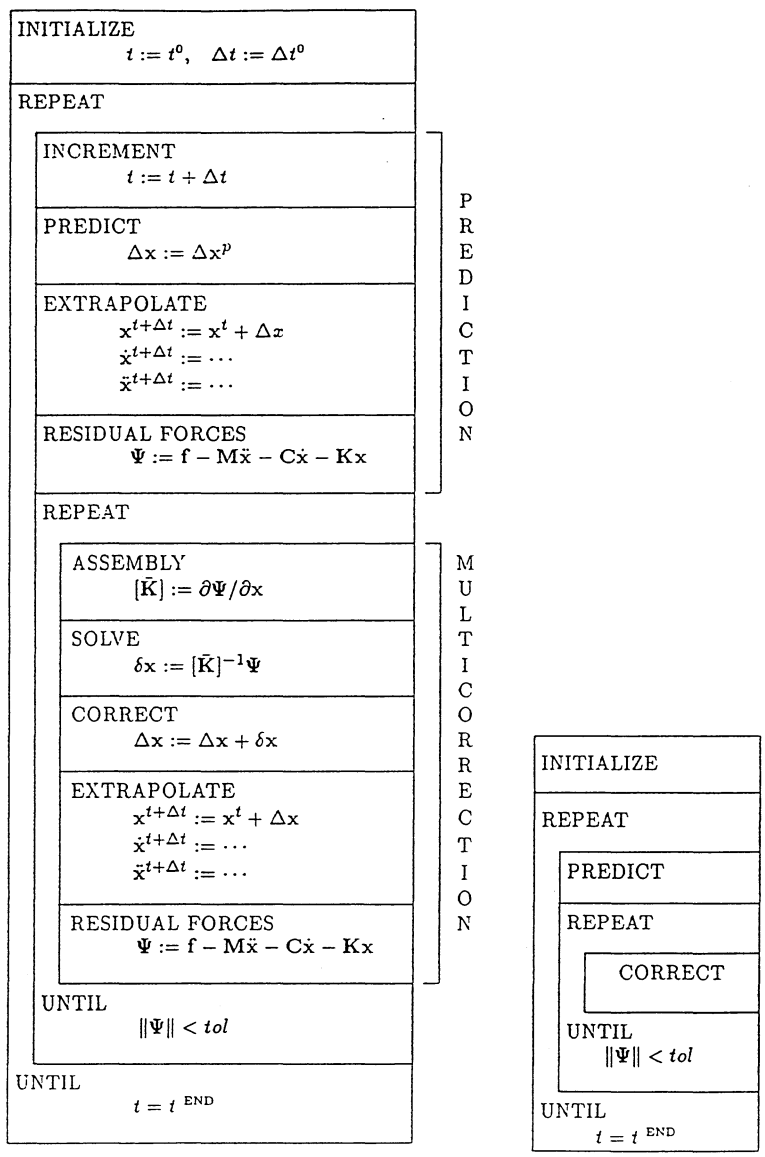
The discretization of the continuous problems to be considered will lead to a nonlinear algebraic system of the form

$$\begin{bmatrix} \mathbf{A}_{11}(\mathbf{x}) & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22}(\mathbf{x}) \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix} \tag{1}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the vectors of nodal unknowns at a certain time step of the two fields under consideration, $\mathbf{f}_1$ and $\mathbf{f}_2$ are the vectors of 'force' terms and $\mathbf{A}_{ij}$, $i,j = 1,2$, are matrices, the dependence of which on the unknown $\mathbf{x}$ has been explicitly indicated. The discussion of problem (1) is enough for our purposes, although it would be straightforward to extend what follows to other situations, such as several-fields problems or other nonlinear dependencies. Observe that in problem (1) a *linear* coupling of

the first equation with the second is assumed, as well as a *linear* behavior of **y** for a given **x**.

The algorithm for the direct solution of problem (1), Strategy (a), is outlined in Box 1. A prediction–multicorrection scheme [3] has been assumed for the solution of the nonlinear problem, without loss of generality. This would be naturally the case for a second order in time problem such as the fluid–structure interaction one, solved via the Newmark method. Note that the resulting algorithm follows exactly the process one would follow to solve an uncoupled problem of similar characteristics, such as one of (uncoupled) dynamic structural analysis. One disadvantage of this strategy is that the structure of the global matrices $\mathbf{A}_{ij}$ is such that entries come from the two different fields, and so, either integrals have to be evaluated in two different domains (Class I problems), or they represent physically different magnitudes (Class II problems). Another disadvantage is the larger size of the global matrix as compared with the ones arising from the different domains/fields. On the other hand, the advantage is that the final algorithm is easily and clearly defined, and its analysis, regarding for instance stability, is feasible.



Box 1. Computational algorithm for Strategy (a).

3. ALGORITHMIC SOLUTION FOR STRATEGY (b)

Let us consider now the use of block-iterative algorithms to solve problem (1). This will reduce the size of the resulting subproblems at the expense of iterating. Assuming that the first equation in (1) is solved first, there are two possible block-iterative schemes, namely,

$$\begin{aligned} \mathbf{A}_{11}(\mathbf{x}^{(i)})\mathbf{x}^{(i)} &= \mathbf{f}_1 - \mathbf{A}_{12}\mathbf{y}^{(i-1)} \\ \mathbf{A}_{22}(\mathbf{x}^{(i-1)})\mathbf{y}^{(i)} &= \mathbf{f}_2 - \mathbf{A}_{21}\mathbf{x}^{(k)},\ k = i-1 \text{ or } i \end{aligned} \tag{2}$$

Here, subscripts in parenthesis refer to iteration counters. For $k = i-1$ this is the *block-Jacobi* (or *block-total-step*) method, whereas for $k = i$ it is the *block-Gauss-Seidel* (or *block-single-step*) method. Observe that the matrix $\mathbf{A}_{22}$ is evaluated using $\mathbf{x}^{(i-1)}$. The use of $\mathbf{x}^{(i)}$ would imply a certain treatment of the nonlinear term $\mathbf{A}_{22}(\mathbf{x})\mathbf{y}$, as will be shown below. From elementary numerical analysis it is known that, under certain conditions, both the Jacobi and the Gauss-Seidel methods converge linearly when applied to linear systems, the convergence rate of the latter being twice higher than that of the former. In their *block* counterparts these properties are inherited, the convergence rate depending now on the spectral radius of the matrices involved.

On the other hand, problem (1) (and also problem (2)) is nonlinear, so that an iterative procedure must be used to deal with this nonlinearity. Both the Picard (or fixed point) and the Newton-Raphson methods are defined by approximating

$$\mathbf{A}_{11}(\mathbf{x}^{(i)})\mathbf{x}^{(i)} \approx \mathbf{A}_{11}(\mathbf{x}^{(i-1)})\mathbf{x}^{(i)} + \beta_1 \mathbf{A}^*_{11}(\mathbf{x}^{(i-1)})(\mathbf{x}^{(i)} - \mathbf{x}^{(i-1)}) \tag{3}$$

$$\mathbf{A}_{22}(\mathbf{x}^{(i)})\mathbf{y}^{(i)} \approx \mathbf{A}_{22}(\mathbf{x}^{(i-1)})\mathbf{y}^{(i)} + \beta_2 \mathbf{A}^*_{22}(\mathbf{x}^{(i-1)}, \mathbf{y}^{(i-1)})(\mathbf{x}^{(i)} - \mathbf{x}^{(i-1)}) \tag{4}$$

where $\beta_j = 1$ or 2 determines if the jth equation in (1) ($j = 1, 2$) is linearized using the Picard or the Newton-Raphson schemes, respectively. Matrices $\mathbf{A}^*_{11}$ and $\mathbf{A}^*_{22}$ arise from the derivation of $\mathbf{A}_{11}$ and $\mathbf{A}_{22}$ with respect to $\mathbf{x}$.

Using expressions (3) and (4), the linearized version of problem (1) is

$$\begin{bmatrix} \mathbf{A}_{11}(\mathbf{x}^{(i-1)}) + \beta_1 \mathbf{A}^*_{11}(\mathbf{x}^{(i-1)}) & \mathbf{A}_{12} \\ \mathbf{A}_{21} + \beta_2 \mathbf{A}^*_{22}(\mathbf{x}^{(i-1)}, \mathbf{y}^{(i-1)}) & \mathbf{A}_{22}(\mathbf{x}^{(i-1)}) \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(i)} \\ \mathbf{y}^{(i)} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 + \beta_1 \mathbf{A}^*_{11}(\mathbf{x}^{(i-1)})\mathbf{x}^{(i-1)} \\ \mathbf{f}_2 + \beta_2 \mathbf{A}^*_{22}(\mathbf{x}^{(i-1)}, \mathbf{y}^{(i-1)})\mathbf{x}^{(i-1)} \end{bmatrix} \tag{5}$$

Problem (2) is nonlinear and problem (5) is linear, but coupled. For each problem, either the nonlinearity or the coupling could be dealt with in a nested iterative loop. However, there is the strong temptation to use a single iterative loop to deal both with the nonlinearity and the coupling. Starting from problem (5), this would lead to:

$$\left[\mathbf{A}_{11}(\mathbf{x}^{(i-1)}) + \beta_1 \mathbf{A}^*_{11}(\mathbf{x}^{(i-1)})\right] \mathbf{x}^{(i)} = \mathbf{f}_1 + \beta_1 \mathbf{A}^*_{11}(\mathbf{x}^{(i-1)})\mathbf{x}^{(i-1)} - \mathbf{A}_{12}\mathbf{y}^{(i-1)} \tag{6}$$

$$\mathbf{A}_{22}(\mathbf{x}^{(i-1)})\mathbf{y}^{(i)} = \mathbf{f}_2 + \beta_2 \mathbf{A}^*_{22}(\mathbf{x}^{(i-1)}, \mathbf{y}^{(i-1)})\mathbf{x}^{(i-1)} - \beta_2 \mathbf{A}^*_{22}(\mathbf{x}^{(i-1)}, \mathbf{y}^{(i-1)})\mathbf{x}^{(k)} - \mathbf{A}_{21}\mathbf{x}^{(k)} \tag{7}$$

where $k = i-1$ or $k = i$. Either for $\beta_2 = 0$ (Picard linearization of $\mathbf{A}_{22}(\mathbf{x})\mathbf{y}$) or for $k = i-1$ (block-Jacobi coupling) Eq. (7) reduces to

$$\mathbf{A}_{22}(\mathbf{x}^{(i-1)})\mathbf{y}^{(i)} = \mathbf{f}_2 - \mathbf{A}_{21}\mathbf{x}^{(k)} \tag{8}$$

From Eq. (4) it may be concluded that for $\beta_2 = 1$ (Newton-Raphson's linearization of $\mathbf{A}_{22}(\mathbf{x})\mathbf{y}$) *and* $k = i$ (block-Gauss-Seidel coupling) Eq. (7) is simply

$$\mathbf{A}_{22}(\mathbf{x}^{(i)})\mathbf{y}^{(i)} = \mathbf{f}_2 - \mathbf{A}_{21}\mathbf{x}^{(i)} \tag{9}$$

This is perhaps the simplest choice to implement: once $\mathbf{x}^{(i)}$ is known by solving Eq. (6), it is used to evaluate the matrix $\mathbf{A}_{22}$ and then Eq. (9) may be solved. We stress that this *natural* approach implies a high order linearization of the second equation in problem (1) *and* the use of the block-Gauss-Seidel coupling.

The algorithm for the block–iterative solution of problem (2), Strategy (b), is outlined in Box 2 *for field number 1*. Obviously, field number 2 would follow an identical procedure. Again, the use of a prediction–multicorrection scheme has been assumed. Note that the depicted algorithm follows very closely the

process described in Box 1, apart from the *export/import operations*, the evaluation of the interaction terms, and the *unmodify/modify steps* which save unnecessary re–evalutions of the internal forces.

The advantage is now that the structure of the matrices in the LHS, $\mathbf{A}_{11}$ and $\mathbf{A}_{22}$ is such that entries come only from the field currently considered, and so, integrals have to be evaluated only in that domain, and they represent physically homogeneous magnitudes. Moreover, for many practical applications these matrices are symmetric. The difficulty of evaluating the terms appearing in the RHS of Eq. (2) is problem dependent. For Class I problems, they only involve integrals over the "interaction boundary", and this is "seen" from both domains. For Class II problems, the domains where the two equations in Eq. (2) hold overlap totally or partially, so that difficulties can only arise if different meshes or interpolations are used for both fields. Finally, note that the *two* systems of equations to be solved are smaller in size and with reduced band–width, as well as better conditioned, as compared to that yielding from Strategy (a).

On the other hand, the disadvantage of the block–iterative solution of Eq. (2), is that iterations will be needed *even if the problem is linear* (note the need to check global or overall convergence in Box 2). This is not specially inconvenient if the problem is nonlinear, as equilibrium iterations would be required anyway, or if the coupling effect is not too strong. Regarding the stability analysis of this approach, it is certainly complicated and very problem dependent. It may happen that, even if an unconditionally stable algorithm has been used for every one of the fields, the overall algorithm may still be conditionally stable [4]. In practice, as the size of the time step will always be limited by accuracy considerations, this may not be a serious drawback. Stabilization methods have been proposed to ensure unconditional stability of the block–iterative solution for specific problems under certain circumstances [1,4], but they require matrix operations that destroy the modularity (and so, the main motivation) of this approach. A new formulation that seems to preserve modularity has been recently proposed [5] for thermo–mechanical problems.

An important point to remark regarding stability of this iterative technique is that time integration stability will also depend on the tolerance demanded to achieve overall convergence. The algorithm will be, in general, conditionally stable, provided *full* overall convergence is achieved for each time step. However, as this is impossible or impractical to manage in real computations, a new source of instability will come from the tolerance specified by the user. Sensibility to this factor is again very problem dependent. No difficulty was found by the authors in the numerical examples run for fluid–structure interaction (see Section 5), even if slack tolerances were used. On the other hand, this dependence of the stability on the convergence tolerance used was evident in some thermally–coupled flows for specific situations solved in reference [7].

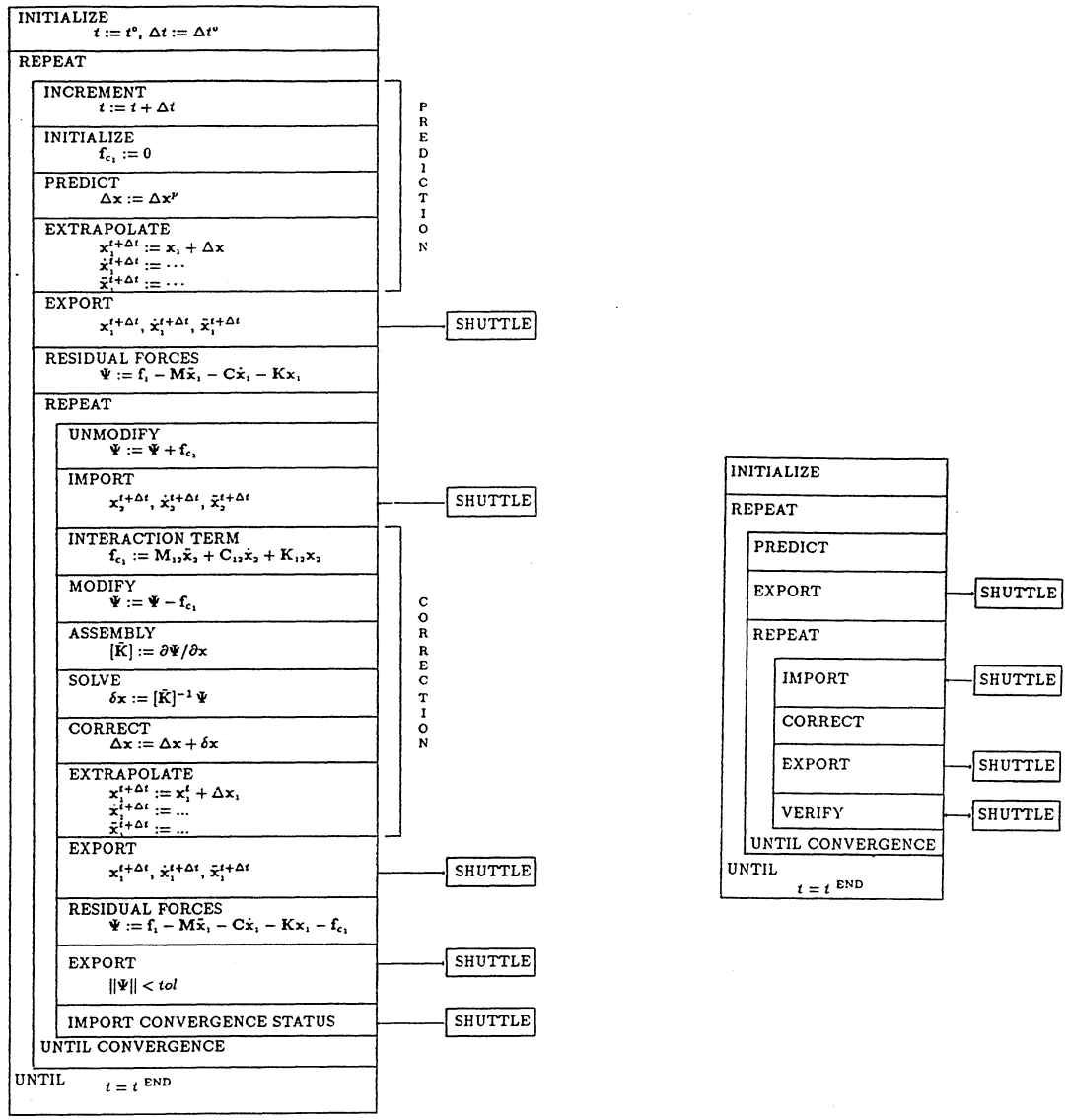
In the following, we shall apply these ideas to a particular problem, showing that dealing with the nonlinearity and the coupling within a single iterative loop is effective. Roughly speaking, the number of iterations due to the nonlinearity and the coupling *do not* add up. Even with weak nonlinearities, they are who govern the global process, so that the coupling is achieved automatically, and unexpensively.

4. PROGRAM *SHUTTLE* AND PARALLEL–PROCESSING

SHUTTLE is the *master code* that controls the overall block–iterative algorithm described in Section 3. To achieve this, it must perform the following tasks, regarding the execution of the *slave codes* who are responsible for the solution of each of the single field equations in Eq. (2):

- To communicate the different *slave codes/processes*, transfering information from one program/domain to another, if (and only if) required.
- To control the overall coupled algorithm, checking convergence, forcing the *slave codes/processes* to iterate if (and only if) required; and
- To synchronize the execution of the different *slave codes/processes*, ensuring that each process runs as efficiently as possible, and that all of then iterate and/or advance in time simultaneously.

The implementation of this code is based on the facilities for inter–process communication that the *UNIX* operative system has avalaible to the user. Under this system, any process can create a *socket* that immediately is made accessible to other processes. When a process creates a socket, the operating system stops this process until another one connects itself to the same socket. At this moment, both processes are re–started and the system establishes a communication pipe between them so that any information written by a process into the pipe can be accessed by the other. Details on the use of this *UNIX* facility

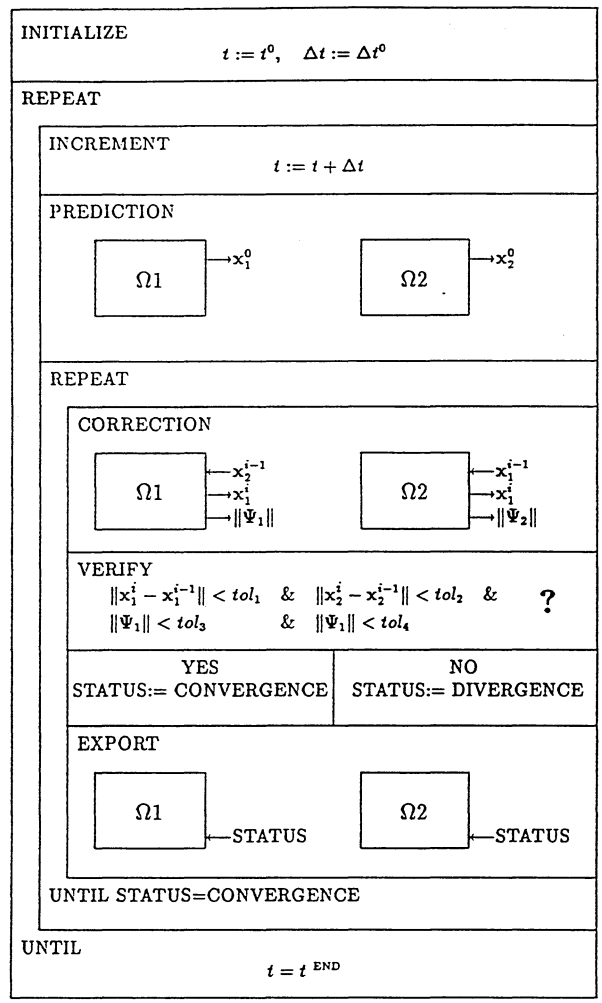


Box 2. Computational algorithm for Strategy (b).

are not given here, but they can be found in any standard user manual.

The Task Diagram for the master code is shown in Box 3. Note that it follows, as closely as possible, that of each slave code (see Box 2). Note that the task labelled "PREDICTION" consists on importing the individual predictions from the slave codes. The task labelled "CORRECTION" consists on: (a) exporting to all the processes the variables from the previous iteration, and, (b) importing from all of them the iterative corrections and the local convergence status. The task labelled "CONTROL" consists on checking that all the tolerances specified for each of the processes as satisfied, both at local (individual) level as globally. Finally, the task labelled "SYNCHRONIZE" consists on exporting to each process the overall convergence status for the current iteration.

The code is programmed following a hierachy of objectives. Each objective is fulfilled by achievement of sub–objectives. The main objective is obviously that the analysis arrives at the final time; this is done in a step–by–step fashion. For each time step, the objective is to achieve global and local convergence; this is attained forcing all the codes to perform the necessary iterative corrections. During each iteration, the objective is to furnish the slave codes with the required data to evaluate the coupling terms; this is done transferring the necessary information through the *sockets*. Finally, for each communication the objective is to preserve the prestablished turn and the natural dependencies of the coupled problem. So,



Box 3. Task Diagram for Program *SHUTTLE*.

the key–word for the whole procedure is *synchronized communication*. This framework of objectives can be sketched as follows,

- To arrive at the final time of the analysis, advancing in a step–by–step fashion.
  - For each step: to achieve local and global convergence, iterating if necessary.
    - For each iteration: to furnish the required to each one of the slave codes through the sockets.
      - For each communication: to ensure the prestablished turn and the natural dependencies of the coupled problem.

The program so constructed achieves a fringe, but not minor, benefit: to exploit the computer resources optimally, forcing the execution of the slave codes to proceed *in parallel*, or as much in parallel as possible. Let us direct some attention to this matter. For a two field problem, two alternatives are possible for the iterative multi–correction procedure: (a) one process, say Ω_1, iterates in first turn, while the other, Ω_2, follows; and (b) both processes, Ω_1 and Ω_2, try to iterate simultaneously. Fig. 1 shows both schemes.

Alternative (a) does not seem to allow for parallel processing. However, and referring to Box 2, each process exports its variables as soon as possible, that is, before getting entangled with the cumbersome residual forces evaluation. Therefore, just after Ω_1 has exported its variables to *SHUTTLE*, they can be imported by Ω_2, who can start correcting long before Ω_1 has finished doing so. For nonlinear problems, where residual forces evaluation can take quite a percentage of the total CPU time, this is as close as one can get to parallel processing (see Fig. 2).

Alternative (b) is, on the other hand, the obvious choice for parallel processing, but this may be mis-

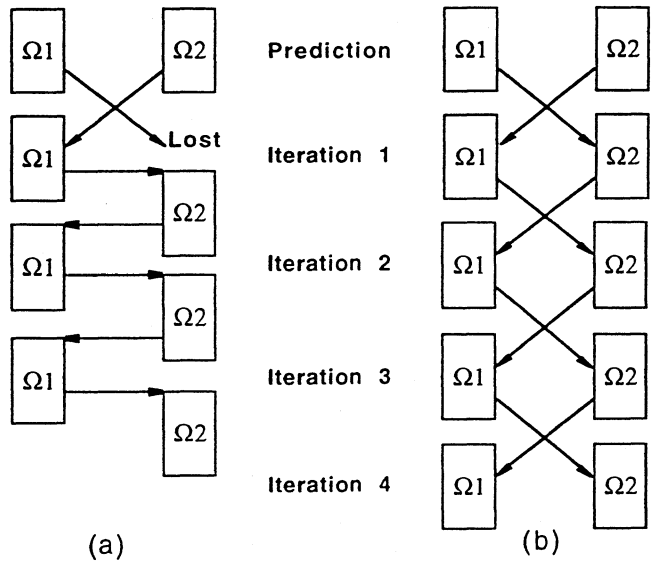


Figure 1. Alternatives scheme for iterative multi–correction:
(a) Ω_1 goes first and Ω_2 follows.
(b) Ω_1 and Ω_2 run simultaneously.

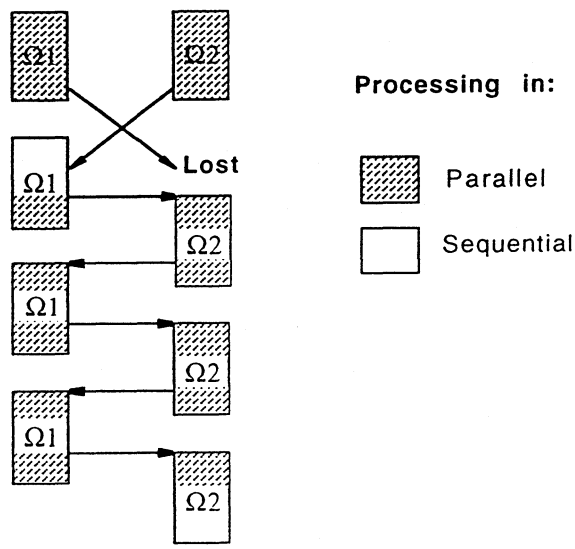


Figure 2. Parallel and sequential processing for alternative (a).

leading. Consider, for instance, the case when *both problems are linear*, and iterating is only required due to the block–iterative scheme adopted. The situation is depicted in Fig. 3. Note that the *S*–shaped sequence marked as *A* is exactly the same for both alternatives. However, alternative (b) spends almosts double CPU time. Nevertheless, alternative (b) can be satisfactory when the problems to be solved are strongly nonlinear on their one, and the interaction terms play a minor role in the iterative process. In this case, both processes would be running in parallel and fully exploiting the computational resources, and the global check on convergence will be automatically satisfied as soon as both processes converge locally.

5. A CLASS I PROBLEM: FLUID–STRUCTURE INTERACTION

Doubtless, this is one of the best known coupled problems in engineering. In this case, the coupling occurs at the interface between two different domains, occupied one by the solid and the other by the fluid. Neither the structure nor the fluid can be solved independently of each other, since the motion of

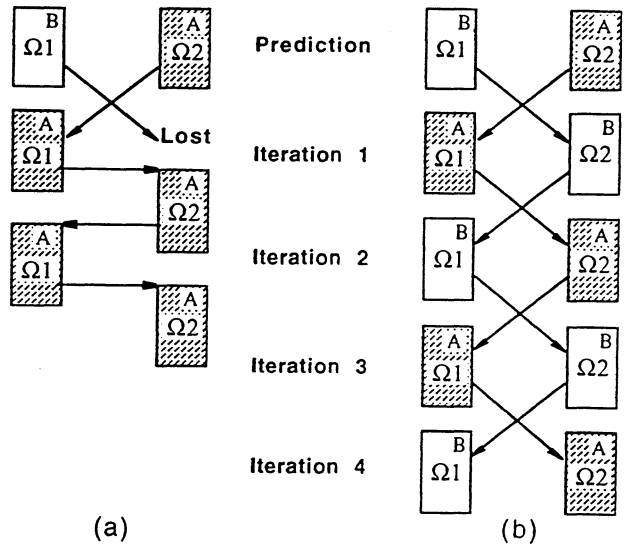


Figure 3. Alternatives (a) and (b) applied to a linear–linear coupled problem.

the structure depends on the pressures of the fluid at the interface, and the fluid pressures depend in turn on the normal acceleration of the wet wall of the solid.

To fix ideas, a particular problem will be considered, namely, the transient analysis of a dam subject to a dynamic excitation. The mathematical model considered here is the Helmholz equation for the fluid and the conservation of momentum for the solid, both equations being coupled through the (moving) interface boundary terms. The constitutive relationship adopted for the structure is a nonlinear isotropic damage model suitable for concrete, whereas a linear elastic model is assumed for the foundation [6].

Without going into details (see, e.g., reference [1]), the semidiscrete problem arising from the standard Galerkin spatial discretization reads as follows:

$$\begin{aligned} \mathbf{M}_s\ddot{\mathbf{a}} + \mathbf{C}_s\dot{\mathbf{a}} + \mathbf{S}(\mathbf{a}) &= \mathbf{f}_s - \mathbf{Q}^T\mathbf{p} \\ \mathbf{M}_f\ddot{\mathbf{p}} + \mathbf{C}_f\dot{\mathbf{p}} + \mathbf{K}_f\mathbf{p} &= \mathbf{f}_f + \rho_f\mathbf{Q}\ddot{\mathbf{a}} \end{aligned} \tag{10}$$

where subscripts s and f refer to the solid and the fluid, respectively. The notation involved in (10) is as follows. The mass and damping matrices are $\mathbf{M}$ and $\mathbf{C}$, $\mathbf{K}_f$ is the matrix arising from the discretization of the Laplace operator, $\mathbf{f}$ are the vectors of force terms, $\mathbf{a} = \mathbf{a}(t)$ is the vector of nodal displacements unknowns in the solid, $\mathbf{p} = \mathbf{p}(t)$ the vector of nodal pressures in the fluid, whose density is ρ_f, $\mathbf{S}(\mathbf{a})$ is the vector of internal forces in the solid ($\mathbf{S}(\mathbf{a}) = \mathbf{K}_s\mathbf{a}$ if a linear constitutive model is adopted), $\mathbf{Q}$ is the coupling matrix and the dot denotes differentiation with respect to time t. The damping matrices account for both the natural damping (assumed to be of Rayleigh type) and radiation boundaries (see, e.g., reference [6] for details). The rectangular matrix $\mathbf{Q}$ comes from the integral over the fluid-solid interface of the product of pressure and displacement shape functions and the unit normal to this interface.

What is interesting for us is the structure of the equations (10). Observe that the coupling is *linear* and the only nonlinearity of the problem comes from the constitutive model for the solid (the dam). Therefore, after using a suitable time discretization (the Newmark scheme, for example) one will be led to a nonlinear algebraic system of the form (1), but now matrix $\mathbf{A}_{22}$ being independent of $\mathbf{x}$, provided this is identified with the displacements and $\mathbf{y}$ with the fluid nodal pressures at a certain time step. Moreover, in this case the system can be easily symmetrized by scaling properly the equation for the fluid pressure.

We are now in a position to apply the iterative techniques described in the previous section. The results to be presented here correspond to the transient analysis of the dam in Fig. 4, where also the finite element mesh for the dam, the foundation and the reservoir is shown. The soil is excited by prescribing a senoidal wave of varying amplitude, also senoidal. Due to the damage model we have employed, the displacements in the dam do not depend linearly on the amplitude of the wave. The horizontal displacement at the left top of the dam is plotted in Fig. 5(a), and the evolution of the global damage index in Fig. 5(b).

Let us discuss now the convergence of the numerical method using the block-Jacobi and the block-Gauss-Seidel schemes. The nonlinearity of the problem will be included in the same iterative loop. However,

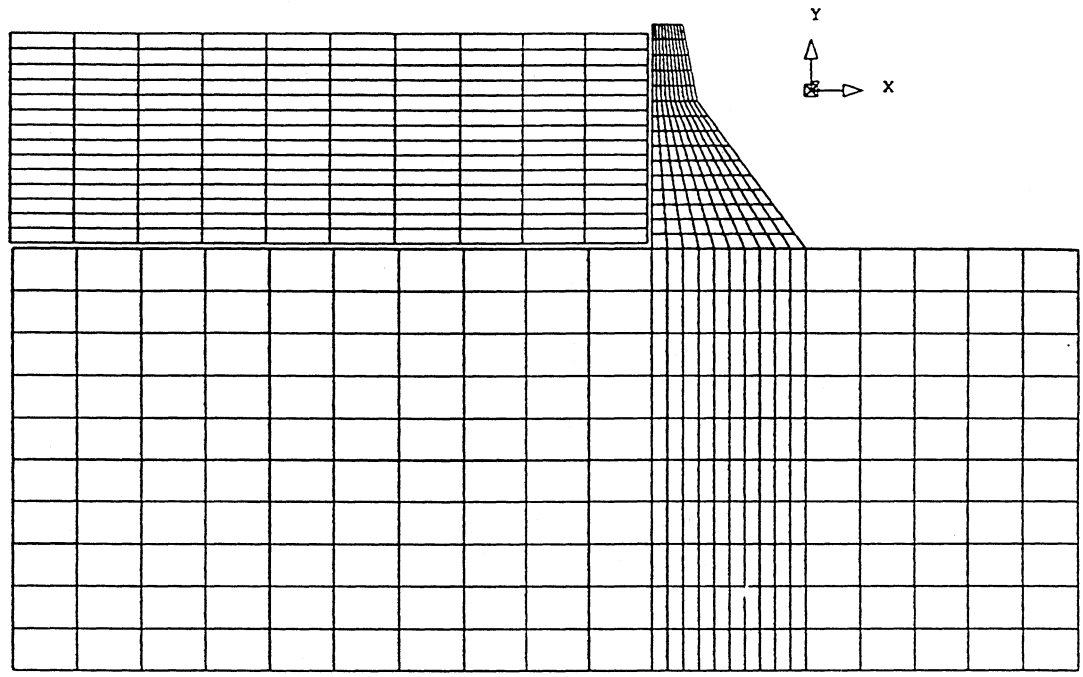


Figure 4. Computational model showing the dam, the foundation and the reservoir

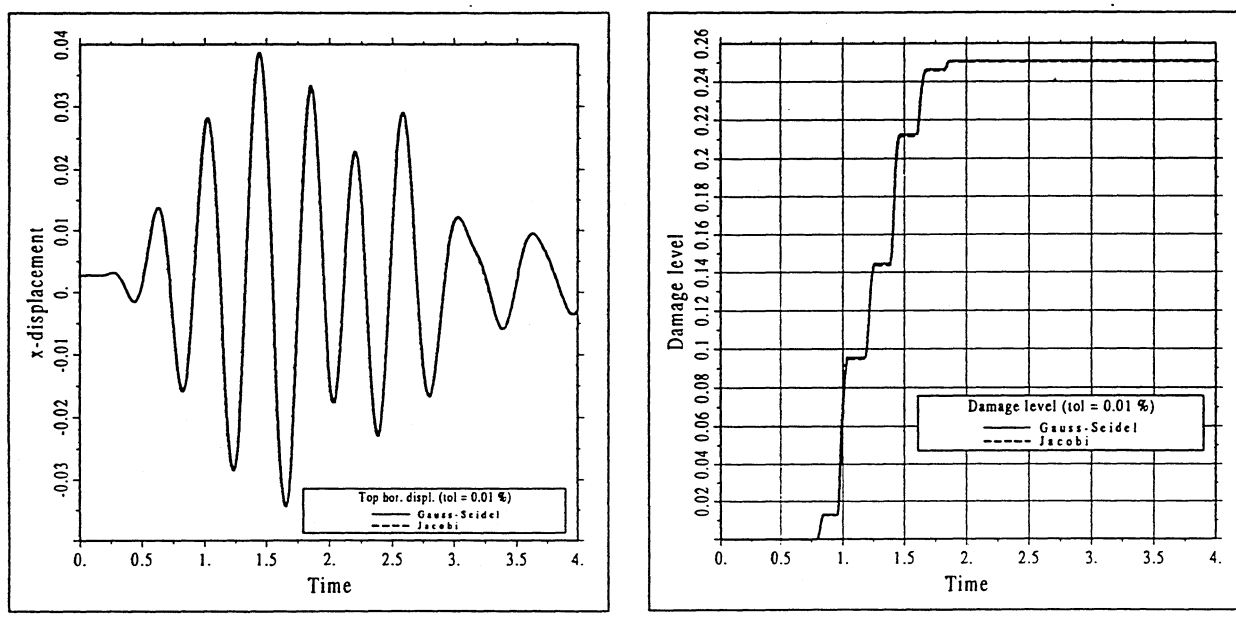


Figure 5. (a) Horizontal displacement of the left top of the dam, (b) Time evolution of the global damage index

from the evolution of the global damage index it is observed that a nonlinear behavior will be encountered only in some time intervals when this coefficient increases. Therefore, this example will serve to compare the performance of the iterative processes both for linear and nonlinear situations. The convergence tolerance in displacements has been set to 0.01 %.

The number of iterations required to converge vs time is plotted in Fig. 6(a) . In this case, the Picard scheme has been used to linearize the equations. The peaks correspond to times when the damage coefficient increases, that is, to nonlinear situations. It is observed that both the Jacobi and the Gauss-Seidel methods need the same number of iterations, and hence it may be concluded that *convergence is driven by the nonlinearity*. When the dam behaves linearly, the Jacobi scheme needs roughly double

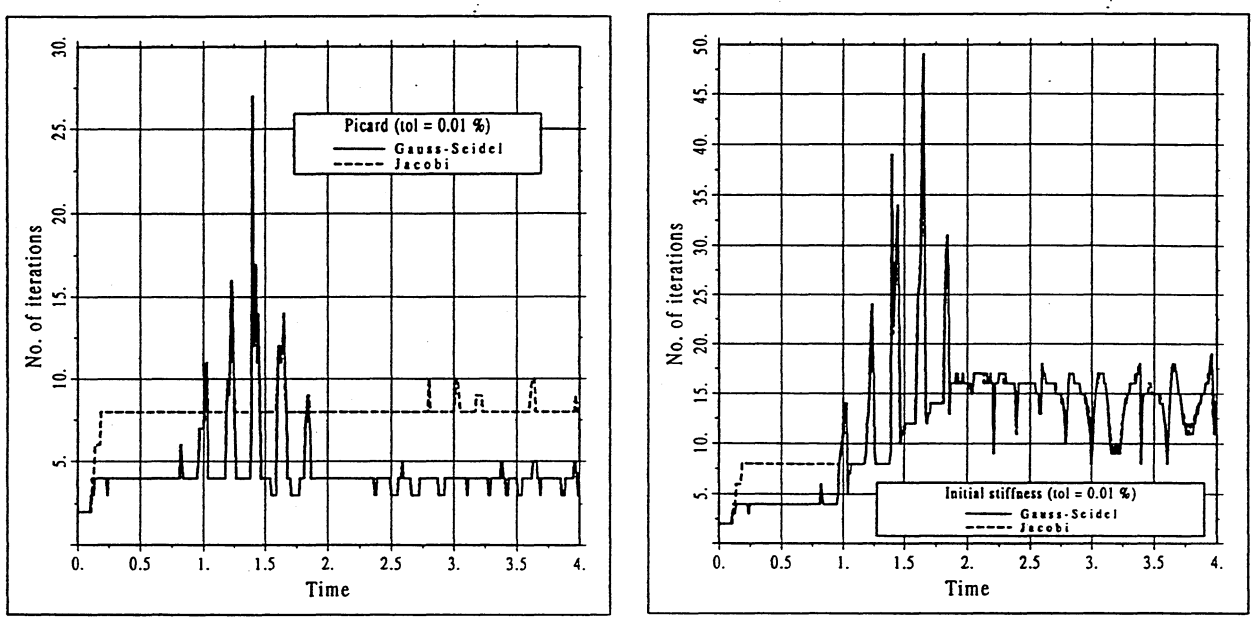


Figure 6. Number of iterations vs time using (a) the Picard scheme, (b) the initial stiffness method

number of iterations than the Gauss-Seidel scheme, as expected.

The Picard method necessitates the formation of the stiffness matrix for the solid at each iteration of each time step. One can also try to use always the initial stiffness as an iteration matrix. We have also considered this possibility, *also including the new iterations within the coupling/nonlinearity loop.* The number of iterations vs time in this case is shown in Fig. 6(b) . It is clearly observed that roughly up to $t = 1$ the coupling drives the process. Then the nonlinearity dominates and, if we compare with Fig. 6(a) , from t approximately 1.5 onwards the reason to iterate is the use of the initial stiffness. The fact that both the Jacobi and the Gauss-Seidel methods yield the same numerical response indicates that *the coupling will be achieved automatically.* The convergence history for $t = 1$ is shown in Fig. 7, from where it is observed that after the sixth iteration the nonlinearity governs the iterative process.

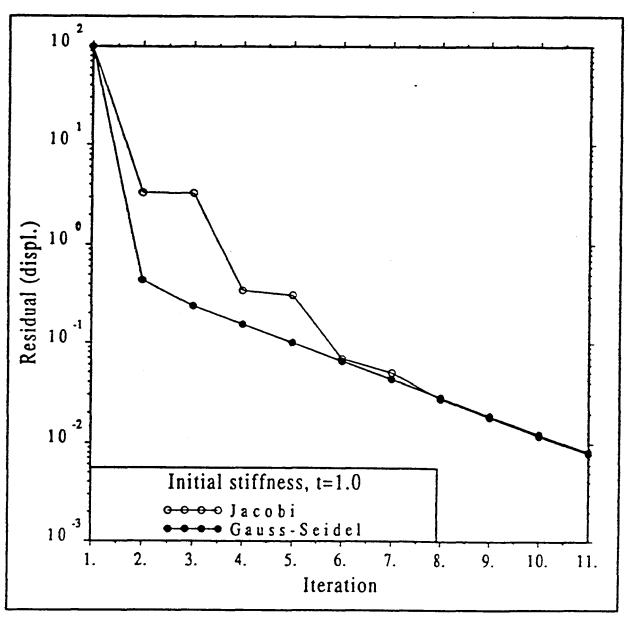


Figure 7. Convergence history for $t = 1$. Displacements

CONCLUSIONS

An algorithmic approach to coupled problems based on the partition of the semi–discrete equation of the system in its single field components has been discussed. The alternative can be seen as a block–iterative solution of the original system. If this technique is used together with a procedure for synchronized communication of the processes involved, two benefits are obtained: modularity of the necessary software and optimal use of the computer resources. Theoretical analysis of the resulting algorithm regarding stability and convergence is very problem dependent. Moreover, the values adopted for the tolerance of the overall convergence within a time step do affect time stability. However, the procedure has been successfully applied to two very different in nature coupled problems such as fluid structure interaction and thermally coupled incompressible flows. Thus, the approach is considered to be not only feasible, but highly competitive with respect to other alternatives. In the case of nonlinear problems, numerical experiments suggest that in most of the cases the rate of convergence will be driven by the linearization of the nonlinear terms, so that the coupling of the equations is achieved with very little additional cost.

REFERENCES

[1] Zienkiewicz O.Z. and Taylor R.L., 1991.
The finite element method. Vol II, McGraw Hill.

[2] Galindo, M., 1993,
Una metodología para el análisis numérico del comportamiento resistente no lineal de presas de hormigón con cargas estáticas y dinámicas, Doctoral Thesis, Technical University of Catalonia.

[3] Hughes T.J.R. and Liu W.K., 1978.
Implicit–explicit finite elements in transient analysis.
J. Appl. Mech. Engng. 45, 371–374 and 375–378.

[4] Felippa C.A. and Park K.C., 1980.
Staggered transient analysis procedures for coupled mechanical systems: Formulation.
Com. Meth. Appl. Mech. and Engng. 24, 61–111.

[5] Armero F. and Simó J.C., 1991.
A New Unconditionally Stable Fractional Step Method for Nonlinear Coupled Thermo–mechanical Problems.
To be published in Int. J. Num. Meth. Engng.

[6] Galindo M., Cervera M. and Oliver J., 1991.
Efficient solution schemes for fluid–structure–soil interaction problems.
Proceedings of 10th World Congress on Earthquake Engineering.

[7] Codina, R., Cervera M. and Galindo M., 1993.
Efficiency of block-iterative algorithms for nonlinear coupled problems.
Proceedings of the 2nd Asian-Pacific Conference on Computational Mechanics.

THE ROLE OF CONVEX WITH SOUTHWEST RESEARCH INSTITUTE AND THE UPPER ATMOSPHERE RESEARCH SATELLITE

Karen Birkelbach
Southwest Research Institute

1. Introduction

Southwest Research Institute (SwRI) is a nonprofit, independent applied engineering and physical sciences research and development organization devoted to technology development and transfer. Headquartered in San Antonio and situated on 765 acres of countryside, there are offices located in Ogden, Utah, Detroit, Michigan, Washington, D.C., and Johnston Island in the South Pacific. Approximately 2,500 employees work in more than 1.5 million square feet of laboratories, workshops, and offices.

SwRI was founded in 1947 by Thomas Baker Slick, Jr., an oilman-rancher-philanthropist who believed that science and technology are the keys to the betterment of mankind. Funding for research comes from both private and government contracts in all areas of research, including, but not limited to, applied physics; artificial intelligence; alternative energy; automation, robotics, and intelligent systems; chemistry and chemical engineering; engine research and development; emission research; fire technology; biosciences and bioengineering; data systems; engineering sciences; fluid systems and machinery dynamics; fuels and lubricants; geosciences; nondestructive evaluation; nuclear safety and waste disposal; space sciences; and offshore technology. One of the major projects awarded to SwRI is the Particle Environment Monitor (PEM) of the Upper Atmosphere Research Satellite (UARS).

2. The NASA/UARS Mission

Human activity contributes to the "greenhouse effect," causing global warming. "Greenhouse" gases consist of carbon dioxide, nitrous oxide, methane, and chlorofluorocarbons (CFCs)[1]. UARS is carrying out the first systematic, comprehensive study of the stratosphere and is furnishing important new data on the mesosphere and thermosphere. "UARS mission objectives are to provide an increased understanding of energy input into the upper atmosphere; global photochemistry of the upper atmosphere; dynamics of the upper atmosphere; the coupling among these processes; and the coupling between the upper and lower atmosphere."[2] The UARS program includes nine experimental groups and 10 theoretical groups "with specific responsibilities for data analysis and interpretation. One important product of these studies will be computer models that simulate processes in the upper atmosphere."[3] UARS operates 585 km above the Earth in an orbit inclined 57 degrees to the Equator. This orbit will permit the UARS to 'see' up to 80 degrees in latitude. With an essentially global view, UARS makes measurements every 36 days over the full range of local times at all geographical locations.[4] Over a period of 72 days, the entire diurnal cycle is sampled. Since the satellite's orbit precesses 5 degrees per day, relative to the sun, it is yawed through 180 degrees every 36 days to keep the solar panels facing the sun and preventing thermal stressing of the satellite components. [5]

1. "UARS-Upper Atmosphere Research Satellite-A Program to Study Global Ozone Change," (Space Research and Management, Inc., 1989), p. 2.

2. "UARS," p. 4.

3. "UARS," p. 4.

4. "UARS," p. 5.

5. Standard Formatted Data Unit (SFDU) Documentation for UARS requirements.

The atmosphere is made of layers of gases that have various temperatures and compositions. The troposphere rises to a height of 10-15 km and is the layer that supports life and weather on Earth. The stratosphere rises above the troposphere to 50 km, where much of the ozone heating takes place. The mesosphere rises to 80 km, where temperatures drop. The thermosphere is the top layer of the atmosphere. Temperatures in the thermosphere reach 2000 degrees C due to residual molecular oxygen absorbing highly energetic solar radiation.

Direct observation of the stratosphere and the mesosphere are technically difficult, so remote sensing from the thermosphere by UARS accomplishes this goal. Previous attempts to observe these two "middle" layers include short rocket shots and balloons, but the problems inherent in these methods return only localized data for very short periods of time.

2.1. Satellite Experiments

Chemistry, dynamics, and energy input are closely coupled in the behavior and structure of the upper atmosphere, so they must be studied together. Chemistry studies will collect data on a large number of key chemical species in the upper atmosphere; dynamics studies atmospheric winds, which distribute the gases that cause atmospheric heating; energy input studies solar radiation and magnetospheric energy at high latitudes, which "produces most of the seasonal, vertical, and latitudinal variability of the atmospheric thermal structure, which in turn controls most of the dynamics of the upper atmosphere."[6]

2.1.1. Atmospheric Composition and Temperature

Four experiments to study atmospheric composition and temperature are part of the UARS. CLAES, the Cryogenic Limb Array Etalon Spectrometer an infrared thermal emission observer which determines concentrations of nitrogens and chlorines, ozone, water vapor, methane, and carbon dioxide. CLAES uses a telescope, spectrometer, and an array of 20 detectors for taking simultaneous measurements at 20 altitudes between 10 and 60 km. ISAMS, the Improved Strastospheric and Mesospheric Sounder, also an infrared molecular emission observer, uses a filter radiometer on a reflecting telescope to measure concentrations of nitrogens, ozone, water vapor, methane and carbon dioxide. The Microwave Limb Sounder (MLS) measures chlorine monoxide, hydrogen peroxide, water vapor, and ozone in the microwave region of the spectrum. Unlike the previous two instruments, MLS uses an antenna, rather than optics, for its data gathering. HALOE, the Halogen Occultation Experiment, measures infrared absorption in the atmosphere and determines the vertical distributions of hydrofluoric and hydrochloric acids as well as methane, ozone, carbon dioxide, water vapor, and members of the nitrogen family.[7]

2.1.2. Atmospheric Winds

Two experiments study atmospheric winds in the upper atmosphere. The High Resolution Doppler Imager (HRDI) and the Wind Imaging Interferometer (WINDII) provide direct observation of wind velocity through measurements of the Doppler shifts of selected emission and absorption lines. HRDI studies the Doppler shift of spectral lines of molecular oxygen below 45 km altitude. Above 60 km, it observes emission lines of neutral and ionized atomic oxygen. WINDII provides wind measurements above 80 km using roughly the same techniques as HRDI. Energy input experiments observe ultraviolet radiation and mag-

6. "UARS," p. 9.

7. "UARS," p. 18-19.

netospheric charged particles from the Sun.[8]

2.1.3. Energy Inputs

Three experiments on UARS provide these observations. The Solar Ultraviolet Spectral Irradiance Monitor (SUSIM) is located on the portion of the satellite facing the Sun and stars to measure solar ultraviolet radiation in the range 120 to 400 nm with a resolution of 0.1 nm. On the same side as SUSIM, the Solar/Stellar Irradiance Comparison Experiment (SOLSTICE) measures solar ultraviolet radiation in the range 115 to 430 nm with a 0.12 nm resolution. SOLSTICE compares the solar ultraviolet output with the ultraviolet radiation of bright blue stars, using the same optics. PEM, the Particle Environment Monitor, determines the type, amount, energy, and distribution of charged particles injected into the Earth's atmosphere and will be discussed in further detail elsewhere.[9]

2.2. Data Flow

In past satellite projects, data has been transmitted from the satellite to ground stations which forwarded the raw telemetry to Goddard Space Flight Center (GSFC), who would then process data into corrected formats and send the corrected data to the principle investigators, usually in large volumes of 9-track reels. With UARS, a new scheme for distributing data is in place where raw telemetry is still received at GSFC, but individual investigators do first level processing at the Central Data Handling Facility (CDHF) and those files are then transferred to various Remote Analysis Computers (RACs) where further data processing takes place.

2.2.1. Telemetry To Ground

Data are collected into frames and recorded from the many experiments onto tape recorders that play back their contents at specific points during the satellite's orbit to the TDRS satellite, which are then transmitted to a tracking station at White Sands, New Mexico. From White Sands, the data are sent to the Data Collection Facility (DCF) at GSFC via land-line, where they are then decommutated into "Level 0" files. Level 0 files are genereated for each experiment as well as for engineering and data quality. The engineering files are used by satellite and experiment operations personnel to track the health and operation of the satellite.[10]

2.2.2. Level 0 to CDHF

Once the data are decommutated, the Level 0 files are sent via electronic network to the Central Data Handling Facility (CDHF) at GSFC. At this point, programs written by UARS programmers and scientists are run to generate files containing other various levels of detail and data transformation. Some experiments have several levels of processing that take the telemetry from the Level 0 files and convert them into sets of files, that at different levels, acquire such properties as containing the physical parameters measured by the sensors, geophysical parameters derived from the previous level of data, and finally various forms of averaging the geophysical data. PEM has only two levels of processing: Level 2, which

8. "UARS," pp. 20-21.

9. "UARS," pp. 22-23.

10. Private Communications with Sandra J. Jeffers, SwRI software engineer for all CDHF code, September 1993.

contains the physical measurements from the PEM instruments and some geophysical parameters (latitude, longitude, altitude, etc. as a function of time), and Level 3at, which averages the data in the Level 2 files across one minute intervals. These are the files that are sent to the Remote Analysis Computers (RAC).[11]

2.2.3. CDHF to RACs

Files are transferred from the CDHF to the RACs using DECnet COPY. Once the files reach the RAC, scientists can use programs developed at the RAC to analyze the data, typically within two to three days of real time. At the PEM RAC, Level 0 data are used to generate Health and Safety plots for the PEM instrument and are current to within one day.

3. SwRI's Role in UARS

The role of SwRI with the UARS mission has been two-fold. SwRI's Dr. David Winningham is the Principal Investigator for the Particle Environment Monitor experiment and in that role directed the development of the experiment program and the building of the instrumentation itself. The second major task is data analysis and interpretation of the data received from the UARS.

3.1. PEM

SwRI, as primary sub-contractor, along with Lockheed Palo Alto Research Laboratory, NCAR, the University of Alaska, and Johns Hopkins University Applied Physics Research Laboratory, designed and built the Particle Environment Monitor, a set of experiments attached to the UARS vehicle to study the type, amount, energy, and distribution of charged particles into the Earth's thermosphere, mesosphere, and stratosphere.

The experiment consists of four instruments: Atmospheric X-ray Imaging Spectrometer (AXIS), High Energy Particle Spectrometer (HEPS), Medium Energy Particle Spectrometer (MEPS), Vector Magnetometer (VMAG).

MEPS sensors measure electron and proton energies in the range of 1 eV (electron Volt) to 5 MeV, while the HEPS sensors measure particles in the 1 eV to 150 MeV range. VMAG measures the strength of the Earth's magnetic field to provide a reference for the plasma measurements and determine energy deposited into the ionosphere by "field aligned currents." [12] The AXIS instrument is an array of 16 X-ray detectors that view the earth from limb to limb. The PEM experiment will provide X-ray images of energy in the range of 3 to 100 keV, allowing scientists to reconstruct the three-dimensional energy input spectrum of electrons up to 1 MeV on a global basis.

The four instruments are in different locations across the spacecraft. HEPS and MEPS are located on booms that respectively point directly away from and directly at the earth. They determine particle distribution at the spacecraft. VMAG is located on the same boom as HEPS and measures the local vector magnetic field. AXIS measures x-ray energy from the atmosphere across the globe and is located on the earth side of the body of the spacecraft.

11. P.F. Bythrow, S.J. Jeffers, R. L. Murphy, R.W. Nightingale, "Particle Environment Monitor Software Specifications, Data Descriptions and Algorithms," 1989, p. 4.

12. "SFDU"

3.2. RACs

There are nine RACs associated with the UARS. When the computer systems and networks were being designed, nearly all of the RAC sites decided to purchase DEC VAXes because the system at the CDHF was a VAX cluster. SwRI examined the capabilities of both the VMS and the UNIX operating systems and decided that for their purposes, a UNIX system was more appropriate. Purchase decisions were based on the central computer idea because workstations were still in their infancy, being very expensive for the number of users they served. Since launch, most, if not all, of the RACs have migrated to a more distributed system of computing using networks of workstations, with UNIX workstations becoming more popular among the VMS-based RACs.

SwRI is home to the Particle Environment Monitor RAC, the Convex C120 and associated network. The Convex is named pemrac.space.swri.edu. Each RAC has at least two connections to CDHF either through Internet and/or leased-line. SwRI's primary connection to CDHF is through Internet with a secondary line directly from CDHF to PEM, a MicroVAX II which provides DECnet to the RAC. Pemrac is not directly involved in data processing on the UARS project. That task is handled by Sun workstations. Data storage is on magneto-optical media in a 56 GB jukebox connected to a Sun Microsystems workstation. Pemrac is currently used for code development for CDHF and it is the central repository for all UARS code.

3.3. SDDAS

The Southwest Data Display and Analysis System (SDDAS) is being developed as a multidisciplinary data analysis tool. SDDAS allows a user to generate correlative data from multiple missions and display it in a number of ways. Based on the Instrument Definition File Set (IDFS), any data in the IDFS format can be read and graphed with SDDAS.

4. SwRI's Selection of Convex

Thanks to Convex's implementation of the VAX extensions to FORTRAN, research for CDHF development code occurs on Pemrac, as well as actual code development, which is then transferred to the VAX cluster at CDHF for testing and production runs. RCS is used for software configuration management. The Convex is also used for CDHF algorithm validation in which CDHF FORTRAN code is transferred from CDHF to the Convex, VMS system and UARS Catalog Services System (UCSS) calls are stubbed, and the code is compiled and run at SwRI. The resulting data are plotted to verify that the data processing algorithms in place at CDHF are indeed correct.

4.1. Need for Supercomputing

The original need for supercomputing for UARS/PEM was to model the instruments to be used in space before they were built. Modifications to the design could be implemented more easily than changes to the hardware. The first codes to run would be ray-tracing programs to help engineers make decisions such as what size holes and how many holes there should be in the grid screen for the sensors. Future plans include running Thermal Global Chemistry Models, developed at the University of Alaska, using data collected from the satellite.[13]

4.2. Choices Available

13. Private Communication with Dr. J. David Winningham, June, 1993.

Based on the large-code requirements, the original project team began looking at several leading super/minisuper computer companies, including Alliant, SCS, and Convex. Cray was ruled out because of the cost of hardware and facilities modifications.

4.3. Selection Criteria

Each of the companies under consideration were given a benchmark version of the ray-tracing code to run on the machines in question. Although performance was an important factor, it was not the only one: the quality of the software, including operating system and compilers; quality of the hardware, less of a factor since it probably would not be changing, making the software issue more important; and finally support from the company.[14]

4.4. Final Selection

The decision to purchase from Convex was based on all these factors. Based on the results from our benchmark tests among Convex, Alliant, and SCS, Convex was the winner hands down. Convex OS has been a very good operating system, with far fewer bugs than many of the other platforms in use at SwRI, the compilers have been very reliable and their promise of excellent support has held true. Convex as a company and the machine itself have both performed to our expectations.

The choices, in 1987, were narrowed down to Alliant, SCS, and Convex. Alliant and Convex seemed to be the better choices because SCS was having difficulties at the time and there existed a strong possibility that they would not be available in the future to support their computer. Benchmark codes were delivered to Alliant and Convex with the agreement that they be run at different levels of optimization with no changes to the source code. The code used for the benchmark was a non-vectorized ray-tracing program written originally for a CDC Cyber 182.

The results came back favoring Alliant in terms of speed, but it was discovered that the source code had been modified by an Alliant software engineer to speed it up. They basically violated the terms of the benchmark test. With the cost and the future factored in, Convex appeared to be the winner. Based on history, it can easily be said that Convex was the best choice since they are still supporting us after four years.

4.5. Other Uses of Pemrac

During the development of the UARS/PEM software, the Space Shuttle Challenger accident occurred, putting all NASA shuttle missions on hold for an indefinite period of time. During this "down" time, the decision was made to allow non-UARS users access to the Convex. The Institute was in a state of flux at the time, also, with the Central Computing Facility ordered to start charging for computer time to recover costs on their VAX 8700. For scientists whose projects were won without computer time costs factored into them, having the Convex available was important since the only other choice for "free" computer time was a VAX 8530. The Convex has been used for comet coma modelling. The code used modelled the atmospheres of comets which included fluid dynamics of the gases flowing around the comet body, complete chemical kinetics, and plasma physics.[15] The models used actual data from the Giotto mission to Comet Halley as parameter constraints. Interestingly, the code used for this modelling found several bugs in the FORTRAN compiler, which Convex later advised us were fixed in the next release.

14. Private Communication with Richard Murphy, April, 1993.

15. Private Communication with Dr. Daniel Boice, August, 1993.

5. Conclusion

The end result of the use of the Convex computer at SwRI is that during times of trouble with the satellite or with the PEM experiment, the careful development work that took place before flight allows the scientists to make decisions on the operation and modes of the experiment based on correct data. Convex has provided superior support in terms of both hardware and software, which has helped to make our jobs much easier.

6. Bibliography

Boice, Daniel, PhD, Private Communication, August, 1993.

Bythrow, P.F., S.J. Jeffers, R. L. Murphy, R.W. Nightingale, "Particle Environment Monitor Software Specifications, Data Descriptions and Algorithms," UARS project documentation, 1989.

Jeffers, Sandra J. SwRI software engineer, Private Communication, September 1993.

Murphy, Richard, SwRI Manager, Private Communication, April, 1993.

Standard Formatted Data Unit (SFDU) Documentation, UARS project documentation.

"UARS-Upper Atmosphere Research Satellite-A Program to Study Global Ozone Change," Space Research and Management, Inc., 1989.

Winningham, J. David, PhD, Private Communication, June, 1993.

European Mass Storage Activities

Chris Hardy
Manager European Data Management Products
European Convex Users Conference
27-29 October
Bilbao, Spain

The success of Convex in the Data Management field up to the end of 1992 surpassed all expectations. It was remarkable in a number of ways but in two most importantly. Firstly, 25+ hierarchical storage management (HSM) systems were sold world-wide which proved there was a LARGE data storage problem in the field. Secondly, the majority of those systems are installed in Europe.

There are two HSM solutions currently available from Convex. UniTree is a package licensed by OpenVision to multiple vendors providing an "open-systems" solution to file migration. FileServ is a package licensed by E-Systems designed to address some of the largest data storage requirements; up to 10000 Terabytes. Both of these solutions address the problem of migrating data from expensive disk to cheaper tape media and staging the data back, when needed, transparently to the user. Architecturally, there are important differences between the two products but that is outside the scope of this presentation.

The Convex Data Management success had been achieved with the aid of the File Server Integration Group (FSIG) based in Dallas. However, this set up was no longer capable of supporting the number of systems and solutions installed. Demands on limited resources were pulling the group in many different directions and many of these were mutually exclusive! There had to be a fundamental change in the way Convex was addressing the Data Management business in order to maintain the momentum.

The new organizational structure for Data Management reflects the importance Convex is placing on this business. The FSIG development staff have been absorbed into the engineering group and together with a number of additional staff have strengthened their team. The FSIG support staff have moved into the Technical Assistance Centre (TAC) support group and, again, have a singular focus on Data Management. Additional resources have strengthened this group too. Marketing resources have been allocated to focus solely on this market. Finally, and very significantly, Tom Jones is now the Vice-President of Data Management to manage the direction and development of this important sector of Convex's business.

Convex Europe also took a look at how it was managing the Data Management business. At the beginning of 1993 the European Mass Storage Engineering (EMSE) group was established. The group consists of three

members; Ray Davis, Marty Propper and myself. I am the manager of this group as well as being responsible for providing European sales support. Ray Davis works on special developments, integration and test of Data Management systems. Marty Propper, recently relocated from our US organization, specializes in support for Data Management products. EMSE personnel are a European wide resource and are not tied to any particular country. Ray Davis and Marty Propper are located in Frankfurt together with our support facilities. I work out of the European Headquarters in London.

I also fulfil a marketing role through discussions with prospects and customers across the face of Europe. I feed this information to the engineering, marketing and support teams in Dallas to represent the needs of the European market. This is important because there are some differences between the European, US and Pacific Rim markets. Europe has a greater number of medium size installations, the US has fewer larger installations while the Pacific Rim are requesting support for different media types and robotics

There have been some contractual events in the last year that have had an impact that is worth noting. Essentially, they allow Convex to concentrate on user solutions and developing the business now that the negotiations are complete.

1. A joint Marketing, Sales and Support agreement was signed between Convex and E-Systems for the FileServ and VolServ products. This agreement allows Convex to go after the high-end business and because of its larger sales force than E-Systems and its knowledge of the marketplace has already taken a number of sizeable orders.

2. Convex signed a contract with OpenVision for UniTree v1.7 and all future releases on Convex. Titan no longer forms part of the relationship. The closer relationship enables a better flow of information, e.g. bug fixes in one direction and enhancement requests in the other. Convex and OpenVision are exchanging test technology to ensure that a better product gets released at source.

3. Convex signed a contract with OpenVision to license the UniTree product on HP 9000/700 series workstations. This provides an entry-level system for customers starting out with HSM solutions. An upgrade path to a Convex system is possible by moving the UniTree application and the robotics to the Convex once the workstation runs out of steam.

There were a number of open issues affecting the Convex Data Management market that needed to be resolved in 1993. One of these problems was to identify a replacement for the product RSB. After evaluating a number of network backup products for functionality, availability and ability to form a good working relationship with the developers the product SM-arch from

Software Moguls was chosen. It is currently in Beta test at a number of sites on Convex and will be generally available later this quarter.

Some of the main features are:

- It has a GUI to set up and administer the product
- It works seamlessly with either UniTree or FileServ but is also able to use the tape interface directly
- Users are able to retrieve their own files across the network without operator intervention
- Backups can be scheduled centrally or locally depending on set-up
- The software can backup "Unix", VMS, PC and Macintosh systems

Another problem to address in 1993 was a direct result of our success up to, and including, 1992. UniTree version v1.5 was installed and put into production at a number of sites. However, v1.7 with its enhanced functionality was expected shortly. To upgrade from v1.5 to v1.7 one of the databases needed to be completely re-worked and it was an irreversible process. Once a site is in production any loss of data is catastrophic. So, Convex had to write some tools to make this a fail-safe process which took a little longer than anticipated. The second problem was with v1.7 itself. Many of the bug fixes made to v1.5 did not make their way into the v1.7 release. Before we could consider moving v1.5 customers to v1.7 all the previous fixes had to be re-implemented. We are confident this will not happen again! As a result of these two issues some customers will have witnessed delays to the v1.7 release. These are as frustrating to you as to us but Convex and OpenVision feel confident that as we move forward together these regression problems will be greatly reduced.

Support is extremely critical to Data Management sites because a file server that is down has a serious knock on effect to the machines it is serving data to. UniTree and FileServ training have been conducted in Europe for the local Convex technical support staff over the past few months. EMSE provides an additional level of European support to the field offices with the TAC beyond that. Training for new customers has taken place on site during installation and a programme of customer training will be announced shortly which is aimed at complementing the initial training.

I have selected three European installations to highlight what is possible with Convex Data Management solutions.

1. University of Odense

Ray Davis has just completed a special development that has been installed during October. It consists of UniTree running on the Convex with HP Optical disk support as it's backing media. Additionally, UniTree running on each of two HP workstations provides a disk cache for the users while utilizing the tape complex of UniTree running on the Convex. Thus, the

workstation data is migrated and staged across the network to the Convex. This is one of the, if not THE, first distributed versions of UniTree running today. The advantage of this solution is that only one robot need be purchased to support, effectively, three UniTree installations.

2. DKRZ

The German Climatic Research Centre in Hamburg, Germany, has recently upgraded from UniTree v1.5 to v1.7 and is one of the first sites to have two different media types for its backing media; 3480 and VHS. The 600 tape metrum robot (8TB) is used to store large files while the STK Silo (1.2TB) uses the 3480 tapes to store smaller files. This is achieved by using the family of files facility available with v1.7. An enhancement is planned to allow the automatic migration of files to different media based on file size, UID, GID, etc. Another future enhancement will be the automatic support of multiple layers in the hierarchy, e.g. data moves from magnetic disk to optical disk to 3480 to VHS to D2 to Optical tape and back to magnetic disk when needed.

3. ULCC

The existing FileServ system with 3480 robotics support has recently been successfully upgraded to 2.1.9. It provides archiving facilities for the users of the C3860 Super Computing facility.

A second system is a joint project between Convex and ULCC. A Convex running FileServ is controlling a DataTower (6TB of storage) and two ER90 drives. ULCC plans to provide a national data repository featuring fast access and retrieval. Development of Meta-Data applications is a major part of the project. This facility will also provide a demonstration and test facility for anyone interested in this technology. Please contact your local Convex representative or myself to co-ordinate this activity. ULCC do not want to be swamped with requests!

A recent Data Management FOCUS (Focus On Customer User Solutions) meeting recently conducted with 10 luminaries in this field suggested some areas where Convex should be spending its resources.

The solutions on offer need to be quality products that are reliable and robust. This is being addressed by the Convex support and engineering groups and great progress has been made. Even better quality products will follow.

The solutions must have the functionality the user needs to use and also manage the systems. User input in this area has been taken and, together with OpenVision, our engineering group is addressing these issues.

Performance is expected of the Convex software but this is not as high a priority as the preceding points. A performance release of UniTree, however, is planned for Q1 of 1994.

Finally, the FOCUS group were extremely interested in a distributed file system together with migration capabilities. DFS with its global name space and its performance advantages over NFS looks to be the leading solution. DFS needs to be generally available in order for it to become a de-facto standard. In the interim, AFS can provide a solution to this problem and as a forerunner to DFS will be field upgradable to DFS. Convex will have both the AFS server and client software available in Q4 1993. It is currently running at a number of Beta sites. The next step will be to add migration capabilities based on either UniTree or FileServ that will not require any changes to the AFS clients. This is expected to be available by the end of Q2 1994.

The success of Convex in the Data Management field has continued into 1993 with the addition, this year, of 20+ HSM systems world-wide. They are providing backup, archive and file server solutions for both technical and commercial customers. With the availability of UniTree on the HP workstations more commercial customers can be expected. Also, with the increasing installed base of FileServ customers Convex expects an exciting future with its Data Management solutions.

SUPERCOMPUTING AID IN NEURAL NETWORK RESEARCH

Santiago Rementeria, Josu Oyanguren and Gurutze Marijuán
LABEIN Research Centre

1 Introduction

Neural network computing is becoming an important technology as it gradually shows its effectiveness in application areas ranging from signal processing and pattern recognition to combinatorial optimization and cognitive or biological modelling. The distributed and parallel nature of artificial neural networks offers the promise of a high degree of robustness and fault tolerance currently lacking in many sequential computing applications. Although neural network research dates back to some decades ago, it has been in the last seven years that it has emerged as a hot topic. New theoretical results and practical developments have led many researchers and institutions to devote considerable effort to the study and application of this alternative to conventional computing.

On the other hand, vector and parallel supercomputers offer the storage size and speed required to get maximum performance in numeric applications like neural network training and operation. Supercomputing lends itself naturally to neural network research because most of the operations involved in the simulation of such devices are vector and matrix-oriented. Such powerful machines can exploit the fine-grained parallelism inherent in neural architectures speeding up their computationally costly training process.

This paper tries to give an overview of the way supercomputers can help in the development of computationally intensive simulations of artificial neural systems. In the following section we will introduce neural networks and two learning algorithms for two different types of neural architectures. After stressing the suitability of supercomputers for the simulation of this type of information processors (section 3), we will describe the experiments carried out with the objective of building a prototype system for electrical utilities to forecast hourly and daily electricity consumption. Several results and a first assessment of the performance achieved using a CONVEX computer are also given in section 4. Finally, section 5 concludes the paper by bringing out the most important results and ideas.

2 Neural networks basics

Artificial neural networks are computing systems made of many simple computing units (also called nodes or *neurons*) interconnected by links [1,2]. Such elementary processing elements receive inputs from a number of other units or from the external world. An activation (or transfer) function, which is usually nonlinear, is then applied to the

weighted sum of the inputs. The resulting output value can be distributed through weighted connections to other artificial neurons. The way in which neurons are interconnected defines the flow of information within the network, and is called the architecture of the neural network. No central memory in the conventional sense is required, as that concept is replaced by weight setting. Interconnection weights are adjusted in the training phase through an algorithm called learning rule. This allows artificial neural systems not to be explicitly programmed, but rather being taught by examples.

In our research we tested a *feedforward* network and a fully *recurrent* one. In feedforward networks the set of units can be partitioned into layers, with links from each unit in the k-th layer being directed to units in the (k+1)-st layer. Layers located between the global inputs and outputs are commonly termed *hidden*. Feedback or recurrent networks, on the other hand, contain inter and intralayer feedback loops. Each unit in a fully recurrent network is connected with all the other units -including itself-, and so the network loses its layer structure. Feedback connections allow the units in a recurrent network to receive information about past states of the system. Figure 1 shows a three-layer feedforward network with six input and two output units and a network with three fully recurrent units and one output.

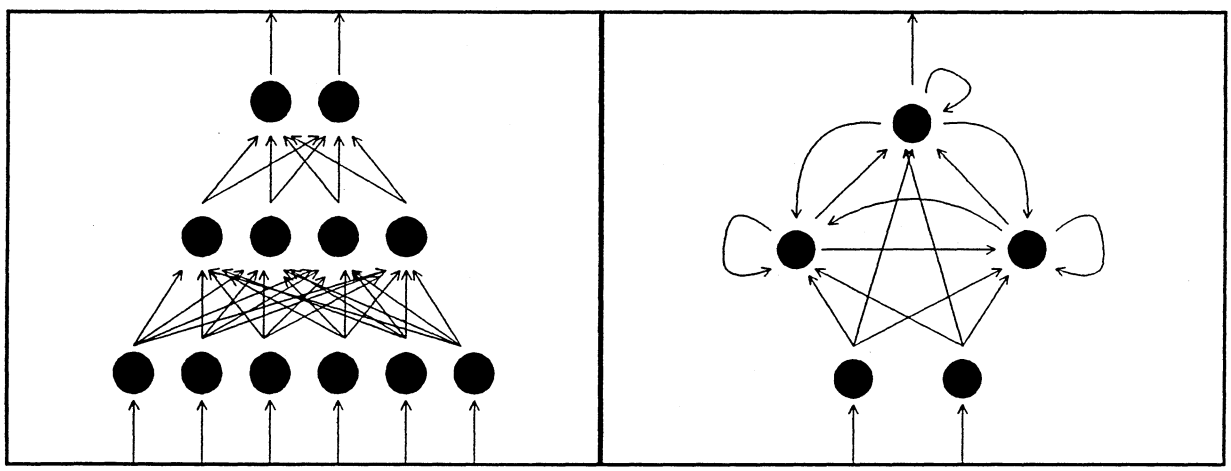


Figure 1. Examples of a feedforward and a recurrent network

The learning methods employed were the *Backpropagation (BP)* [3] and *Real Time Recurrent Learning (RTRL)* [4] algorithms, respectively. Backpropagation, also known as the generalized delta rule, is the most widely used learning algorithm, and it is intended for multilayer feedforward networks. It is basically a gradient descent algorithm. The most common activation function for this type of network is the sigmoid function:

$$x_j = \frac{1}{1+e^{-\sum_i w_{ij} x_i}}$$

where w_{ij} is the weight of the link from unit i to unit j and x_i is the activation of unit i.

The following is a common version of backpropagation:

1.- Initialize link weights randomly.
2.- Clamp the input pattern to input layer units.
3.- Compute the activation function in all units from the input layer to the output layer.
4.- Compute the error, δ_j, for output units:

$$\delta_j = x_j\,(1-x_j)\,(d_j-x_j)$$

where d_j is the target of output j.
5.- Propagate δ_j backwards.

$$\delta_j = x_j\,(1-x_j) \sum_{i \in \textit{output layer}} \delta_i\, w_{ji}$$

6.- Update link weights.

$$\Delta w_{ij}(t) = \alpha \delta_j x_i + \beta \Delta w_{ij}(t-1)$$

where α and β are two learning coefficients.

The RTRL algorithm is an extension of BP to train fully recurrent, continually running neural networks. It is suited to model dynamical systems described by the temporal sequence of their input-output pairs. The activation function is also here the sigmoid function. The RTRL algorithm can be summarized as follows:

1.- Initialize link weights and activation nodes randomly.
2.- Clamp the input pattern to the set of input nodes.
3.- Compute the activation function until the result becomes stable.
4.- Compute the error, e_k, for output nodes.

$$e_k(t) = \begin{cases} d_k(t) - x_k(t) & \textit{if } k \in T \\ 0 & \textit{otherwise} \end{cases}$$

where T is the set of output nodes
5.- Update the link weights.

$$\Delta w_{ij}(t) = \alpha \sum_{k \in U} e_k(t)\, p_{ij}^k(t)$$

$$p_{ij}^k(t) = \frac{\delta x_k(t)}{\delta w_{ij}}$$

being U the set of all nodes
6.- Adjust p^k_{ij}:

$$p_{ij}^k(t+1) = x_k(t) \sum_{l \in U} w_{kl}(t)\, p_{ij}^l(t) + \delta_{ik}\, x_j(t)$$

Both BP and RTRL are very powerful algorithms whose main drawback is that they require a great deal of computation on each update cycle.

3 Simulation on supercomputers

The ideal way to test the properties of artificial neural systems is through the construction of hardware emulations, but it is still costly to build electronic (VLSI) or optical neural devices. In most situations researchers resort to simulations running on conventional computers to study and test different network architectures and to assess the efficiency of algorithms before they are implemented as a piece of hardware. The computing power required for such simulations can be accomplished in different ways:

- using specialized hardware systems like bus-oriented processors and coprocessors attached to personal computers or workstations to simulate relatively small neural networks [5].

- using mainframes, supercomputers and massively parallel computers. MIMD [6] and SIMD [7,8] computers provide the power needed for simulations of large neural networks, but they are not so widely available as workstations and personal computers. In [9] it was concluded that the supercomputer category is the fastest operating implementation option for artificial neural networks.

Artificial neural networks are well suited for simulations on supercomputers. By the term *supercomputer* we will refer to a class of computers supporting not only large and high-speed memories and input/output operations, but mostly some degree of parallelism understood as the capability to simultaneously process tasks ranging from jobs and programs to loops or statements. Thus, some combination of multiprocessing, vector processing and concurrent scalar processing will be expected from supercomputers.

The relevance of parallel computing stems from the fact that neural architectures and learning algorithms are also parallel in nature. We have seen in the previous section that layers of neurons and weights can be regarded as the building blocks of many neural network models. Each neural unit is assumed to process information independently. At the implementation level collections of neurons could be associated with individual computer processors. Alternatively, neuron arrays can be represented as vectors and the connections between these arrays of units (i.e. weights) can be represented as matrices. A second degree of parallelism refers to the processing of training examples. Several steps of algorithms like backpropagation and RTRL can be vectorized or parallelized, thus bringing a considerable reduction in computation time. Weight matrix and unit activation updating, for example, are inherently parallel procedures. Individual contributions made by each training pattern to the adjustment of the network weights can be linearly combined so, in principle, patterns could be processed independently and concurrently (*batch* updating of weights). Moreover, most supercomputers have resources like floating point processing or fast multiply-accumulate operations (a majority of neural computations is of the type multiply-and-add) that provide additional support for efficient neural network simulations.

The processing, memory and input/output speed provided by supercomputers can be enormously beneficial for neural network *design*. The *training* times required by algorithms like BP or RTRL can result too long for them to be practical in complex applications on conventional computers. Huge databases have to be processed in the

training phase involving lots of time-consuming vector and matrix operations. Neural network building is a practically *ad hoc* procedure for which some heuristics are known but no scientifically sound guidelines exist yet. Deciding on the optimal number of hidden units or tuning the algorithm parameters are typical decisions that are based on an extensive experimentation. Supercomputers allow to broaden this experimentation stage in order to get a (near-)optimal neural network design much faster. The *performance* phase requirements are usually less severe so, after having been trained off-line on a supercomputer, the network may well be operated on a smaller computer, typically a workstation.

Regarding software, there are three options for neural network modelling. The first, obvious one is to implement neural nets using high-level computer languages. Although they are not the most widely used choice yet, object oriented languages seem conceptually appropriate for neural network design. The second option lies in using mathematical simulation packages. This is less effort demanding but also less flexible than the previous option. Finally, there exist neural network development tools that ease the development of standard and relatively small networks providing adequate development interfaces. Unfortunately, they are not that practical when designing novel architectures or implementing large networks.

Most software simulations of neural networks on supercomputers are implemented in high-level languages like C, so optimizing compilers can be used to make the source code suitable for vector or parallel processing.

4 Case study in an industrial application

We chose electric energy consumption forecasting as a practical problem to test the potential of neural networks. Predicting tomorrow's load as accurately as possible is a task performed daily by engineers in power generating and distributing companies worldwide. Such short-term forecasts play an important role for a reliable, safe and economic operation of power systems. However, electricity consumption is highly unpredictable as it is influenced by factors like the hour of the day, the weather and random disturbances whose effect on the load is uncertain. Artificial neural networks are an alternative to conventional statistical techniques like regression, exponential smoothing or time-series modelling that are normally used in this task [10].

Before starting the experimentation stage the set of available load data was divided in two sets. The *training set* consisted of hourly load and temperature information between 1987 and 1990, whilst the hourly consumptions during 1991 constituted the *test set*. In our experiments we used a CONVEX C-3820 computer with two processors providing a computational rate of 200 Mips and 250 Mflops and with 256 Mbytes of crossbar memory with three access channels (500 Mbytes/s). The software was written in C language and a CONVEX C compiler was used.

Two different series of experiments were carried out. Experimentation began assuming a one hour ahead prediction problem. This means trying to predict the consumption of the following hour having hourly information of past consumption values. Additional information like past temperatures or the day of the week was also used in later

experiments. Although this task is not too realistic -the most useful short-term forecasts are the ones done the day before-, it served as a first step in getting acknowledged with the application. The predictions done with a lead time of one hour may constitute a best-case situation useful to set the performance limits of neural nets for more realistic time horizons. The second series of experiments assumed a lead time of 48 hours, which can be considered the most unfavourable situation in one day ahead forecasting.

For the one hour ahead problem we compared the performance of multilayer networks trained through BP and fully recurrent networks trained using the RTRL algorithm. After extensive experimentation it was concluded that the latter type behaves better in this kind of applications. Hence, for 48 hours ahead forecasting only fully recurrent networks were considered.

Table I shows the performance achieved with a fully recurrent neural network and different input vectors.

Table I. Neural network performance
(correctness on the test set)

Forecasting horizon	Input: L_{t-1}, L_{t-2}	Input: L_i(ARIMA)	Input: L_i(ARIMA), T_t(forecast), Q_i(qualitative)
1 hour	96.5%	98%	98.2%
48 hours	--	91%	94%

Using only information relative to previous loads it was observed that knowing the load values of the preceding two hours suffices to achieve a mean squared error (MSE) of 3.5% in the test set. Being this a pure time-series problem, ARIMA modelling can be used to reveal the features which are statistically more relevant [11]. Taking such features as input of the neural network the MSE decreased to around 2%.

It was observed that, for the one hour ahead problem, adding qualitative information like the month, day and time of the prediction and whether the current day is a holiday or not, and quantitative one like the temperature forecast do not contribute much to decreasing the MSE (only about 0.2%).

The situation is a bit more complex in the 48 hour ahead forecasting problem. Having a wider prediction horizon increases the uncertainty and so the results are necessarily worse than in the previous problem. A statistical method was used to determine the most relevant input features. The recurrent network trained with those features as inputs attained a 91% of correctness in the test set. In contrast to what was observed in the one hour ahead problem, additional information is quite helpful in this case and the MSE is reduced in about 3%. This is because the information contained in the load values of 48 hours before the forecast is much less than the one implicit in the load value of the immediately preceding hours. Thus the importance of additional information in realistic load

forecasting problems.

Another performance measure for assessing the potential advantages provided by supercomputing in neural network design is the training phase speed-up. Although having the C code running on the CONVEX was almost straightforward, small optimization efforts did not yield as good results as expected. Compiler optimization does not eliminate the burden of carefully having to write the program with vectorization and parallelization in mind, though the information generated at compile time has been found useful to detect weak points in the code. The average improvement obtained over a Sun SPARC II workstation amounts to a factor of 6. We are currently recoding the learning algorithms to better exploit vector processing and enhance this provisional result [12,13,14]. In order not to waste the potential of supercomputers, the level of effort involved in this task should be justifiable. We believe it is, since the potential pay-back in terms of speed is considerable and most changes made to programs will be portable to other neural network applications. Moreover, it is expected that future high-performance computers with vector and parallel processing capabilities will be more widely available, so any improvement done in this sense now will be fully exploited soon.

5 Conclusions

In view of the experience gained and the preliminary results obtained, there are several conclusions that can be drawn from the current line of investigation. In what concerns neural networks, they have proven to be a robust black-box approach in applications like the one considered here. The architectures implemented allow the user not to worry about any explicit modelling of the data. Such models are learned by the system starting from a sequence of recorded cases. The results obtained in the load forecasting problem are very satisfactory, being apparent that the performance depends on the particular type of network chosen and the concrete application involved.

Given the sensitivity of the network performance to the inputs of the system, we have confirmed that traditional statistical methods can alleviate considerably the task of deciding what particular features are more relevant. Although this observation may seem trivial, many neural network developers still insist on following "common sense" or blind trial-and-error approaches to select the network inputs. Unfortunately, trial-and-error cannot be completely banished from neural network building, and lacking a methodology to determine the optimal architecture of the network is arguably the major drawback of this technology.

Although operation speed is an unarguable asset of neural networks, their training slowness is a clear inconvenience. And it is here where supercomputing is seen as a valuable tool. Sufficiently complex artificial neural networks could be trained on supercomputers and then ported to other machines in order to be integrated with the computing environment of the application. High-speed processing has beneficial effects on neural network design and makes working methodology more flexible. Fast training times improve productivity and contribute to more reliable solutions. Some researchers have reported a very favourable performance of neural networks implemented on supercomputers. Our experience in this respect, however, prevents us from being conclusive until more experiments with more refined versions of the code have been

carried out. The speed gains obtained up to this writing are not negligible but should be amenable of improvement. Exploiting vector and parallel processing capabilities of supers often requires adapting learning algorithm implementations. We are currently working on code optimization as, from what we have experienced, compilers are not the panacea.

Acknowledgements

The work reported in this paper started as an internal research effort and is now part of an ongoing project supported by a grant from the Department of Industry of the Basque Government. We are grateful to Félix Alonso from Iberdrola S.A. for having provided us with data for the experiments.

References

[1] Knight, K., *Connectionist ideas and algorithms*, Communications of the ACM, vol. 33, no. 11, November 1990, pp. 58-74.

[2] Lippmann, R.P., *An introduction to computing with neural nets*, IEEE ASSP Magazine, vol. 4, no. 2, April 1987, pp. 4-22.

[3] Rumelhart, D.E., Hinton, G.E. and Williams, R.J, *Learning internal representations by error propagation*, in Rumelhart, D.E. and McClelland (Eds.) "Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations", MIT Press, Cambridge, MA, 1986, pp. 318-362.

[4] Williams, R.J. and Zipser, D., *A learning algorithm for continually running fully recurrent neural networks*, Neural Computation, vol. 1, 1989, pp. 270-280.

[5] Hecht-Nielsen, R., *Neurocomputing*, Addison-Wesley, Menlo Park, CA, 1990.

[6] McIntire, G., Villarreal, J., Baffes, P. and Rua, M., *Design of a neural network simulator on a transputer array*, in Proceedings of the First Annual Workshop on Space Operations Automation and Robotics (SOAR 87), 1987, pp. 111-118.

[7] Murali, P., Wechsler, H. and Manohar, M., *Fault-tolerance and learning performance of the back-propagation algorithm using massively parallel implementation*, in Proceedings of the Third IEEE Symposium on the Frontiers of Massively Parallel Computation, IEEE Computer Society, Los Alamitos, CA, 1990, pp. 364-367.

[8] Deprit, E., *Implementing recurrent back-propagation on the Connection Machine*, Neural Networks, vol. 2, 1989, pp. 295-314.

[9] DARPA, *DARPA Neural Network Study*, AFCEA Press, Washington, DC, 1988.

[10] Highley, D.D. and Hilmes, T.J., *Load forecasting by ANN*, IEEE Computer Applications in Power, vol. 6, no. 3, July 1993, pp. 10-15.

[11] Box, G.E.P and Jenkins, G.M., *Time Series Analysis: Forecasting and Control*,

Holden-Day, 1976.

[12] Ahalt, S.C., Chen, P., Chou, C-T. and Jung T-P., *Implementation of a vector quantization codebook design technique based on a competitive learning artificial neural network*, The Journal of Supercomputing, vol. 5, 1992, pp. 307-330.

[13] Wang, C-J., Wu, C-H and Sivasundaram, S., *Neural network simulation on shared-memory vector multiprocessors*, in Proceedings of the 16th Annual International Symposium on Computer Architecture, ACM, Baltimore, MD, 1989, pp. 197-204.

[14] Koike, N., *Neuro-computing and its application in supercomputer*, in Proceedings of the Singapore Supercomputing Conference '90, World Scientific, Singapore, 1991, pp. 260-265.

CONVEX—ON THE WAY TO A EUROPEAN CORPORATION

John Hughes
Convex Computer, Europe

1993—a watershed or simply more of the same? The presentation will discuss changes in European legislation, policies and practices, their impact on Convex and the Company's response. It will also discuss the short and long term steps that Convex is taking to align its Europe, Africa and Middle East (EAME) in response to our perception of the future of the EC. A brief review of EAME business results will also be provided.

CONVEX'S SOFTWARE PRODUCTS FOR RISC WORKSTATIONS

Jon Gelsey
Convex Computer Corporation, Tx, USA

The presentation will describe what current software products are offered, what future plans for products we have and what RISC platforms we plan to offer the products on. We would like to get feedback for the attendees about which additional items they want and what changes need to be made to the current CXSOFT product offerings. We will also describe the charter for CXSOFT and discuss our experiences so far.

FILE SERVERS - THE USER'S PERSPECTIVE

Jørgen Olsen, Dept. of Academic Information Systems, Odense University (DOU)
& Lennart Sorth, Danish Meteorological Institute (DMI), Denmark.

During the last three conferences various file serving solutions have been presented. In Oxford we heard of CSM, UNITREE and E-MASS. We also saw an optical disk device (HP Juke Box) placed beside - but not connected to - the Convex computer.

Having heard of the vendor's dreams of helping us with their wonderful solutions, this paper will try to look into some of the file management problems that can entice users to consider buying one of them. Further we shall also try to describe some problems and difficulties that may arise during the selection and installation process.

Due to the deadline for the submission of this paper, its presentation at the conference will also cover the experience gained between the beginning of September and the end of October 1993.

INTRODUCTION

When you consider buying a file server solution you soon discover, that you are in virgin territory. Your initial intention of visiting an installation which has been using the solution you would like to buy for at least six months - and talk to them before signing the contract - is impossible.

Certainly, if you want to invest in proven technology, e.g., buy a dedicated Convex server connected to multiple Hosts through a network and managing a multi-million dollar storage device, you can do that. But if you are just looking for a cost-effective alternate solution to your more normal data management problems, youcan expect responses such as : *"Sure, we can do it. It is in the next release that will be out in a few months!"*

CONTENT OF THIS PAPER

Initially the plan was to describe the solution selected at DOU.

Because of an earlier discussion of the subject by DMI at a meeting in the Danish Users Group, it was then decided to combine our experiences in one paper.

The intention was to describe our two different environments and data management problems as the background for our selection of different solutions.

This would be supplemented by a report on the experiences with the installed systems during the period (4-6 months) between the installations and the conference.

Circumstances described later in this paper have modified these intentions.

WHAT DOES A USER WANT?

That question is easy to answer:

a) Disk space enough for processing regardless of demand.
b) On-line access to all data files regardless of the time and the date as well as the age of the file.
c) Backup of files that will ensure that no system failure - hardware or software - will result in loss of data.

The above demands have become more complicated to fulfill during the recent years.

Earlier the demands referred to a single system and its users. Today most users are connected to a network so the demands are that those services must be provided - somewhere in the network!

CASE I

DANISH METEOROLOGICAL INSTITUTE (DMI)

DMI's main business is weather forecasting. This involves handling of enormous amounts of data that must be processed rapidly.

The main processing is done by a Convex 3880 with 2 GB of main memory and 3 GB of

RAID 5 disk. A Convex 3220 (512 kB mem & 11 GB of Disk) is allocated for file management.

Further the institute has: A Vax-VMS-cluster consisting of two 6410's - with 32 MB memory and 4 GB disk, more than 100 SUN workstations, mainly IPX (but ranging from IPC/SLC to IPX/LX with GX-graphics), 300 PC's of which 20 or so are used for operational data collection.

NETWORK

Network (Ethernet) with three 3Com NetBuilder-II bridges - connects 18 Ethernet segments. These bridges and the two Convexes are interconnected by FDDI.

The network distributes data to seven remote sites, primarily in domestic airports - including Greenland. This distribution runs on separate 64 Kbit lines.

The average data management is 2 GB per day. The average file size is 1 MB - ranging from zero to 100 MB.

SELECTION OF ARCHIVAL SOLUTION

After an initial investigation covering CSM, FILESERV and UniTree as well as several of the media managed by these systems, CSM was selected.

The main reason was its ability to offer *native* file migration; an important feature if you want to get to your migrated data with maximum flexibility and maximum performance.

Before the actual installation we unfortunately discovered that CONVEX had decided *not* to continue the development of CSM; in fact they wanted to give up the whole project. So the choice was narrowed to FILESERV or UniTree.

As FILESERVE is (was?) the more expensive one and since most of our colleagues in meteorology have chosen UniTree - so did we!

UniTree combined with a METRUM RSS48 robot (48 tapes each of 14 Gb) was purchased, and is for the moment running in acceptance test.

Because of the well-known poor performance of the UniTree handling of the robot, we waited for release 1.7 because we were told that the tape-buffering implemented in that version would result in a significant improvement of the write-performance.

RESULT OF INITIAL TESTING

One month after installation we are not very pleased. The performance is way below our expectations. The flexibility is non-existent and we have even experienced loss of data!

To retrieve a purged file we might have to wait 5-10 minutes even on an unloaded system. In our opinion the delay should have been one minute or so. The long delay seems mainly due to the performance of the Metrum-drive and/or its driving software.

The performance of NFS is inferior especially for small files e.g., files of less than 10 MB that are the only files that anyone would move via NFS (at a rate of 200 kbits per sec).

The graphics describing the results of some of the initial tests are given in Appendix A.

VARIOUS POINTS OF CRITICISM

- Release 1.7 of Unitree does *NOT* offer tape-buffering. Regardless of what your manual says and what your Convex sales rep have told you.
- The installation manual accompanying the release is *VERY* wrong in many places. This makes it extremely difficult even for experienced Convex system consultants to install the beast.
- The migration policy is nearly not tunable only an exponent based on *space*time* (since last ??) exists. No migration control based on UID/GID or directory/hierarchy is available.
- It is awkward that one has to generate user accounts *twice*. First in /etc/passwd using 'nu', then in UniTree using 'unsau'!

In general UniTree does not seem to trust anything available in your system. It would rather invent everything itself. The result is that system management becomes tedious and nonstandard.

All the things that one would like such a mass storage system to offer one could *invent* (write) oneself. There now is only *one* single file system (acutally one per running UniTree complex, but considering the number of processes and data files involved, having more than one should be avoided) with *unlimited* capacity. Into that system we could migrate stuff based on *anything-you-like* criteria by creating all sorts of sexy shell/perl/tcl scripts.

The unfortunate - but rather important - issue is : *We are also forced to invent and do it ourselves!*

A FEW GOOD POINTS

We are quite pleased with the robot control software written by Convex.

The part of the system we are really impressed by is the disk-cache handling. It is extremely nice to be able to include/exclude disks from the cache-pool while the system is running, and to be able to do it without any impact on the stability of the system.

CASE II
ODENSE ENGINEERING HIGH SCHOOL (OT) & ODENSE UNIVERSITY (OU)

OT has 1900 students and a staff of nearly 250, while OU has nearly 9000 students and a staff of 850 people of whom nearly 500 are academics.

A Convex 3240 (1 GB memory, 20 GB disk) takes care of the number crunching (OA research, structural analysis (ANSYS), and molecular research (GAUSSIAN etc)). Two HP-9000 systems (/845 & /827) handle larger packages (mainly of mathematical & statistical nature). One of them is also our main mail-server.

The network - more than 7 miles of internal cabling - connects 7 different geographical locations in the Odense area (see Appendix B). The total number of segments are 10 normal plus 40 thin cable segments.
Different locations are connected by 64kb, 128 kb and 2 Mb lines.

More than 800 PC's and 450 Mac's are connected to the network, directly or through one of the 12 Novell networks. Also a number of separate segment serves clusters of workstations (70 SUN, 30 HP/4xx & /7xx and various SG's & R6000's) spread out over the area (Appendix B.1).

INITIAL INTENTIONS

There is no end to the amount of disk space that users can fill up. There is also a need for backup, and in an educational environment automation of that function is desirable.

It was also decided to investigate the possibilities of offering automated backup of file systems in the larger workstation environments where that administrative function is often overlooked - from time to time with disastrous results.

In contrast to DMI there is in an academic environment no large production of data that must be managed. There ar two main reasons for that:

- Most projects have a limited span of life. If the data must be stored for later use e.g., if the investigation is repeated at a later date where it might be of interest to compare with earlier collected data - the files are normally moved to off-line storage or moved to the National Data Repository for safe keeping.
- We tend to forget that even though the servers of the academic network have near to 25 GB of disk space, the users as a group - on their PC's, Mac's and workstations - have much more! A conservative estimate indicates that the total amount of user allocated disk space in our environment approaches 125 GB (incl. netservers).

Certainly academic users would like to have more disk space available, but the normal complaint is that a number of files relating to earlier experiments migh be of interest later, and thus should be available. Unfortunately this means dumping them off-line to regain the amount of disk space needed for the actual running of the application.

TWO DIFFERENT ARCHIVAL SOLUTIONS

From the start we would like the system to manage the files of both the HP-systems and the Convex3240. So a dedicated system was of little interest.

The initial survey indicated that within our limited budget we had two choices:

- A stand-alone system (EPOCH) based on a SUN-platform with 1-2 GB disk space (cache) and a HP Juke Box with perhaps 20 GB (32 optical disc's) storage capacity.
- A UniTree based solution running off the Convex and with a HP Jukebox (20 GB or more).

DIFFERENT FUNCTIONALITY

The way those two systems functioned were rather different.

- EPOCH -
  As a stand-alone system relies on the interface with the various other systems defined by their operating systems. Because of the importance of performance, use of standard interface was not good enough; special routines were used. So complications with new releases might arise if inconsistencies between the OS levels in our systems and the levels expected to work together by the Epoch system occurred.
- UniTree -
  is an application running on - in this case - the Convex.

  The demand of processing power was not regarded as critical (given as maximum 10% of one processor during peak demand periods). What happens is. You tell UniTree which files or file systems you would like it to manage. It then takes the files, moves them into its own file system, and from there on manages them in its own way.

SELECTION OF ARCHIVAL SOLUTION

A testing of the EPOCH system indicated that the performance was in order. Unfortunately the handling of the Convex was a bit primitive because installation of a fully functional client module - according to the vendor - demanded changes in the kernel of the Convex OS. Something Convex had refused to do. So the file management was done through a script.

In October 1992 UniTree was the final selection - with an addition of a 2.5 GB disk device bringing the total up to the 20 GB mentioned earlier - and with a HP optical disk device (60 GB) as the archive.

The installation was planned to take place on April 16, 1993. Release 1.7 of UniTree was needed before the installation could take place. Earlier levels of the software could not use the Juke Box as data repository. Further the HP-systems should be on level 9 of the HP-UX operating system - a release that at that time had spent 6 months in the *"Will be released next month"* phase, probably with another 2-3 months to go before actual appearance.

Included in our decision was also the fact that Convex had taken over the future implementation of new UniTree-versions starting with level 1.7.

TOWARDS THE INSTALLATION DAY

In January, 3 persons (Convex system manager, HP systems manager & the systems operator that would be in charge of the day-to-day running of this future system) attended two workshops (each of two days) organized in Germany.

Though the hands-on parts of both workshops did not take place, the information gained was of value, but also the source of some concern.

It was discovered that for some reason the maximum number of files managed by a UniTree partition was limited to 2880. Further UniTree managed its files in blocks of 64kB (all .(dot)-files in that size!!).

This information generated quite a bit of reaction. An analysis of our file situation on all three systems indicated that the increase of the filespace in the Convex would increase to 204 %, while in the HP-systems it would increase to 213% respectively 286%.

An analysis of the size of our file system as a function of the blocking factor was done (Appendix B.2). Fax'es and E-mails as well as letters of a more formal nature were exchanged with Convex.

The results were very positive. At a meeting in The middle of March 1993 - we were informed that the number of files per partition had been increased by a factor of four, and that the blocking factor now was dynamic starting with four kB! The software was out in beta testing. Everything worked according to plan.

Two weeks later everything still worked according to plan. However, the plan had changed. As Convex had taken over the implementation of the software, its department of software quality control also came into the picture, and at the end of March it stopped the release dead after having reviewed the beta test results.

The new release date was The end of May, But due to the need for testing against the software managing the HP Juke Box (developed in Europe), our reception date would be The beginning of July.

Unfortunately our landlord (OT) had decided to renovate the building during the summer period, so we had to close down the Convex and move the rest of our facilities to another location for a period of six weeks starting at the end of July. Thus the installation date was postponed to September 20.

Recently it has been discovered that the software for the HP box supplied by DISCOS still provided *only 2880 files per partition and a blocking factor of 64 kb.* So that part of the installation will be further postponed.

AN UPDATE AT THE CONFERENCE

[As the deadline for submission of this paper is 6 weeks before the conference an update of what has taken place during that time period will be given as part of the presentation at this place.]

CONCLUSIONS

After having read the previous pages the reader must feel confident that our conclusion will be: *STAY AWAY from that kind of system!*

But that is NOT our conclusion!

Despite the present state of the art, this kind of system is here to stay.

- The demands for a data repository somewhere in the network is growing. Our users are in the process of overcoming their initial joy at having reached the *do-it-yourself* stage of the personal computer. Some of them have discovered the hard way, that they need a place where they can deposit data files that they would like to access much later.
- The technology is evolving. The 600 KB optical discs of today will be replaced by media that can carry 4 times as much information.
- Convex is with us(!!).

  Though the present state of the UniTree software is something of an embarrassment to us all. Convex has committed themselves. We believe that their notorious reputation for being able to deliver quality software will have an impact on the future development of the product. A stability (1.7.5) release is rumoured to be in beta-test. A further release with performance enhancements should reach us early next year.
- Standards are evolving. Whether they will be known as Andrew File System, DCE or something quite different is rather immaterial. Their acceptance, though some years in the future, will solve a number of the interface problems that are bothering us today.

FINAL REMARKS

Certainly, if you are not in too much of a hurry, either "*wait until you can actually see the solution you are interested in buying*" or ensure that your contract says :" *No music - no money!*".

If Convex is interested in speeding up the whole process, we recommend that they set up more formal communication lines with a number of interested sites that have bought the UniTree solution. The feed-back might enable them to create a better list of priorities for future enhancements, than they can do by just listening to the salesmen who are fighting to get into the next release those features they have (unfortunately) promised already exist.

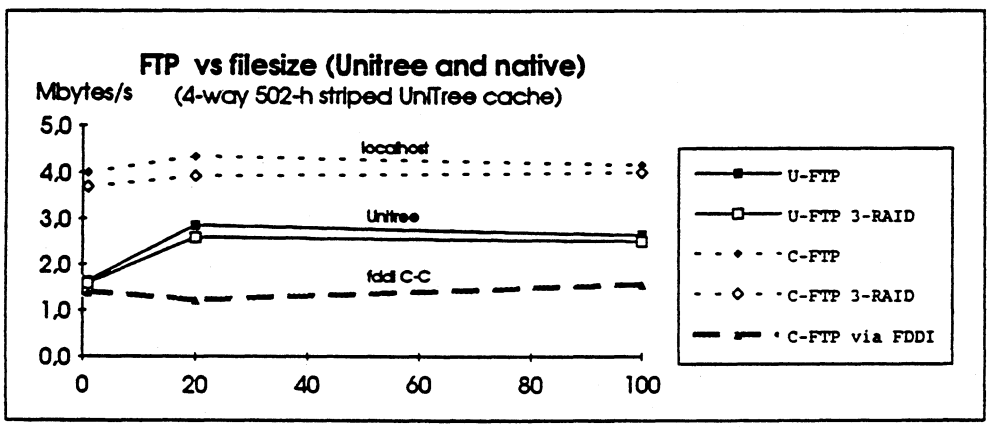
Without reference sites [= satisfied users of the product] it will be increasingly difficult to expand the number of users!

Unitree benchmark

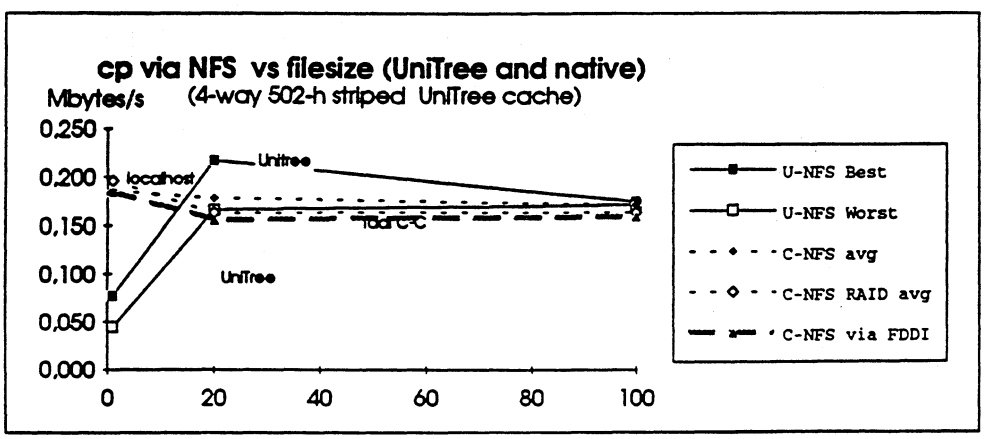
Appendix A

Survey at DMI 930729 by Peter Kaae Convex, and Lennart Sorth, DMI
Convex C3220 512M and 11G IDC/IPI 502 disks + one 504.
Metrum RSS-48 and UniTree v1.7 just installed.
Only one IDC controller (4 IPI ports) makes disk distributing important.

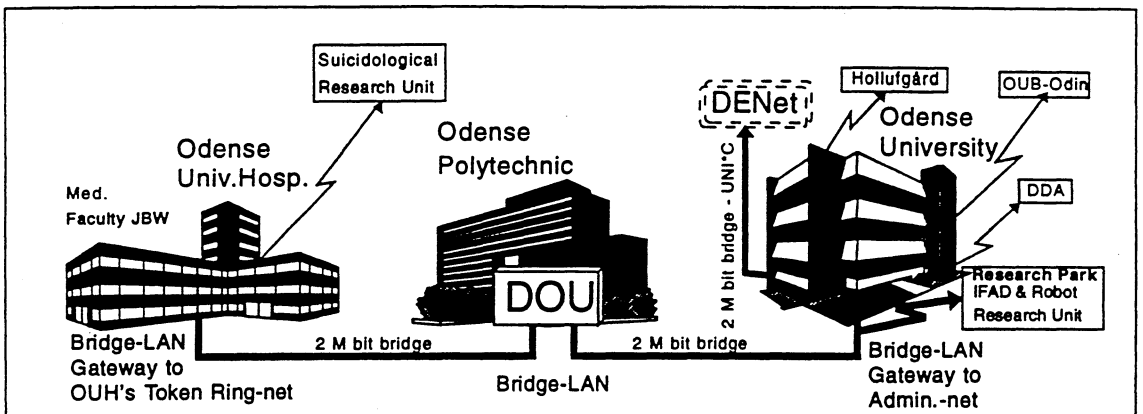
summary - Unitree vs Native			
Cache: 4Way STRIPE	1 Mb	20 Mb	100 Mb
UniTree:			
ftp to UniTree	1,6	2,9	2,6
NFS cp UniTree AVG	0,045	0,167	0,172
NFS cp UniTree BEST	0,077	0,217	0,175
Unitree RAID:			
ftp to RAID UniTree	1,2	2,2	2,2
NFS to RAID UniTree AVG	0,040	0,083	0,087
NFS to RAID UniTree BEST	0,073	0,104	0,090
Native:			
ftp to native	4,0	4,3	4,2
NFS cp native AVG	0,186	0,179	0,172
NFS cp native BEST	0,192	0,182	0,175
Native RAID:			
ftp to RAID native	3,7	3,9	4,0
NFS to RAID native AVG	0,196	0,164	0,164
NFS to RAID native BEST	0,196	0,166	0,164
Via FDDI:			
ftp to native	1,408	1,224	1,563
NFS to native AVG	0,185	0,157	0,160
NFS to native BEST			



Its evident that the UniTree-FTP is quite slow when handling small files - must be due to the UniTree nameserver. It seems that unamed does a very poor job whan handling several simultaneous requests... (!)
In general the native FTP is much faster - no matter filesize.
The native FTP thru FDDI (C3880-C3220) seems to perform around the 1.5 Mbyte/sec



On top of being very slow the UniTree NFS is also very stochastic - giving very different results from time to time. This may again be explained by the slow unamed - the deviation is therefore not seen as much for large files.
Native NFS seems to be FASTER for small files (but it may be a coincident)
The only thing UniTree caould be proud of is the NFS performance for files larger than 10Mb - but who would transfer 10++ Mb with an overall performance of 200kb/sec ???
Native NFS over FDDI is undistinctable from native NFS to localhost. (great!)



LAN & COMPUTERS managed by DOU

BETA:
CONVEX 3240
Research and Educ., AVS
NFS, ANSYS, Gaussian90
NAG, Sciconic, GAMS
1.024/20 GB mem./disc

OPTICAL ARCHIVE:
HP,60GB, Unitree

NET:
8 diff. locations, 10+40 segm.
12 km of internal cabling
LAN - TCP/IP - 10 Mbit/sec

DELTA
HP9000/827
Research & Educ.
NFS,SPSS,SAS,E-mail
48/2.0 MB mem/GB disc

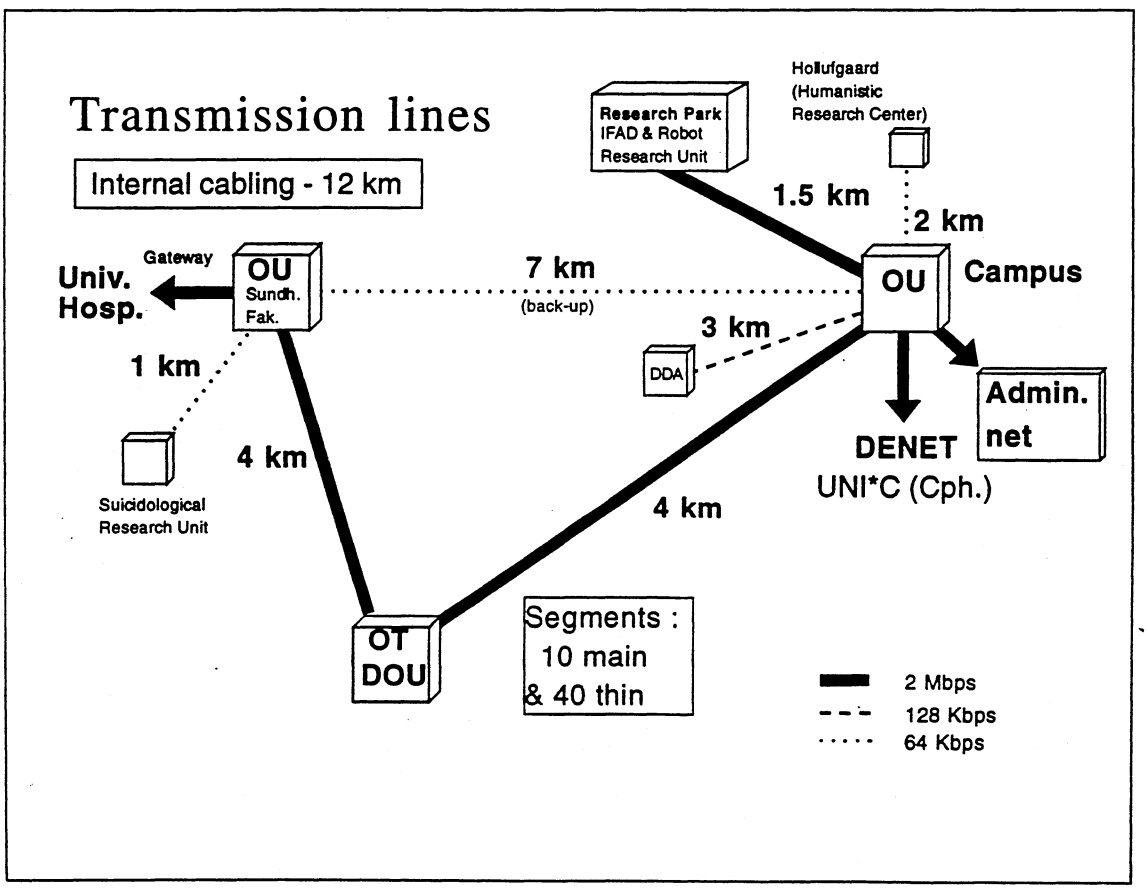
KAPPA:
UNISYS 5000/50
X.25 comm.
8/170 MB mem./disc

800 PC's + 400 Mac's connected

GAMMA:
HP9000/845
Research & Educ.
NFS, SPSS, NAG
24/2.6 MBmem/GBdisc

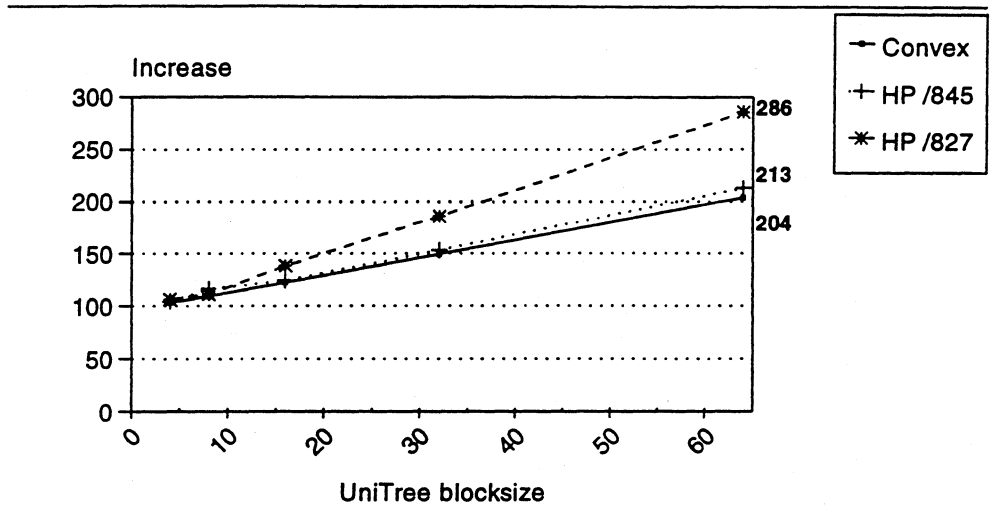
Misc. connected systems:
HP4xx/7xx OT & Dep. of Phys. OU
SUN & Meiko-MPP, Mat.Inst. OU *)
HP9000/3XX(3), OT
PC- og Mac-net (8)

*) Run by Dep. of Comp. Science.



Increase of total size of files

- as a function of the blocksize



Before convertion to UniTree = 100.

EXPANDING TO NEW MARKETS—TARGETS AND CHANCES

Jim Balthazar
Convex Computer Corporation, Tx, USA

Supercomputers have always used the most advanced technologies available to gain the highest performance possible for the technical market. New markets are emerging that need these same high-performance technologies. These are not the traditional technical application areas, but they still share the same needs of the high-performance technical user. What markets can use these high-performance technologies, and how will Convex succeed in these markets will be the subject of this presentation.

MASS-STORAGE—STATUS AND OUTLOOK

Dan Henderson
Convex Computer Corporation, Tx, USA

Mass-Storage is a market Convex has great advantage in. We have made major resource commitments to a complimentary strategy of data management and compute servers. By combining the data management products with the meta-series, SAS and SPP, the mass-storage opportunities are numerous.

This presentation will present an updated status of Convex's mass-storage activities and present an outlook for Convex's opportunities in the mass-storage market.

The event daemon exposed

Measuring and sizing fileserver activity

Kevin Ashley
ULCC
K.Ashley@ulcc.ac.uk

Abstract

The event daemon in ConvexOS allows an application to intercept significant events in a filesystem and carry out a range of actions when they occur. The daemon can permit or disallow the completion of events, and undertake arbitrary actions before allowing the event to conclude. The obvious application is for a hierarchical or virtual filestore, but other, more varied, uses are possible ; this paper describes how the daemon interface can be used to gather performance information for sizing a virtual filestore, and how it can be used to implement other functions, such as access control lists.

Some of the work described in this paper is still in progress. It is likely that the final presentation will be somewhat different from this paper, which should be seen as providing background material for the presentation itself.

1. The problem of sizing a virtual filesystem

A number of studies have been undertaken over the years to attempt to model the performance characteristics and requirements for hierarchical filesystems. However, almost without exception, they have been done on existing hierarchical systems such as CFS which have been in place for a number of years. They have also usually worked with only a very small sample of data because of the problems of collecting large amounts of log data on such systems. The studies have provided some useful information: for instance, supercomputer sites have tended to find that files are written to at a relatively continuous rate day and night, since file creation is mainly done by batch jobs, whereas file reads show a marked variation, with most taking place during normal working hours, since most of these are initiated interactively.

It is inevitably the case that, when a system has been in place for so long, users have altered their behaviour to suit the limitations and shortcomings of the system. Measuring what users do now does not measure what they want to do or need to do, but only what they are able to do. The techniques described here can be used on an existing, normal filesystem and one presumes would give a picture of what access patterns to files could be expected if a system behaved ideally. They can also be applied to existing EMASS/FileServ or CSM-controlled filesystems, but not to those controlled by Unitree.

One of the problems in specifying a fileserver or data management system to implement a virtual or heirarchical filesystem is that it is difficult to predict what requirements one will place on it. Suppliers can usually provide good statistics on the total throughput a system can provide, the number of stores and retrievals per hour it can handle, the number of tape

mounts and the average access time to an offline file. However, to know what this means in a particular environment means collecting statistics from that environment to determine what demands it will place on the system. This is currently a difficult task. Even if one assumes that one has an existing, disk-based system which will expand to a heirarchical system, it can be difficult to determine how many files are accessed per day, how often, whether they are updated, and therefore how big to make the disk cache in a virtual system, how many migration drives to support and how fast they should be.

ConvexOS, however, does provide the mechanism to create a daemon to collect such information. The *event daemon interface*, as it is known, has been a standard part of ConvexOS since version 9.1. It is the mechanism used by FileServ (and CSM) to intercept filesystem events and migrate files on to disk as necessary. It exists in every ConvexOS system, and is documented in programmer's guide for version 10.0 systems onwards. Enabling it merely requires the alteration of a single OS tunable. Using it to produce a full-scale file migration system is a large project and would require significant effort. However, using it to merely gather statistics is a much simpler task, requiring only a few weeks of effort. Analysing the statistics thus gathered is, however, a far more time-consuming project.

2. Event Daemon Basics

At this point, it is worth giving a brief description of the ConvexOS event daemon and its relation with the rest of the system.

Up to four event daemons can co-exist within a single system. Each file in a filesystem can be associated with any or all of the daemons. If an existing filesystem (or part of it) is to put be under the control of a daemon, a once-only operation must be performed to associate each file of interest with the appropriate daemon, via the `DMON_FDSETASSOC` subfunction of the `dmon_fcntl()` system call. If it is necessary to collect information about allocation of new disk blocks to existing files, it will be necessary to use the `DMON_FDSETFLAGBITS` function to set the 'migrated' bit in the file's inode. If a file is associated with more than one active daemon then events are passed to the daemons in priority order, with daemon 0 receiving the callout first. One extra word in the inode is used for event daemons - a 16-bit mask which indicates daemon associations, an 8-bit field which is available for use by the daemon itself, and an 8-bit field reserved for system use. One field controls whether inheritance is in effect - if it is, then files created within a daemon-controlled directory are automatically associated with the daemon. This is the usual mode of operation (but not the default for the system call.)

Real migration daemons also need to worry about associating themselves with filesystems to receive notification of when free space has passed a low or high threshold. Security or statistics daemons don't need to use these events.

A daemon, when it initiates itself, needs to register with the operating system to declare itself as an event daemon. It does so using the `DMON_INIT` subfunction of `dmon_ioctl()`, having first allocated an kernel rpc channel with `DMON_CHANALLOC`. At this point, it declares which daemon it is (a number from 0 to 3) and which events it is interested in intercepting. The events it can specify are:

- open
- close
- read
- write
- create

- remove
- access - use of `access()` system call
- bmap - attempts to read or write existing, but migrated, blocks
- bmap_hole - attempts to allocate new blocks to the file
- truncate - attempts to truncate blocks from the file
- rename
- symbolic link
- link
- chmod
- chown
- faults
- low-block conditions on a filesystem (LOBLK)
- high-block conditions on a filesystem (HIBLK)
- filesystem full (NOBLK)

Only a small subset of these events are available if the file being monitored is a special file.

Once it has registered, the daemon enters a simple loop waiting for events to occur, processing them, and acknowledging them to the kernel. Most events cannot proceed until the daemon notifies the kernel that they are allowed. The calling process is blocked whilst the daemon is deciding. Some events (the filesystem block events, and bmap_hole) are asynchronous. The daemon is merely notified by the kernel, and a reply is neither expected nor possible.

The action taken by the daemon is entirely under its own control. It may allow or disallow events using entirely random criteria, it may use supporting databases to decide how to respond to an event, and it may take many hours before deciding whether an event is to be permitted or not. A migration daemon, for instance, will wish to know about new files being created, and existing files being modified, so that it can mark them as requiring new offline copies to be made later. It also wishes to discard offline copies of files which have been altered or deleted. It needs to know about attempts to access offline files so that it can bring them online, and it may also use the frequency of access to files to decide which files to remove from the online cache when space on disk is low. The daemon needs to take care that it does not spend too long processing a callout, and that it does not place itself in a deadlock situation by attempting to access a file which it controls itself.

Two mechanisms are available to daemons to assist them with the latter points. For the processing of time-consuming operations, the daemon can create a child process and inform the kernel that the child is handling all events for a single file temporarily. The child can deal with the migration of a file from offline storage, for instance, while the master daemon continues to process other events for other files. When the child or paremt needs to actually write to the file in question, it can suspend callouts on the file, which allows it to perform operations on the file without generating further daemon callouts which it cannot process itself.

If attempts are made by a process to access a daemon-controlled file when the daemon is not present, the process can either block until the daemon becomes active, or can receive an immediate error: (`ENODMON`). Which happens is controlled by the process flag `UMIGNOBLK`, or by use of `FNDELAY` or `O_NDELAY` in the `fcntl()` or `open()` calls made by the process.

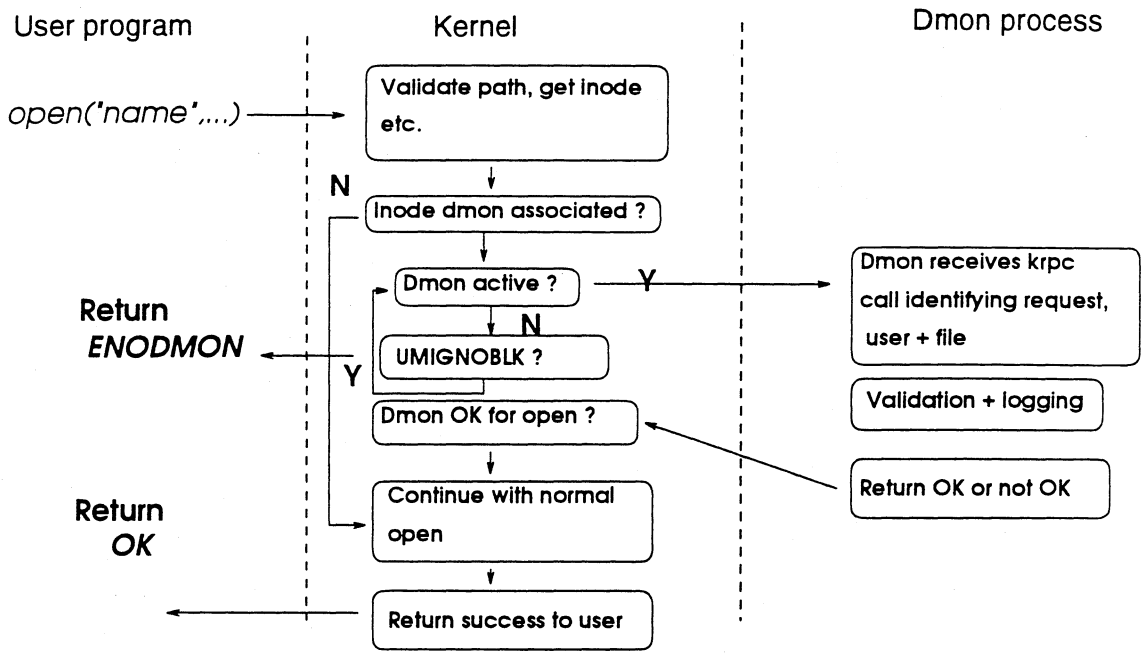


Figure 1: Event daemon operation

3. Collection of statistics

Armed with the above information, the reader should now see how easy it is to collect information from an existing filesystem which could give one an extremely detailed picture of what the demands on that filesystem are. An event daemon can be used to monitor every event which would be likely to cause a hierarchical filesystem to take action. It is simplest to monitor every event (except perhaps reads and writes) and decide later which are important. As well as the event itself, the inode, filename, parent inode, requesting pid, user and application and a timestamp should all be recorded.

The statistics can be recorded in a bare format and should ideally be output in large chunks (at least a megabyte at a time) to avoid distorting the I/O throughput on the system. It is prudent to cause all processes in the system to have the UMIGNOBLK flag set, so that if the statistics event daemon fails for any reason, users are able to continue to access files without being aware of it.

The statistics themselves may require frequent processing. If one assumes that a typical event record consumes 100 bytes, and that hundreds of events per second can take place at busy times, then it should be realised that it may not take long to collect 100 Mbytes of event records. It may even be necessary to acquire a virtual filestore in which to store the filesystem event logs.

4. What numbers do we want ?

There are a number of constraints on the performance of fileserving and hierarchical filestore systems. Suppliers can usually give good estimates of the delays involved for each of the following:

- Offline access time - How long it takes to mount an offline volume, and position to the appropriate point. The time is measured from the point at which access is attempted to the point at which transfer of data begins. It usually varies between set limits depending

on the position of the file on the media. On some systems, the requesting process may begin access to the data almost immediately.

- Offline transfer time - How long it takes to transfer all the data to disk. Dependent on file size.
- Offline busy time - How long a drive is tied up handling a request to bring a file offline. Measured from the beginning of the request until the point where the drive is able to complete another action. May or may not include dismount time.
- System overhead for onlining - the CPU and I/O resources consumed by bringing a file online. May include database access and similar actions
- System overhead for migration - the CPU and I/O resources consumed in selecting files for copying to offline storage, and then moving them.
- Migration busy time - How long a drive is tied up handling offline migration of files.
- Disk speed
- Cache management overhead - how much time and resources are consumed in deciding which files to remove from online cache when space is short. Such resources may be consumed in a slow, continuous fashion, or may be consumed in spurts (such as when a filesystem full event occurs.)
- Cache management criteria - Although different criteria may not appear to affect performance directly, a system with inflexible management criteria, such as one which only uses the age of a file to determine if its online copy should be removed, may perform very poorly in some environments, requiring much larger disc caches or many more migration events. One which allows many factors to be taken into account, and which allows the weighting to be varied depending upon the type and location of file will provide greatest efficiency.

Factors which we do not know, but need to take into account when sizing a fileserver, are the size and speed of the disk cache, the number and speed of offline devices supporting it (optical disk or tape ? how many drives) and the power of the CPU supporting it. These are factors which we can model to some extent in our analysis of the statistics gathered using the statistics event daemon. In addition we will wish to model the effect of changing the criteria for moving files offline or online. Some systems only allow a single factor, such as file age, to be used when determining which files should have their online copies removed. Some allow more complex weightings, using features like file age, file size, last use and decayed frequency of use. Others can also use this data to decide whether or not to even make an offline copy - for instance, if a filesystem includes a number of short-lived files, it may not be worth making any offline copy of a file which is not at least 12 hours old. In our modelling, we may wish to compare two products, one of which offers more flexibility in determining which files remain in the cache, or we may wish to compare one product with itself, using different weightings and observing their effect on the total simulated load.

5. Analysis of statistics

As noted above, the factors we would like to get guidance about include the number and speed of offline drives required, and the size of our online disk cache. A particular modelling session may examine some or all of these criteria, and different subsets of the event records will be required for different models. As well as using a representative set of event data (collected over a week, at least) one needs to start a modelling session with a snapshot of the filesystem, and some assumptions about the size of the cache and the initial state of the files. In fact, the best modelling is done when one initially runs the statistics daemon for a few days

to build up information about which files are currently active ; one then takes the snapshot, which records information about every file in the filesystem, including its size and all inode dates. The simulation process is begun at the time the snapshot is taken. It, together with the access records from the previous few days, is used to build up a simulated initial state for the filesystem, in which some files are considered to be offline and some online.

Further modelling is fairly simple if we merely wish to examine the desired cache size. Assumptions are made that file storage and retrieval is instantaneous. The model uses the collected event records to simulate what files would need to be brought online when. It tracks the total space used online, and uses model parameters to decide which files to move offline when the simulated filesystem becomes too full. The output of the model will show the number of file movements taking place at any given moment, and can tell us how much data is being moved on or offline at any given time. We can also discover how many processes would have to block while offline files were retrieved, and when they were doing so. All this information can be useful - it may be acceptable for batch processes to block overnight, for instance, but not for interactive sessions to do so during the day.

This simple model requires three inputs, apart from the collected event data and filesystem snapshot data: the size of the disk cache, the algorithm used to truncate old files from disk, and the inputs to that algorithm. For instance, the algorithm used by FileServ makes use of the last access time to a file and the size of the file. For each of these variables, the system manager sets weights to be applied to all files in a specified range. For example, files smaller than 20 Kbytes may be given a weight of 1000 ; files between 20Kbyte and 1Mbyte a weight of 10; and larger files a weight of 1. Similar weights are applied to ranges of file ages. Finally, a weight is computed for each file by multiplying the file size by the appropriate weight and the age by the appropriate weight and adding them. This is used to order the files to decide which it is most desirable to keep online. The usual scheme makes the system favour truncating large files and old files ; a young, large file is usually better to move offline than a number of small, old files. More complex models might take into account the filename, making exceptions for certain files. One also might wish to model the fact that FileServ, for instance, will not even make a tape copy of some files until they have exceeded a certain, site-defined age. Thus certain files may not be eligible for removal from disk when very young since they will have no tape copy.

The numeric inputs to this model can easily be implemented in an X interface via sliders and simple dialog boxes. One can rapidly alter the parameters, have the simulation rerun, and see the effect on number of file movements and processes blocked in a graphical format, which is easy to assimilate.

Performing a fuller model, which takes into account numbers of drives, access time to files, mount times and processing times, requires a more complex simulation system and much more thought and preparation for the user performing the modelling. One needs to feed many parameters to the model, such as speed and number of drives, seek and transfer times and the frequency with which migration policies are initiated to make offline copies of files or to remove online files from disk. A usable interface would need to encapsulate many of these factors in simpler choices - for instance, the user could simply select 'drive type' from a list offering 3480, D2, DAT or optical, and the parameters for access time, transfer rate, etc. would be set from parameters known to the program. Expert users, or those with a lot of patience or time to spare, could define new drive types along with all the characteristics appropriate to them.

Such a model could tell us many other things about the system. For instance, most tape-based systems rapidly run into a problem whereby tapes become fragmented if files are altered after entering the system, or if they are deleted. All systems except pure archives exhibit this behaviour and most have some procedure, manual or semi-automatic, for coping with it. A detailed enough model could predict how soon such events are likely to occur. Some

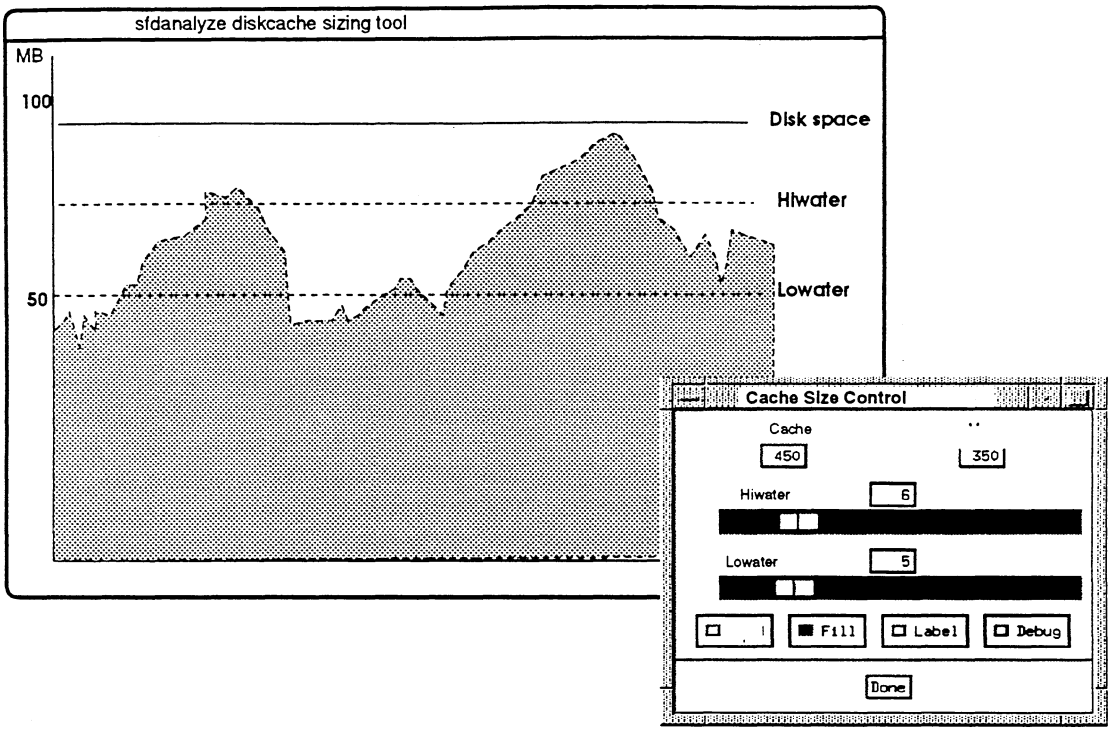


Figure 2: Analysis of disk cache

media do not have infinite lifetimes - rewritable optical media, for instance, may only sustain 5,000 writes to the same disk block before becoming read-only media (or even worse, read-never media). Again, detailed enough modelling can tell us how quickly this is likely to occur with a particular pattern of file use.

There are obvious flaws in a modelling process which uses data gathered from any existing system, whether or not it is already hierarchical. As is the case with the studies of systems such as CFS, users accesses to files is to some extent affected by what they know the system is capable of. A user wishing to use very large files on a system without a hierarchical filestore may need to recreate some files whenever they are needed because the storage does not exist to keep them online for a long time. In a hierarchical system, such files may be created once and allowed to migrate to offline media, being brought online again where necessary. This may save significant processing resources on the host system. Such flaws need to be acknowledged when we make use of simulations, but one always has to realise that no simulation is perfect. Nonetheless, significant insights can be gained into the difference, say, that a 30% increase in disk cache size may make to the performance of a system. Once the tools exist, significant savings can be made by knowing in advance whether an investment in equipment is likely to make a useful difference.

6. Using the daemon interface to implement ACLs

What has been covered so far is a standard application of the daemon, not far removed from the implementation required for a true virtual filestore. However other, more exotic, possibilities exist. The fact that a single file may be associated with up to four event daemons means that these other possibilities can co-exist with other applications such as FileServ. One possible use that has been investigated by the author is a security daemon, implementing a form of Access Control List (ACL) security in a system. Access Control Lists will be familiar to anyone who has used VMS, UNICOS or ConvexOS Secure. They allow finer-grained control for access to files than the traditional Unix owner, group, world method. Most significantly, they allow a user to decide who can and cannot access particular files, without requiring system managers to create appropriate groups. User A may be permitted to read a file, user

B may be permitted to write it, and others may be denied access completely. User C may be granted 'control' access, meaning that they are permitted to change the access permissions on the file. In addition, some ACL systems allow for the creation of Alarm ACLs, which may permit or deny access, but also alert the user or system manager of the access attempt. ACLs placed on directories can cause default permissions to be applied to all files created within that directory, as well as controlling access to the directory file itself.

ACLs implement what is known as an "access matrix" in security models: one can view the filesystem and users as a large matrix, with one user per row and one file per column. Each cell in the matrix represents the access which one user has to one particular file, and each cell's contents is independent of others. This model is impossible to realise properly with Unix 'ogw' protection mechanisms. One can look at the system from two perspectives: examining the rows provides a capability-based view. For each user, it tells us which objects that user has access to. Examining the rows provides an authority-based view; for each object (or file), it tells us who us authorised to manipulate it.

The event daemon is well suited to implementing an ACL system. The first question which arises is where the ACL itself is to be stored. One possibility is to use a database ; this forces creation or modification of the ACL to take place via dedicated tools, and means that ACL information is likely to be lost when whole directories are moved from place to place using *cp* or *tar*. Complex tools need to be be built to reconcile the database with the filesystem when, for instance, the filesystem is restored from dumps after a disk failure. Another possibility is to store ACLs within 'dot' files (those whose name begin with a ".") in the same directory. One could still provide tools to create and manipulate these files, but they could also be edited directly with a simple editor if need be. The event daemon controls access to these special files as well, ensuring that only authorised users can change or delete them. One could also force access to the files to take place only via dedicated tools if this were required.

The database or dot files need to contain simple records defining sets of capability/authority pairs - in simple terms, a list of who is allowed access to what files. "Capability" will typically be a user name, but could also be a group name or some other, more generic capability. For instance, one might wish to grant, or deny, specific types of access to any process running in batch, or to processes which originate from a session outside one's own site. The event daemon knows which PID is requesting access, and can easily follow the process tree to discover if the process originates from a telnet or rlogin session from outside the site.

The actions an ACL event daemon takes can be summarised as follows:

- Watch for creation of ACL control file (.aclcontrol) in a directory. When it is closed, read and cache its contents.
- Watch for updates to ACL control file, and check that they are from an authorised source. Invalidate in-core cache for that directory when control file updated.
- Monitor open, close, remove, rename, read and write events. Read and cache appropriate ACL control files as necessary, and use them to implement access policy.
- If security ACLs are supported, record events which trigger security ACLs. Users or system managers could be notified by mail of these events, probably by generating messages after an appropriate delay (10 minutes ?) so as to batch related events together.

Special care needs to be taken when attempts are made to remove an ACL control file. Obviously, only an authorised user (one permitted to alter the ACL) should be permitted to do this. But also, one may wish to protect the owner from accidentally removing the control file, unless the entire directory it controls is being removed as well. It is possible for the event daemon to handle this either by only permitted the removal of a control file via a special tool (aclremove), or by checking that after the ACL control file is removed, its parent directory is also removed within a short interval and recreating the file if not.

There are shortcomings with this simple system. Since file manipulation utilities such as *cp* and *mv* are not aware of the existence of ACLs they will not propagate them when moving files. This may or may not be desirable. Utilities such as *tar* will generally pick up ACL control files along with the rest of a directory, and will attempt to restore it when the tar file is unpacked. How one handles this is a matter of policy - any desired action can be taken by the ACL daemon in these circumstances.

7. Other possibilities

Event daemons can be used to implement a number of other features. They can be used, for instance, to provide a simple output spooling facility where files are entered into the system via ftp, rcp, DECnet or any other mechanism. ULCC currently operates such a system (not using the event daemon) whereby its C220 provides access to a microfiche and film plotter. Files are sent to this system from all over the UK, and the size and nature of the data make lpr an unsuitable mechanism for transmitting them. In any event, lpr does not provide sufficient security to know who is responsible for printing the files.

An event daemon monitoring a particular section of filestore could react as soon as a file is created in the filestore. It can verify where the ftp or rcp session originates before the file is closed, and record this along with any other accounting information required.

Event daemons could even be used to implement some form of access to non-Unix filestores, although it is difficult to see how to do this effectively in anything other than a read-only fashion.

8. Conclusions

The event daemon interface is a little-used but powerful tool for gaining insight into activity within a filesystem and can provide valuable data for the sizing of new fileservers and data management systems, and the collection of data from existing systems. Use of such data to simulate the activities of real systems can lead to useful insights being gained into their design. The event daemon interface is also capable of supporting a number of other useful functions.

9. Acknowledgements

I am grateful to all those at Convex and E-Systems who have passed on useful snippets of info about the behaviour of the event daemon, but particularly to Debbie Neuts for supplying the Event Daemon Interface Manual. Many thanks are due to all those in the ULCC systems group who have worked so hard to get our hierarchical filestores active and discovered a great deal about them in the process, but particularly Christine Bourne, Bob Pockney and Charles Liston.

Kevin Ashley, ULCC
October 1993

INPUT-QUEUE SCHEDULER

Wolfgang Stahl
DKRZ Hamburg, Germany

The input-queue scheduler was developed by Convex Germany for the German Climate Computer Centre. This software computes the inter-queue priority from the consumed CPU time. The input-queue scheduler can control individual queues. Logfiles are produced of jobs that are queued, running, finished, and of jobs that were running during an initialization. The presentation will show how the input-queue scheduler works as well as giving implementation details.

THE HISTORY, PRESENT STATUS, AND FUTURE PLANS OF THE ACADEMIC COMPUTER CENTRE CYFRONET – CRACOW

Marian Noga, Marek Księżyk and Zofia Mosurska
Academic Computer Centre CYFRONET – CRACOW

1. Introduction

Ancient Cracow, the city with 700,000 inhabitants located upon the Vistula river, the old capital of Poland until the 17th century, belongs traditionaly to the leading cultural and scientific centers in our country. In Cracow there are 12 universities (among those is the renowned Jagiellonian University founded in 1364, whose most famous student was Nicolaus Copernicus) and a lot of research institutes. The academic community of Cracow numbers about 8,000 scientists and about 51,000 students.

According to the data taken from the Polish Statistical Yearbook 1992, there are 101 academic schools in our country. They are located in 24 cities. The whole Polish academic community numbers about 60,000 scientists and about 418,000 students. The most important academic centres are Warsaw (17 academic schools, 10,000 scientists and 74,000 students) and Cracow. Other significant Polish academic centres are Wrocław, Poznań, Gdańsk, Łódź, Katowice, Lublin and Szczecin. A variety of research institutes in Poland should be also mentioned. There are 371 non-academic institutes with about 77,000 employees. 75 institutes belong to the Polish Academy of Sciences (PAN). Among 11,000 employees of PAN there are 4,400 scientists.

The history of academic computing in Poland started in the middle sixties and is closely related to both political and economic situation. The early seventies happened to be a period of relative prosperity (western credits which we have been paying back so far!). Many Polish academic schools and research institutes obtained their computers during this time. There were first of all Polish ODRA computers (with licensed ICL software), RIAD computers (IBM 360 like) and SM minicomputers (PDP like), both manufactured in various countries of Eastern Europe. There were also two Control Data CYBER computers and one French IRIS computer installed for the needs of the scientific community of Poland in the middle seventies. Unfortunately the economic crisis first and political crisis later on, appeared in the late seventies. There was no money for buying modern equipment for science. Furthermore our scientists were not allowed to use the biggest computers in western institutions, even when carrying on joint research projects with these institutions.

Well known changes in our country happened in 1990. We still don't have sufficient funds for science but already in 1991 Poland was able to install its first vector supercomputer. By the way, CONVEX C120 delivered to Cracow has been the first supercomputer not only in Poland but in all of Eastern Europe. The current political situation is improving. Just now we are waiting for an export license for the C3820 supercomputer. In 1990 the Polish academic community joined the European Academic and Reaserch Network (EARN), and in 1991 connections with INTERNET became a reality.

2. The Academic Computer Centre CYFRONET – CRACOW as a leading Polish computing centre for academic community

2.1 History

Progress, the present situation and the future of academic computing in Poland may be exemplified by the activities of the Academic Computer Centre CYFRONET – CRACOW. CYFRONET was established 20 years ago, as an independent non-profit organization under the authority of the Ministry of National Education, to provide the Cracow academic schools and some research institutes with powerful facilities for tasks which were beyond the scope of their own computer installations. CYFRONET delivers computer power combined with a wide range of application software and computer network services to members of the academic research staff and students, runs courses and seminars, produces manuals and provides advisory services to the users, everything free of charge. CYFRONET has been financed by the Ministry of National Education and the State Committee of Scientific Research.

Our first computer, the Control Data CYBER 72-16 was delivered in 1975. But already in the early eighties it was obvious, that the CYBER system was not suited to cope with the variety of large scale calculations. After some years of stagnation, a working group set up by the Council of Rectors in 1989, recommended courses of action in two main areas: academic computer networks and supercomputing in Cracow.

Following the careful analysis of the computer market we determined to buy the CONVEX C120 mini-supercomputer. Obtaining an export approval for the system turned out to be a long difficult, and highly political process. Finally, the export license for the first vector supercomputer in all of Eastern Europe was granted in March 1991. On April 22nd the C120 supercomputer system was installed in Cracow. The characteristic feature of the CONVEX utilization has been the very quick employment of the full capabilities of the system. Our small C120 computer has proved not to be up to the needs of the Cracow academic community. Therefore we decided to ask CONVEX to apply for the export license for the C3210 supercomputer. The whole approval procedure was much easier this time and the C3210 was installed in Cracow in June 1992.

Another aim of our activity is to accelerate the availability and utilization of the computer networks. Cracow has already got both EARN (1990) and INTERNET (1991) world-wide connections, we are also using DECnet and X25 experimentally. We are realizing the development plan of the future Cracow Academic Computer Network. The first, most important parts connecting the Jagiellonian University and the University of Mining and Metallurgy with CYFRONET on a fiber optics basis have already been set up.

The IBM 4381 computer was installed in 1990. It was the first equipment which was more powerful than the old CYBER computer. Utilization of our "Big Blue" for numerical calculations has never been significant because of the operating system which has been unknown to the academic schools of Cracow. Another reason was the installation of the C120 in 1991 which obviously proved to be much more attractive to the users. The main cause for the purchase of the IBM computer was to obtain the EARN membership by the Polish academic community. PLKRCY11, the third old EARN node in Poland was installed on our IBM in November 1990.

It has been connected by the BSC link to the Polish central node (PLEARN) installed on the IBM 3090/170J at the Warsaw University first. Both those nodes have been connected by the SNA link later on.

The most important events in our twenty years history are shown below:

1973 - Establishing CYFRONET-CRACOW
1975 - Control Data CYBER 72-16 with 20 remote terminals
1979-80 - Minicomputers
1981 - New CYFRONET building under construction
1982 - New CYFRONET building suspended

1982-87 - Mini- and microcomputers
- Personal computers
1988 - Designing of the Regional Computer Network started
1989 - Changing the direction of CYFRONET
- Council of Rectors decided to develop CYFRONET
- Personal computers
1990 - Own telecomunication cable
- IBM 4381
- Personal computers
- Cutoff the mini- and microcomputers
1991 - EARN started
- Cutoff the CYBER computer
- CONVEX C120
- SUN workstations
- Removal into new building
1992 - CISCO AGS+
- First sector of the fiber optic connection
- SUN workstations
- INTERNET started
- CONVEX C3210
- MicroVAX 3100 and VAX 8200
- Technical Support Cooperation Agreement between CONVEX and CYFRONET
1993 - Application for a grant from the State Committee of Scientific Research
- Application for the C3820 export license
- Cutoff the IBM 4381 computer

2.2 Hardware

The Academic Computer Centre CYFRONET – CRACOW, established in 1973, was equiped with modern and relatively big (at that time) Control Data CYBER 72-16 computer with 20 remote terminals in 1975. This computer has been used until 1991 when it was replaced by the CONVEX C120 mini-supercomputer. The C120 with the Processing Data Rate equals 175 was then the most powerful computer we could get export permission. Although the installation

of this computer was a real "mile stone", according to our expectations it has proved not to be sufficient for subsequent increasing needs. Thanks to the positive changes in the Polish political situation we were able to install the C3210 supercomputer in June 1992. The C3210 with 128 MB of memory and 4 GB of disk space is now our main computer.

Some months after the installation, the new C3210 supercomputer proved again to be too small. Therefore the second C3200 processor (together with VMEbus and FDDI interface) has already been ordered and is expected at CYFRONET soon. Much more disk space is also urgently needed and will be purchased. But we are sure that a significant improvement can be only achieved after the installation of the C3820. Our approval procedure for that powerful supercomputer is now in progress at the US DOC and COCOM. We hope that our export license will be granted in November 1993.

Nowadays we are using some workstations at our computing centre, too. They are four SUN SPARCstation IPC and three SUN SPARCstation SLC with the total disk capacity of 3 GB. Beyond the UNIX equipment we have got some others systems running under VM/SP, VMS, and DOS. In 1990 the second-hand IBM 4381 computer with 4.1 GB of disk space was purchased. This computer has been used as the EARN node until September 15th, 1993. In 1992 we installed VAX 8200 (second-hand) and MicroVAX 3100 computers. A lot of 286, 386 and 486 PCs are also in operation. All computers ranged from PC/AT to CONVEX C3210 as well as some pieces of peripheral equipment (printers, plotter, terminal server) are integrated in the Local Area Network based on the Ethernet standard. Connections with the outside world are granted by the CISCO AGS+ router. In order to protect operational security an uninterruptible power supply (UPS) of the SPECTRUM P3/40kVA type, manufactured by ABB (Switzerland) has been installed.

2.3 Software

CYFRONET – CRACOW works for the needs of a very heterogeneous group of users that represents different scientific disciplines and permanently requests new various software packages. Therefore our important goal is to install a wider range of system and application software.

Our C3210 supercomputer was delivered with the following basic software: ConvexOS, CXBatch, CXdb, CXpa, TCP/IP, NFS, EMACS, FORTRAN, C, PERL, and VECLIB. Later on, according to the needs of our users we installed CXWindows and some programming languages (PASCAL, ACSL, PSL, REDUCE, GNU CC) as well as a significant number of application software packages, both licensed and public domain type. Among them might be mentioned NAG, CERN, BEASY, NISA/EMAG and FITPACK as the most favourable software products of our users. Some others packages like PERMAS or SSPICE will be purchased in the near future.

The SUN workstations are mainly equiped with graphical packages and network services software. Graphical software consists of windowing systems, GKS, XV, XGL, PHIGS and GNUPLOT nowadays. Two of our SUN IPS workstations are used for the purpose of INTERNET. There are Name Server and Mail Server as well as software for a world-wide information exchange service installed on the first SUN. The second one is dedicated to the monitoring of the whole Polish INTERNET and works under the Net Central 1.3 of CISCO. Data are collected through the SNMP protocol.

Both MicroVAX 3100 and VAX 8200 computers are used for networking exclusively. The first VAX works under the VMS 5.5 operating system, the second one under VMS 5.3-1. The only software installed on the MicroVAX is DECnet Full Function (router). Networking software of the VAX 8200 computer contains DECnet End Node (WIGOR), Argonne NJE, EARN End Node, and TCP/IP. This computer replaced our IBM 4381 as the EARN node on September 15th, 1993.

2.4 Users

According to its statute CYFRONET – CRACOW has been working for the needs of the Cracow scientific community, mainly for seven academic schools (the biggest beeing the Jagiellonian University, the University of Mining and Metallurgy, and the Cracow Technical University), six institutes of the Polish Academy of Sciences, and the Nuclear Physics Institute.

Only the most experienced users which perform large scale numerical calculations are using our C3210 supercomputer. On September 1st, 1993 there were 294 CONVEX users except the CYFRONET staff. 48% of them were the Jagiellonian University employees (mainly from the Institute of Physics, Astronomical Observatory, Institute of Chemistry, and Institute of Computer Science) and 33% - those of the University of Mining and Metallurgy (mainly from the Faculty of Electrical Engineering, Automatics and Electronic, Faculty of Metallurgy, Faculty of Physics and Nuclear Techniques, and Faculty of Mining Surveying and Environmental Engineering).

EARN has been our first world-wide computer network in Cracow. The number of EARN users registered at our IBM 4381 computer has exceeded 1,600 in August 1993. In Cracow there are also about 100 EARN users registered for the IBM 4361 of the University of Mining and Metallurgy. The IBM 4381 users represent different range of computer knowledge. Among them are computer scientists as well as people from the Academy of Art, Academy of Music or Theological Seminary. Most of our EARN user are now migrating to INTERNET. We plan to have not more than 100 EARN users registered for our VAX 8200.

The first INTERNET connection was established one year after EARN, but it is INTERNET, the network that became rapidly more and more popular among the academic community of Cracow. CYFRONET has already registered nearly 700 users for the SUN IPC which services INTERNET. The total number of scientists and students who are using INTERNET via their own LANs is difficult to estimate. There are hundreds of users registered for local hosts among the academic schools and research institutes of Cracow. For example, there is a SUN SPARC-server 690MP, more than 50 SUN SPARCstations, over a dozen workstations and servers of DEC, HP, and IBM as well as about 200 PCs interconnected through several LANs at the University of Mining and Metallurgy. Only in the building belonging to the Institute of Physics of the Jagiellonian University there is a SUN SPARCserver 670 MP, more than 30 worksations, and about 200 PCs connected to the network.

2.5 Activities

The main goal of the Academic Computer Centre CYFRONET – CRACOW is to provide high

level computer power and network services for the academic community of Cracow.

The C3210 supercomputer has been very heavily utilized since the first days after its installation. The CPU is overloaded permanently (more than 90 % of CPU time) and there is always a lot of "long" and "very long" jobs in the batch queue. The second C3200 processor first, and the C3820 computer later on are required very urgently.

An average monthly number of INTERNET sessions at our Mail Server equals nearly 7,100 for the last eight months. A monthly analysis of the traffic load for Cracow results in more than 2,500 MB "sent" and more than 5,500 MB "received".

An important area of the CYFRONET activities is to incorporate new ideas and technologies of modern computing into the curriculum of academic schools and research institutes. CYFRONET has been performing different training, seminars and consultation for beginners as well as for advanced users from its very beginning. Lately the Seminar on Supercomputers (1991/92) and the Seminar on Computer Networks (1992/93) have become very popular. They were carried out by qualified CYFRONET staff and by noted specialists from various universities. All seminar participants appreciate the highly professional level of the presentations very much. The first CONVEX Supercomputer Seminar for 70 participants from different academic cities was held in Cracow in 1992. The next one (Meta-Computing) will be organized in November 1993.

Our publishing plays also a remarkable role. We already have prepared more then 200 various training materials in different forms (printed manuals, scripts, computer files). There is also an Information Bulletin published at CYFRONET.

CYFRONET always has been supporting various activities in the area of computing and networking in Cracow. The Council of Rectors has decided to develop a joined automated library system with a central catalogue for scientific libraries of Cracow recently. It will be VTLS system using an existing computer network. CYFRONET representatives are involved in both the project management and the technical team.

Another form of our activities has been the establishing of the local unit of the NASK - Service organization at CYFRONET (NASK = Research and Academic Computer Network). Its goal is to support the users of computer networks (WAN, MAN and LAN) in planning, designing and managing of those networks, installing a networking software, negotiating purchases of telecommunication equipment, traning, etc. An important thing is also monitoring the whole Polish INTERNET traffic load.

A significant event for the CYFRONET activities has also been the conclusion of a service partner agreement with CONVEX. CYFRONET will perform training, hard- and software installations and maintenance of all CONVEX systems in Poland as well as in neighbouring countries in Eastern Europe. Four people of our staff have already been trained in Dallas for those purposes.

We also plan to organize a server of a public domain software for the needs of the Polish academic community. It will decrease the traffic load on international INTERNET connections

and reduce costs related to the simultaneous storing of big files on different disk systems.

2.6 The Cracow Academic Computer Network

The Cracow academic community has carried out its plan of the Metropolitan Area Network generally approved in 1991 and detailed in the early 1992 in a consequent way.

We have built a fiber optic connection with the Jagiellonian University and the University of Mining and Metallurgy. The next one will be with the Cracow University of Technology. The fiber optic cable with several CISCO routers (4 x AGS+, 1 x CGS/3, 4 x IGS/R) forms the backbone of the Cracow Academic Computer Network. The remaining universities and research institutes of Cracow are using telephon lines now but it should be changed into fiber optic in the future. Transmission between Cracow and Warsaw (which has international connections) is performed on the TELBANK radioline (64 Kbps) right now. We are thinking about our own satelite international connection in the near future.

Usage of the CISCO routers enables us to interconnect all TCP/IP, DECnet as well as Novell local networks. To-day's campus network conforms to the Ethernet standard (10 Mbps) and will transform into Metropolitan Area Network based on FDDI ring (100 Mbps) and connecting CYFRONET with three biggest academic schools of Cracow soon. Subsequent fiber optic connections with the remaining institutions in town are related to general plans of establishing the network of fiber opic cables in Cracow by the Polish Telecommunication Ltd. There is also another possibility of using TELBANK connections in Cracow.

CYFRONET has just applied for a grant from the State Committee of Scientific Research. Those funds are needed to achieve the high level of networking comparable with the present world standards which is requested by the Cracow scientists stronger and stronger every day. According to the plan some additional fiber optic sectors should be built and the newest powerful equipment should be bought. The list of necessary purchases contains: HP 9000/800-H70, two CISCO 7000 routers, ASM3-CS Communication Server System, FDDI interfaces and transceivers, ten CISCO IGS/R routers, and two DECconcentrators 500.

We do hope that the future of our computer network in Cracow will be the new ATM technology.

3. Computer Networks in Poland

The history of the computer networks in Poland started in the early eighties. The first experimental computer network linking computers installed at the academic schools in Wrocław, Gliwice and Warsaw, was ready in 1985. The experience achieved during this project was basic for the KASK project (KASK = National Academic Computer Network) which was realized in the late eighties. Both projects have never been very successful. One reason was a lack of high-performance host computers and a very poor standard of telecommunication equipment. Another obstacle was caused by the then existing policital situation. Information transfer between West and East was prohibited in both directions.

The real development of the Polish academic computer network started after our political changes in 1990. On the one hand Poland has joined the main world-wide computer networks like EARN/BITNET and INTERNET as well as the Scandinavian DATAPAK (X25) and the German DFN (X25). On the other hand we can purchase various types of high-performance workstations and servers as well as the newest networking equipment now. Thanks to these new possibilities computer networks in Poland might be characterized by their very dynamic development.

The managment of the Polish academic computer network has been centralized since May 1991. The NASK organization (NASK = Research and Academic Computer Network) was established then. NASK has been guided by the Board of Directors. Other directing bodies of the NASK organization are the Users Council with Special Interest Groups (INTERNET and DECnet), the Technical Director and the Regional Operators. There are two main areas in the NASK organization activities. One is to represent the academic and scientific users of the networks towards the Polish Telecommunication authorities as well as toward the Ministry of National Education and the State Committee of Scientific Research. The other is to take care of the backbone network. The every-day activities of the NASK and the permanent development of the network are mainly financed by the State Committee of Scientific Research.

The NASK Network covers the whole Polish territory and is divided into twelve regions. NASK supports the four main network protocols: X25, BSC/SNA, DECnet and TCP/IP. All Polish Wide Area Networks have a star configuration with the central point in Warsaw. Next Warsaw is linked to Stockholm (satelite connection, 2 Mbps) and Vienna (cable connection, 64 Kbps). Additionally there is also a link between Wrocław and Berlin (cable connection, 9.6 Kbps). Among the interurban connections there are one satelite connection with the speed of 64 Kbps (Toruń), three TELBANK radio connections with the speed of 64 Kbps (Cracow, Wrocław and Poznań) and the remaining regions are linked with the speed of 9.6 Kbps. The local connections inside the cities are mainly cable connections with the speed of 9.6 Kbps but already there also exist some fiber optic connections. Most of the network equipment has been purchased from CISCO and MEMOTEC.

Historically, EARN has been the first "real" computer network in Poland. By now there are 23 Polish EARN nodes in operation and about 10,000 EARN users. Nevertheless INTERNET proved to be the most widespread Polish computer network immediately after its installation. To-day about 2,000 various host computers at the Polish academic schools and research institutes are linked to this network. Vigorous activities of the INTERNET users bring more than 90% of the total traffic load of all NASK networks in Poland.

4. The future of academic computing in Poland and the CONVEX Supercomputer Initiative

Polish academic schools and research institutes use a very heterogenous computer equipment. There has been no common purchase policy in the past. Two CONVEX C3210 and four C120 vector supercomputers as well one IBM 3090 computer (unfortunately without a vector feature) are the biggest computers installed for the needs of the scientific community. The main

part of the computer power is provided by some hundreds of various workstations of SUN, HP, IBM, DEC and Silicon Graphics. Some old IBM and VAX computers are still in operation. Moreover there are also the thousands of PCs.

The rapidly increasing demand for computer power as well as new computing technologies result in a need for significant development. Obviously there are lot of discusions conserning directions of this development. The Polish owner of the "Supercomputing in Central Europe" discussion list has pointed out some typical opinions in a following way:

- We need CRAY supercomputers. (Only CRAY supercomputers are "real" supercomputers.)
- We should rather spend money on CONVEX supercomputers. (They are significantly cheaper than CRAY supercomputers so we can afford it easier.)
- Both CRAY and CONVEX supercomputers should be bought. (For the sake of competitiveness.)
- We don't need any supercomputer. (There is enough supercomputers all over the world. Why don't we use facilities in Los Alamos, NCSA, JPL, CERN etc.)
- We don't need any supercomputer. (Workstations are sufficient for our needs. Some of the workstations outperform cheapest minisupercomputers and they are still much cheaper.)
- Let's spend all of the available money on developing networks.
- Let's have everything: fast networks, workstations and supercomputers. (But it's difficult to be healthy, wealthy and beautiful altogether.)

The end-choice of the solution has two unpleasant conditions. First, there are not sufficient funds for science in our present economic situation and second, the US DOC and COCOM restrictions for the highest technologies still effect us.

Our Ministry of National Education and the State Committee of Scientific Research have decided not to divide their limited budget. That's why they plan to establish three, well equiped supercomputer centres in Cracow, Warsaw and Poznań. They will be linked together by a high-speed computer network. The development of the NASK computer networks has been carried out rather smoothly. The problem is to purchase appropriate supercomputers. "Appropriate" means powerful, not too expensive and with an export license.

Following a request for offers, CONVEX has announced its Supercomputer Initiative for Polish academic schools. CONVEX offered its supercomputers (among them the C3820 supercomputer for Cracow) at a very attractive conditions. CONVEX is also interested in entering a software cooperation with appropriate Polish institutions. In order to improve the hard- and software maintenance of the CONVEX supercomputers already installed in Poland as well as those expected in the near future, CONVEX has signed a service partner agreement with CYFRONET – CRACOW.

CONVEX has already delivered seven supercomputers (4 x C120 and 3 x C3210) to Poland. We are sure that this number will increase very soon. Only our plan contains the second C3200 processor, the C3820 supercomputer and the C120 mini-supercomputer (as mail and file server). The C3820 should be expanded with the 8 x HP 730 Cluster node and transformed into a Meta-Series system later on. The last step of our development project should result in a CONVEX SPP system. Some other academic cities are also interested in CONVEX supercomputers. Nevertheless it is Cracow, the city that claims the title of "CONVEX Town" and that invites all of you to participate in the 94'ECUC conference!

PISCIS
INTERACTIVE PROCESSOR FOR NUCLEAR FUEL MANAGEMENT

Asunción Querol, Antonio Crespo
UITESA Juan Bravo 49 bis, 28006 MADRID (Spain)

Introduction

PISCIS (Processor and Interactive System for Core Intelligent Simulation) is a graphic system developed by UITESA (Spanish consulting and technical services company) to aid in the fuel management and plant operation activities of Nuclear Power Plants (NPP). The primary tool used by PISCIS is a code for BWR (boiling water reactor) neutronic simulation. The actual system runs with the codes PRESTO or SIMULATE-3.

The general idea was to use a well known and qualified simulator to do the calculations and to develop an interactive graphic system (PISCIS) to aid in the result evaluation and input generation for the simulator.

The PISCIS system was designed with three main objectives:

- Decrease the calculation time required to run the simulator and obtaining results.
- Improve the engineer's problem visualization by displaying the simulator results using a graphic screen.
- Improve the program interface by automating the generation of simulator input through an interactive, user's friendly system.

Hardware and software evolution

PISCIS is written in C language and uses Motif libraries for the graphical interface. A large effort has been made since the first version of PISCIS (1989) using SL-OOGMS[1] libraries for the graphic interface, up to the present one using Motif, passing intermediate versions based on DECwindows. Better portability was the clear aim conducting the consecutive versions.

Also the capabilities of the system have been increased with the different versions. The first one included black and white displays, single case analysis, one simulator execution each time, and a reduce number of quarter core maps. The present version manages color displays, fuel shuffling, multiple case analysis (chain cases), simultaneous execution of several cases, curves with combined results, printing of summary tables with user selected variables, increased number of maps in any quarter of the core and optional full-core representation.

The user interface has also evolve to do it more and more user-friendly by the extensive use of pulldown menus and popup windows.

The possibility of a real-time version which processes the data coming from the Data Acquisition System of a Nuclear Power Plant instead the results of a simulator code is being considered for the early future.

Parallel to the graphic evolution of PISCIS, a new effort has been made to create links to different versions of the simulator codes SIMULATE-3 and PRESTO.

---

[1] Sherrill-Lubinski Object Oriented Graphical Modelling System.

The enlarge of PISCIS capabilities was possible in part due to the hardware evolution. The first version of PISCIS run on an old VAXstation in which had not been possible to execute several simulator cases at a time. Even PISCIS and one simulator execution was quite lot so a link was made to run the simulator code on the CONVEX C-120 and PISCIS on the VAXstation, sending and receiving input and output data through the net when required.

The increasing power of hardware at lower cost, and the arrival of X-terminals have come to simplify the situation. Presently, both PISCIS and the simulator code run on the CONVEX computer or any VAX belonging to the UITESA net, and the interface can be displayed at any X-terminal, workstation screen, or PC screen. PISCIS display at Silicon Graphic workstations belonging to the IBERDROLA net is also possible by means of the CISCO router connecting both nets.

Program description

PISCIS creates a graphic environment to monitor and control the executions of the simulator code. output of different calculations are read by PISCIS, which processes and displays the information in coded core maps. User input through the mouse-device and the keyboard is allowed to modify the core management design displayed at the screen (control rod pattern, core flow, fuel shuffling,...) in order to produce a new input file for executing a new case of the simulator code. This processes can be repeated until a case satisfactory to the user is found. Options to monitor the execution and to abort it, if necessary, are available.

The objective of the simulator code is to solve the coupled three dimensional neutron diffusion and two-phase flow equations which govern the steady-state behaviour of a boiling water reactor. The inputs to this program include reactor thermal power, core flow, feedwater temperature and flow rate, control rod patterns and the operating history (exposure, void fraction, control rod positions) of each bundle in the core. The simulator calculates the three dimensional power shape and reactor eigenvalue and updates the bundle operating histories through exposure steps. It also calculates the thermal mechanical limits in the core.

Graphic functions

The basic functions of PISCIS are to display the results of the simulator and interact with the user to modify control rod patterns or refuelling designs. The following graphical information is available to the user:

- General plant operating values like core thermal power, core flow, inlet subcooling, core pressure, core and cycle exposure, core eigenvalue, etc.

- Quarter core or full core maps displaying bundle information: bundle type, bundle identification and bundle exposure.

- General quarter core or full core map displaying the bundles where some thermal limit violation occurs.

- Quarter core or full core maps displaying the following thermal limits:

    - Relative bundle power.
    - Bundle critical power ratio (CPR).
    - Bundle peak linear heat generation rate (MLHGR) and axial node of maximum value.
    - Bundle maximum planar average linear heat generation rate divided by the design limit (MAPRAT) and axial node of maximum value.
    - Bundle infinity multiplication factor (Kinf).

- Bundle preconditioning interim (PCI) data and axial node of maximum value.

- Quarter core or full core maps displaying the control rod positions and the shut down margins (SDM).

- Axial distribution of core average power, exposure and control fraction.

- Axial distribution of bundle power.

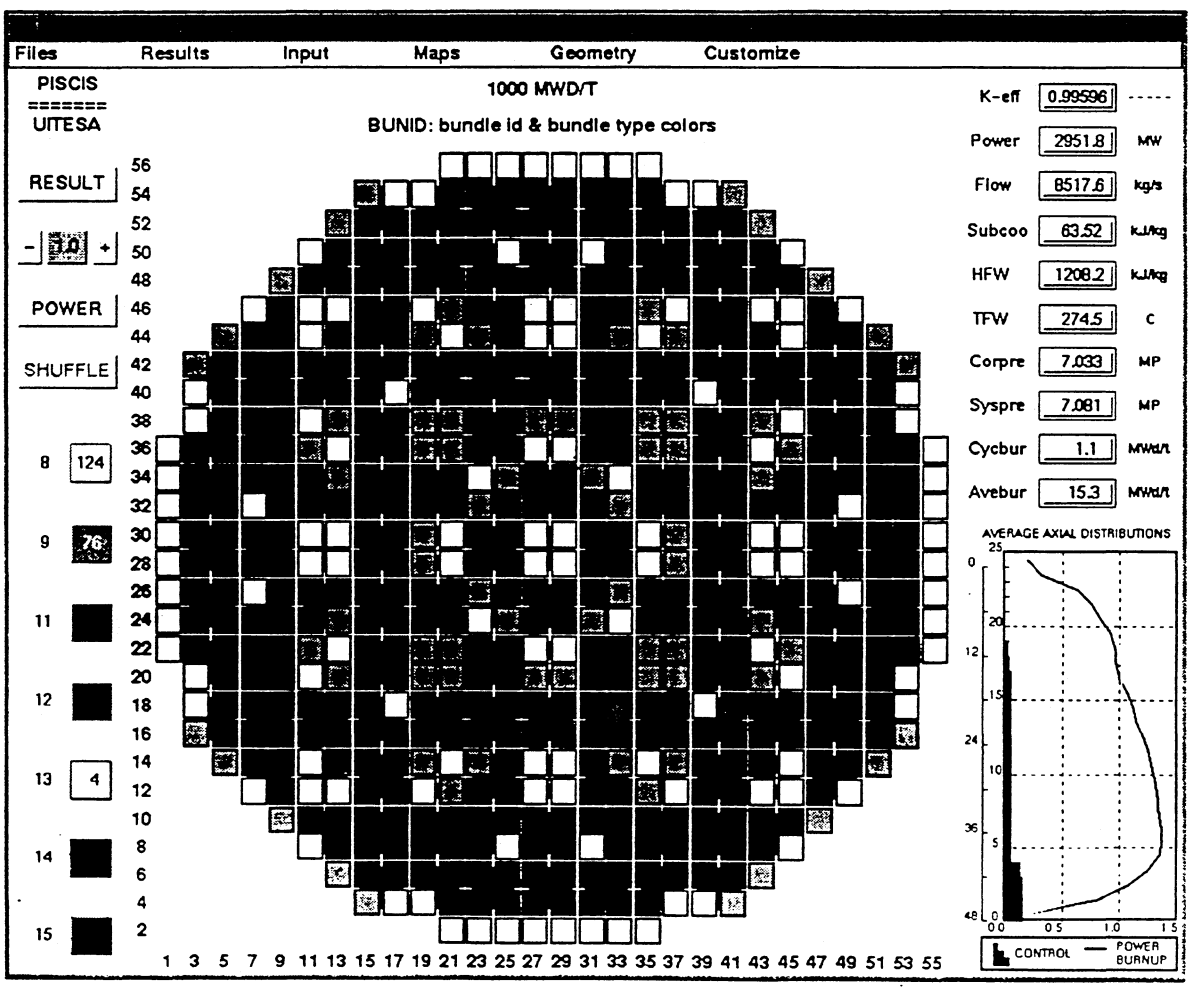
- Summary tables with user selected operating values or simulator input data.

- Graphs representing the evolution with exposure of operating parameters and limiting values for chain cases.

A printout of a typical display is shown in the following figure. This display is obtained after the simulator case has completed and the output file has been requested.

Figure 1

PISCIS full core map with bundle identifications



Interactive functions

PISCIS provides an easy and effective user interface through the use of a "mouse" operated cursor which can be used to generate the control rod position, bundle movement instructions and changes in general data. PISCIS uses this instructions to prepare a new input file and a new execution of the simulator can be ordered by the user. While the case is running, the user can represent previous simulator cases or print out, through a laser printer, the most relevant figures of the case displayed on the screen. All the iterative process is therefore strongly automated trough the use of the PISCIS program.

In the actual scope of the PISCIS program a complete cycle management can be designed and simulated. The applicability of the actual version of PISCIS covers:

- Fuel discharge, load and shuffling.
- Control rod pattern search.
- Core exposure accumulation.
- Haling to end-of-cycle (EOF) projections.
- Shut down margin (SDM) calculations.
- Load following ramps.
- Xenon transients simulation.
- Chain case combining shuffling, Haling, burn-up and SDM cases for cycle design.

Two PISCIS versions, one for engineering design and one for plant operating support are available. In the near future the operating support version will be upgraded with real-time data processing.

It is important to remark that the guidelines used in the PISCIS development are based in extensive previous design and cycle management experience. This ensures the usefulness of the final product.

Conclusions

The interactive system developed provides a very powerful and useful tool to obtain a fast estimate and easy understanding of core performance. Qualified simulator cases run every one or two minutes and results are easily displayed so a complete cycle analysis can be completed in a fraction of the time required before the introduction of interactive methods.

The PISCIS system with SIMULATE-3 as simulator and Motif as graphic interface, is going to be used for cycle management design work of Cofrentes NPP (Spanish BWR) by the IBERDROLA (Spanish utility) engineers. The PISCIS version using PRESTO as simulator and based on DECwindows, is already used at EGL Zurich offices for Leibstadt NPP.

COMP.SYS.CONVEX, sharing Information around the World

or
all
you
wanted
to know
about Usenet
and were afraid to ask...

Name: Jean-Pierre Malisse

Company: MUMM[1]/UGMM[2]/BMM[3]

Address: Gulledelle, 100

B-1200 Brussels

Country: Belgium

Phone: 00 32 (0)2 773 21 36

FAX: 00 32 (0)2 770 69 72

Email: cammejpm@camme.ac.be

This paper consists in an historical overview of the creation of the comp.sys.convex newsgroup.

Introducing first the Usenet concepts and group hierarchy, how it works, how it is managed, what are the "ethics" rules of the Net, this article will present as an applied example the different steps bringing the newsgroup comp.sys.convex up from an idea to a reality.

Besides an official presentation of the Newsgroup Charter, pointing out what were the rationale to promote the newsgroup and explaining in what way this solution will complement the current users mailing list and "icontact" services, a commented historical and statistical report of the voting process describing how in time and space the votes progressively reached our system, how concerned users react and how the votes were processed.

[1]Management Unit of Mathematical Models of the North Sea and Scheldt Estuary
[2]Unité de Gestion du Modèle Mathématique de la mer du Nord et de l'Estuaire de l'Escaut
[3]Beheerseenheid Mathematisch Model Noordzee

1. History of comp.sys.convex

1.1 The pionneers (1990)

Thanks to the mail digests of info-convex collected by Karen Birkelbach (karen@pemrac.space.swri.edu) we were able to find back the names of the first disciples of comp.sys.convex.

The first mail promoting the idea of a comp.sys.convex newsgroup in info-convex originates from David A. Rasmussen (dave@csd4.csd.uwm.edu) in info-convex interest group the 21st of May 1990 :

> I am going to formally ask about it in news.announce.newgroups and comp.sys.super. I guess after a couple week's discussion, one can vote on the groups creation, and eventually the group may or may not get created.

Already enthousiastic people supported the idea like ...

Bill Jones (jones@hermes.chpc.utexas.edu),

> Its about time!
> yes... (it took about to years to come out)

Michael Denisvich (michael@scripps.edu),

> I would/will vote in favor of a comp.sys.convex newsgroup.

Daniel Pommert (daniel@uxh.cso.uiuc.edu),

> I'm in favor of starting the news group comp.sys.convex also but ...

Gustavo Sanchez Gomez (gustavo@cica.es),

> I would vote also in favor of a comp.sys.convex newsgroup.

and eventually Charles Severance (crs@convex.cl.msu.edu)

> I like the idea of a news group.
> 1) How do these things work?
> 2) Can I get a bunch of people here at MSU to vote for the group?
> 3)Who can vote?

the process seemed to be really well engaged, the first two parts of the newsgroup creation requirements were issued but unfortunately the voters were not enough and the newsgroup had to be reported for a new trial.

1.2 The new generation (1992)

A timid discussion we had at the European User Conference in Hamburg in 1991 brought no further progress, and the idea was buried for another year.
Eventually, at the Oxford European user Conference 1992, Volker A. Brand (volker@sfb256.iam.uni-bonn.de), Elisabeth Dregger-Cappel (dregger@convex.rz.uni-duesseldorf.de), and J.-P. Malisse (cammejpm@camme.ac.be) decided this time to suggest the idea more firmly to the committee.
And effectively at the end of the board session the European Convex User Conference Committee 92 decided to support officiously the idea of the newsgroup and designed a "volunteer" :-) in order to organize the vote.

1.3 What USENET is not ?

Thereafter we had to explore Usenet in order to know how to create a newsgroup and we

1. Usenet is not an organization but is well self-organized.
2. Usenet is not a democracy but allows people to express themselves and to gather.
3. Usenet is not fair but is not unfair by the way people usually interact.
4. Usenet is not a right but when you use it you find difficult to miss it.
5. Usenet is not a public utility but helps a lot of people.
6. Usenet is not an academic network but lot of academic people take part to it.
7. Usenet is not an advertising medium but a newsgroup well supported could bring indirect advertisement.
8. Usenet is not the Internet but...
9. Usenet is not a UUCP network but ...
10. Usenet is not a United States network but...
11. Usenet is not a UNIX network but...
12. Usenet is not an ASCII network but...
13. Usenet is not software but...

We discover also that Usenet was self-documented, the only problem was that you usually need to have an access to know to have an access. We will try hereunder to provide pointers which could help you to find more quickly your way through this flowing maze of information.

1.3 Clue dates

The Request for Discussion was published 1st of November 1992 with the closure date of 21st of November 1992.
The Call for Votes began after the day after the Closure day of the Request for Discussion and ended
A first mass acknowledgement was sent to the mailing lists and newsgroups the 3rd of December 1992.
December the 15th 1992, Charles Severance, back from a campaign in Europe, asked his "bunch" of people of MSU to vote for the newsgroup.
The vote was closed on December the 17th of 1992 at 23:59 (GMT)
The comp.sys.convex newsgroup was actually created the 16th of January 1993.

2.USENET structure

2.1 Hierarchy

Newsgroup are organized in a tree structure defining a kind of hierarchy, the mainstream Usenet is divided into seven categories :

COMP
Computer related technical discussion, including also software sources. In general, groups in COMP are expected to have direct value to a site's commercial or academic goals.
Example : comp.sys.convex, comp.sources.misc, comp.binaries.ibm.pc, comp.lang.perl, comp.mail.sendmail

SCI
Discussions marked by special knowledge relating to research in or application of the established sciences.
Example : sci.geo.meteorology

REC
Groups about recreation, hobbies, sports, entertainment, leisure, the arts, etc.
Example : rec.folk-dancing

NEWS
Groups pertaining to USENET related networking.
Example : news.answers, news.announce.newsgroups, news.announce.newusers

SOC
Groups about social issues and social interaction. The humanities, etc. (excluding the recreational arts.)

MISC
Group addressing themes not easily classified into any of the other headings.
Example : misc.jobs.offered

TALK
Groups about topics that often engender heated or emotional debate. Politics, religion, abortion, philosophy, text editors, discrimination etc.

These "world" newsgroups are (usually) circulated around the entire Usenet. European Usenet and EUnet sites take only a selected subset of the more "technical" groups, and controversial "noise" groups are often not carried by many sites in the U.S. and Canada.

EUNET
Groups local to Europe countries
Example : eunet.jokes

Also available are a number of "alternative hierarchies :

ALT
is a collection of newsgroups which are being distributed by a collection of sites that choose to carry them. Many Usenet sites are not interested in these groups. Anything and everything can and does appear; subjects include sex, the Simpsons, and privacy.
Example : alt.sources

GNU
Groups concentrating on interests and software with the GNU Project of the Free Software Foundation.
Example : gnu.gcc.bug

BIZ
The "biz" hierarchy is not part of USENET. The trial.biz hierarchy, however, exists for the creation of commerce related groups.
Example : biz.tadpole.sparcbook

VMSNET, BIONET, CLARI, etc.
These hierarchies are not part of mainstream USENET. They have their own rules for group creation which you must investigate

2.2 Moderated vs Unmoderated

Some newsgroups insist that the discussion remain focused and on-target; to serve this need, moderated groups came to be. All articles posted to a moderated group get mailed first to the group's moderator. He or she peridoically reviews the posts, and either posts them individually to Usenet, or posts a composite digets of the articles for the past day or two. This is how many mailing list gateways work.

3.REQUIREMENTS FOR GROUP CREATION

3.1 Newsgroup creation

The document that describes the current procedure for creating a new newsgroup is entitled "How To Create A New Newsgroup." Its common name, however, is "the guidelines."

If you follow the guidelines, it is probable that your group will be created and will be widely propagated.

HOWEVER: Because of the nature of Usenet, there is no way for any user to enforce the results of a newsgroup vote (or any other decision, for that matter). Therefore, for your new newsgroup to be propagated widely, you must not only follow the letter of the guidelines; you must also follow its spirit. And you must not allow even a whiff of shady dealings or dirty tricks to mar the vote. In other words, don't tick off system administrators; they will get their revenge.

So, you may ask: How is a new user supposed to know anything about the "spirit" of the guidelines? Obviously, he can't. This fact leads inexorably to the following recommendation:

>> If you are a new user, don't try to create a new newsgroup. <<

If you have a good newsgroup idea, then read the "news.groups" newsgroup for a while (six months, at least) to find out how things work. If you're too impatient to wait six months, then you really need to learn; read "news.groups" for a year instead. If you just can't wait, find a Usenet old hand to run the vote for you.

Readers may think this advice unnecessarily strict. Ignore it at your peril. It is embarrassing to speak before learning. It is foolish to jump into a society you don't understand with your mouth open. And it is futile to try to force your will on people who can tune you out with the press of a key.

These are guidelines that have been generally agreed upon across USENET as appropriate for following in the creating of new newsgroups in the "standard" USENET newsgroup hierarchy. They are NOT intended as guidelines for setting USENET policy other than group creations, and they are not intended to apply to "alternate" or local news hierarchies. The part of the namespace affected is comp, news, sci, misc, soc, talk, rec, which are the most widely-distributed areas of the USENET hierarchy.
Any group creation request which follows these guidelines to a successful result should be honoured, and any request which fails to follow these procedures or to obtain a successful result from doing so should be dropped, except under extraordinary circumstances. The reason these are called guidelines and not absolute rules is that it is not possible to predict in advance what "extraordinary circumstances" are or how they might arise.
It should be pointed out here that, as always, the decision whether or not to create a newsgroup on a given machine rests with the administrator of that machine. These guidelines are intended merely as an aid in making those decisions.

3.2 The Discussion (*R*equest *F*or *D*iscussion)

1) A request for discussion on creation of a new newsgroup should be posted to news.announce.newgroups, and also to any other groups or mailing lists at all related to the proposed topic if desired. The group is moderated, and the Follow-up-to : header will be set so that the actual discussion takes place only in news.groups. Users on sites which have difficulty posting to moderated groups may mail submissions intended for news.announce.newgroups to announce-newgroups@rpi.edu.

The article should be cross-posted among the newsgroups, including news.announce.newgroups, rather than posted as separate articles. Note that standard behaviour for posting software is to not present the articles in any groups when cross-posted to a moderated group; the moderator will handle that for you.

2) The name and charter of the proposed group and whether it will be moderated or unmoderated (and if the former, who the moderator(s) will be) should be determined during the discussion period. If there is no general agreement on these points among the proponents of a new group at the end of 30 days of discussion, the discussion should be taken off-line (into mail instead of news.groups) and the proponents should iron out the details among themselves. Once that is done, a new, more specific proposal may be made, going back to step 1) above.

3) Group advocates seeking help in choosing a name to suit the proposed charter, or looking for any other guidance in the creation procedure, can send a message to group-advice@rpi.edu; a few seasoned news administrators are available through this address.

3.3 The *R*equest *F*or *D*iscussion for COMP.SYS.CONVEX

This is the formal Request for Discussion on the creation of a new newsgroup called "comp.sys.convex" (not moderated) who was posted in the News on November 1, 1992.

This announcement was cross-posted to newsgroups whose readers may have interest in the discussion about the new group;

Newsgroups:news.group,news.announce.newgroups,comp.sys.super,comp.lang.perl,comp.graphics.avs,comp.lang.fortran,comp.unix.bsd,comp.unix.misc,eunet.news,sci.physics,sci.geo.fluids,sci.geo.meteorology
Followup-to: news.groups

3.3.1 NEWSGROUP comp.sys.convex (not moderated)

3.3.2 PURPOSE

The newsgroup 'comp.sys.convex' would be a forum for discussing issues related to Convex computer systems hardware & software, common code optimization problems, system administration specific to ConvexOS. It will NOT be a forum for forwarding sensitive information as e-mail and individual contact are for this purpose.

It could be the best, and quickest way to exchange questions and answers, hints and tips on optimization, have contacts between users having common problems and maybe some common solutions.

3.3.3 RATIONALE

Very often Convex users receiving news and having questions don't know really where to post their questions in such a way that it could be answered by some other Convex users. In the same way, people who could be able to answer the question are not necessarily inspecting all articles where Convex is mentioned, such a newsgroup will usually permit better follow-ups for Convex issues. Users could discover lot of indirect Convex matter accidentally in comp.lang.perl (T. Christiansen), sometimes in gnugroups, etc...

Convex User Groups exist in U.S., Europe, and the Pacific Rim; Mailing lists have been going in the U.S and Europe for several years; but this way of communicating is not particularly adequate for users of multi-recipient systems, i.e., each message is sent to each registered user, even on the same systems, moreover each user must ask to be explicitely registered.

A newsgroup would permit all users of a Convex computer receiving the News to receive only one copy of each message and to share it potentially with all the users of the systems.

This newsgroup could be a prolongation of each annual Convex User Groups meetings, permitting not only to those (s)elected people having the chance to take part to these interesting reunions but also to more common users, students, researchers, etc... to share their experiences.
In the era of communication, newsgroups are the easiest way to build bridges across oceans and permitting of people with common interests to be part of an electronic community wherever they are located in the world.

3.3.4 CONTENT

Miscellaneaous
- Convex FAQ
- bi-monthly (?) mailing list report
- vectorisation, parallelisation problems and challenges
- Convex FTP servers
- ECUC, WCUC (conferences)
- jobs available
- other events that concern Convex systems and/or users

Hardware
- Convex processors (GaAs, ...)
- PA/RISC in MPP
- ASAP, MMU, bank conflicts
- Convex backup peripherals
- native/IEEE floating point
- interfacing 3rd party products to Convex system

Software
- Public domain software
- Software releases discussion and announces
- Languages (Fortran/C,...) & optimising compilers
- Application compiler
- Special libraries (scilib, nag, ...)
- Applications
- Convex AVS crosspostings
- Software development

System
- Network integration
- Unix compatibility
- System tuning
- File servers
- Real-Time systems
- Meta MPP
- Benchmarking and performance measurement

System administration
- Backups strategies
- Queue management and accounting
- File systems (striping, mirroring,...)
- Security
- Perl on Convex

This newsgroup will be in conjunction with the existing mailing lists to allow users who do not receive News to submit and receive newsgroup traffic and take part in the discussions.

3.3.5 NAME
comp.sys.convex seems to be the name fitting the most adequately the purpose.

3.3.6 NOT MODERATED
Convex users, belong mostly to research, universities, government,... The number of users being relatively restricted compared to some workstations newsgroup (if you don't know what I mean look at the size of comp.sys.sgi) and in order to have questions answered as fastest as possible, this newsgroup would not need to be moderated and indeed would most probably be self-moderated.

3.3.7 NEWSGROUP CREATION
The discussion period will be from November 1st, 1992 to November 23rd, 1992.
Please post all discussion in news.group.
If a consensus is reached by the end of the discussion period, a CFV (Call for Votes) will be posted at that time. The voting period will last for 21 days.

3.4 The Vote (*C*all *F*or *V*ote)

1) AFTER the discussion period, if it has been determined that a new group is really desired, a name and charter are agreed upon, and it has been determined whether the group will be moderated and if so who will moderate it, a call for votes may be posted to news.announce.newgroups and any other groups or mailing lists that the original request for discussion might have been posted to. There should be minimal delay between the end of the discussion period and the issuing of a call for votes. The call for votes should include clear instructions for how to cast a vote. It must be as clearly explained and as easy to do to cast a vote for creation as against it, and vice-versa. It is explicitly permitted to set up two separate addresses to mail yes and no votes to provided that they are on the same machine, to set up an address different than that the article was posted from to mail votes to, or to just accept replies to the call for votes article, as long as it is clearly and explicitly stated in the call for votes article how to cast a vote. If two addresses are used for a vote, the reply address must process and accept both yes and no votes OR reject them both.

2) The voting period should last for at least 21 days and no more than 31 days, no matter what the preliminary results of the vote are. The exact data that the voting period will end should be stated in the call for votes. Only votes that arrive on the vote-taker's machine prior to this date will be counted.

3) A couple of repeats of the call for votes may be posted during the vote, provided that they contain similar clear, unbiased instructions for casting a vote as the original, and provided that it is really a repeat of the call for votes on the SAME proposal (see #5 below). Partial vote results should NOT be included; only a statement of the specific new group proposal, that a vote is in progress on it, and how to cast a vote. It is permitted to post a "mass acknowledgement" in which all the names of those from whom votes have been received are posted, as long as no indication is made of which may anybody voted until the voting period is officially over.

4) ONLY votes MAILED to the vote-taker will count. Votes posted to the net for any reason (including inability to get mail to the vote-taker) and proxy votes (such as having a mailing list maintainer claim a vote for each member of the list) will not be counted.

5) Votes may not be transferred to other, similar proposals. A vote shall count only for the EXACT proposal that it is a response to. In particular, a vote for or against a newsgroup under one name shall NOT be counted as a vote for or against a newsgroup with a different name or charter, a different moderated/unmoderated status or (if moderated) a different moderator or set of moderators.

6) Votes MUST be explicit; they should be of the form "I vote for the group foo.bar as proposed" or "I vote against the group foo.bar as proposed". The wording doesn't have to be exact, it just needs to be unambiguous. In particular, statements of the form "I would vote for this group if ..." should be considered comments only and not counted as votes.

7) A vote should be run only for a single group proposal. Attempts to create multiple groups should be handled by running multiple parallel votes rather than one vote to create all of the groups.

3.5 The *C*all *F*or *V*ote for COMP.SYS.CONVEX

This call for vote was emitted the 1st of November 1992.
A second Call for Vote accompanied by a global acknowledgement of votes received at that moment was emitted the 3rd of December 1992.

Subject: CFV: comp.sys.convex
Newsgroups:news.announce.newgroups,news.groups,comp.sys.super,comp.lang.perl,comp.graphics.avs, comp.lang.fortran,comp.unix.bsd,comp.unix.misc,eunet.news,sci.physics,sci.geo.fluids,sci.geo.meteorology
Keywords: convex,supercomputer,newsgroup,fortran,vector,parallel,unix,BSD

This is a formal Call For Vote on the creation of a new newsgroup called "comp.sys.convex" (not moderated).
The RFD for this group was originated by Jean-Pierre Malisse (cammejpm@camme.ac.be) and was posted in news.announce.newgroups on October 29th, 1992. Discussion in news.groups, in convex mailing list, and personal info requests received since that time has been unanismously favorable, and no significant objections to the group's name or charter have been raised.

NEWSGROUP comp.sys.convex

STATUS: Unmoderated

RATIONALE: (discussed in RFD)

3.5.1 CHARTER:

The newsgroup 'comp.sys.convex' is a forum for discussing issues related to Convex computer systems hardware & software, common code optimization problems, system administration specific to ConvexOS. It will NOT be a forum for forwarding sensitive information as e-mail and individual contact are for this purpose.

This includes discussion of Hardware, Software, System, System Administration topics related to Convex computers.

This newsgroup will be in conjunction with the existing mailing lists to allow users who do not receive News to submit and receive newsgroup traffic and take part in the discussions.

3.5.2 CONTENT:

Convex FAQ, mailing list report, vectorisation, parallelisation,
Convex FTP servers, ECUC, WCUC (conferences),
Hardware, Software, System, and System Administration
related to Convex supercomputer (cf previous RFD for more details)

Voting Instructions and addresses:

"Yes" votes (in support of the formation of comp.sys.convex as described) should be sent to
yes@camme.ac.be

"No" votes (in opposition to the formation of comp.sys.convex as described) should be sent to
no@camme.ac.be

If you want the group 'comp.sys.convex' to be created, include the text:
"I vote YES for the creation of comp.sys.convex."

If you feel the group 'comp.sys.convex' should not be created, include the text:
"I vote NO for the creation of comp.sys.convex."

Variations in subject line format and message body will be accepted as long as the intent is clearly expressed, and does not conflict with the vote address used.

Voting Rules:

- One vote per person. If you vote more than once, only the *last* of those votes will count. Therefore if you change your mind about the vote, you can send a new vote, which invalidates the old vote,as long as it is within the voting period.

- Ambiguous votes will be invalidated. Ambiguous votes contain neither YES nor NO, or contain both, or are conditional, such as "I vote ... if ..."

- Votes posted to any newsgroup will NOT be counted.

- According to the USENET rules, in order for the new group to be created, at least 2/3 of the votes must be "YES", and there must be at least 100 more "YES" votes than "NO" votes.

- For more details on the voting rules being followed, refer to article <spaf-howto_715578991@cs.purdue.edu> in news.announce.newgroups.

Ambiguous vote resolution:

Ambiguous votes -- those whose subject lines or message body do not make clear the voter's intent, or whose vote clashes with the vote address -- sent to either the vote addresses or the reply address will, where possible, be returned to their senders for clarification. Ambiguous votes which can not by returned to their senders or for which no clarification is provided will be identified in vote mass acknowledgements and in the final vote tally.

Voting period:

Votes will be collected for a period beginning when this CFV is approved for publication in news.announce.newgroups and ending on Thursday December 17, 1992, at 23:59 GMT.
Votes received after the expiration of the voting period will be discarded.

You may change your vote at any time before the expiry of the voting period by simply submitting a new vote. The last vote to be cast by any given individual before the end of the voting period will be counted; earlier votes cast by that individual will be discarded.

And remember, it's one person, one vote.

3.6 The Result

1) At the completion of the voting period, the vote taker must post the vote tally and the E-mail addresses and (if available) names of the votes received to news.announce.newgroups and any other groups or mailing lists to which the original call for votes was posted. The tally should include a statement of which way each voter voted so that the results can be verified.

2) AFTER the vote result is posted, there will be a 5 day waiting period, beginning when the voting results actually appear in news.announce.newgroups, during which the net will have a chance to correct any errors in the voter list or the voting procedure.

3) AFTER the waiting period, and if there were no serious objections that might invalidate the vote, and if 100 more valid YES/create votes are received that NO/don't create AND at least 2/3 of the total number of valid votes received are in favour of creation, a newgroup control message may be sent out. In the 100 vote margin or 2/3 percentage is not met, the group should not be created.

4) The newgroup message will be sent by the news.announce.newgroups moderator at the end of the waiting period of a successful vote. If the new group is moderated, the vote-taker should send a message during the waiting period to Gene Spafford <spaf@cs.purdue.edu> and David C. Lawrence <tale@rpi.edu> with both the moderator's contact address and the group's submission address.

5) A proposal which has failed under point (3) above should not again be brought up for discussion until at least six months have passed from the close of the vote. This limitation does not apply to proposals which never went to vote.

3.7 The Results of the COMP.SYS.CONVEX Vote

At the end of the voting period (11:59 pm, December 17, 1992), 299 people had sent in their ballots in response to the CFV.

The proposal passed with a broad margin : 292 YES and 7 NO votes were cast.
this represents 97.7 % of YES votes
2.3 % of NO votes

As usual there will be a five day waiting period before the group comes to existence. It might also take some day more before the gateway with the user mailing list will be set up,
Karen Birchelbach is alledgedly collecting information to make it work.
The newyear holidays would maybe slow down lightly the process.

Newgroup control message(s) were sent out by David Lawrence, the moderator of news.announce.newgroups, after the 5-day waiting period and the 15 days fishing period (Christmas holidays)

I would like to express my gratitude to each and every one of you whose participation has made this CFV a tremendous success

	First estimation (Dec 18th, 1992)		New PERL version of tallying program (1993)[4]
Number of voters	299	Number of voters	299
Number of Yes votes:	292/299	Number of Yes votes:	277/299
Number of No votes:	7/299	Number of No votes:	7/299
		Number of late Yes votes :	15/299

[4]The old version of the vote processing program was not aware of the GMT restriction of the end of vote period

3.7 Why have you voted NO ?

We also intend a little survey of the motivations for the NO-voters :
To the question
"What were the motivations which lead you to express a negative vote (4 answers over 7)" they replied :

> I vote no on every newsgroup proposed which I don't actually intend to read and use; it's my way of trying to keep my .newsrc small(er)
>
> I prefer seeing this info. by mail because newsgroups are an inherent chaotic medium in which it is possible to miss things. If I go on holiday for a week, some news items will expire. Also, I often skip newsgroup for weeks at a time if I'm busy, whereas I always read all my mail.
>
> Just another forum to follow... already following info.conv why do we also need comp.sys.convex?
>
> I am not against the group. My vote was meant to say that I do not have any personal interest in such a group and I have no plans to read/participate in this news group.

3.8 Readership

According to the USENET Readership Summary Report of February 1993, a monthly posting in news.lists from the Network & Measurement Project of the DEC Network System Laboratory in Palo Alto, California, a survey lead by Brian Reid (reid@decwrl.dec.com) based on data taken from various USENET sites and extrapoled to the whole USENET

	Sample	Extrapolation to the Entire Usenet
Number of Sites	708	69000
	1.80%	100%
Number of Users	174208	7381000
Number of Netreaders	47019	1992000

comp.sys.convex has for February 1993 the rank 1380 over a total range of 1882 newgroups.
47% of the USENET sites are supposed to receive the newsgroup.
The effective number of readers in the sample was 214 and an extrapolated number of 8200.
Activity in february was 24 messages, last check in comp.sys.newsgroup indicated an index of 67 since the creation date.

3.9 Possible extensions

- Automatic bi-directional gateway with the existing info-convex mailing list.
- Archiving of monthly comp.sys.convex digests on FTP servers (already done on the german FTP-server).
- When comp.sys.convex will be grown up a FAQ for comp.sys.convex could be settled.

3.10 References and Acknowledgements

Brendan P. Kehoe, "Zen and the Art of the Internet [a Beginner's Guide to the Internet]",
Additional pointers for information "around the World", a Postscript version of the earliest version was available on many ftp sites.

G. Todino/Dougherty," Using UUCP and Usenet", O'Reilly & Associates Inc, 1989
O'Reilly/Todino, "Managing UUCP and Usenet", O'Reilly & Associates Inc, 1989

pit-manager.mit.edu (18.72.2.1) is a MIT machine graciously dedicated to the archiving and the storage of the various periodic postings that are peppered throughout the various Usenet groups. Look in the directory /pub/usenet

The following history and descriptions of Usenet and newsreader software have been reproduced from the document "usenet-software" and "what-is-usenet (original from Chip Salzenberg)"
with the permission of

Mr Gene Spafford,
Software Engineering Research Center & Dept. of Computer Sciences
Purdue University, W. Lafayette IN 47907-1398
Internet: spaf@cs.purdue.edu phone: (317) 494-7825

3.11 Miscellaneous pointers

A group of people at Mc Gill University in Canada got together and created a query system called archie. It was originally formed to be a quick and easy way to scan the offerings of the many FTP anonymous FTP sites that are maintained around the world.

archie.ans.net	147.225.1.2	(New York, USA)
archie.rutgers.edu	128.6.18.15	(New Jersey, USA)
archie.sura.net	128.167.254.179	(Maryland, USA)
archie.unl.edu	129.93.1.14	(Nebraska, USA)
archie.mcgill.ca	132.206.44.21	(Canada)
archie.funet.fi	128.214.6.102	(Finland)
archie.doc.ic.ac.uk	146.169.11.3	(United Kingdom)
archie.au	139.130.4.6	(Australia)

enter archie to log in at the login prompt.

freenet-in-a.cwru.edu (129.22.8.38) is an open-access, free, community computer system, this machine is part of the Cleveland Freenet, sponsored by CWRU (Case Western Reserve University).
Cleveland Freenet is a project of National Telecomputing Public Network and offers a wide variety of communication services from mail, chitchat, and news access. There is no charge for the registration process and no charge for use of the system.

3.12 Convex FTP-servers

info-convex@pemrac.space.swri.edu	129.162.155.100	USA
cug@convex.rz.uni-duesseldorf.de	134.129.128.3	Germany
		United Kingdom

The complete source of the following documents as well as last updated versions are available on the ftp-server pit-manager.mit.edu [18.172.1.27]. Archived files and FAQ files are all in the directory /pub/usenet/news.answers.
The information is available by mail, send an e-mail to mail-server@pit-manager.mit.edu containing:

send usenet/news.answers/TITLE/PART

where TITLE is the archive title and PART is the portion of the posting you want.

Send a message containing "help" to get general information about the mail server, including information on how to get a list of archive titles to use in further commands.

4. What is USENET ?

The first thing to understand about Usenet is that it is widely misunderstood. Every day on Usenet, the "blind men and the elephant"phenomenon is evident, in spades. In my opinion, more flame wars arise because of a lack of understanding of the nature of Usenet than from any other source. And consider that such flame wars arise, of necessity, among people who are on Usenet. Imagine, then, how poorly understood Usenet must be by those outside!

Any essay on the nature of Usenet cannot ignore the erroneous impressions held by many Usenet users. Therefore, this article will treat falsehoods first. Keep reading for truth. (Beauty, alas, is not relevant to Usenet.)

4.1 What USENET is not

1. Usenet is not an organization.

No person or group has authority over Usenet as a whole. No one controls who gets a news feed, which articles are propagated where, who can post articles, or anything else. There is no "Usenet Incorporated," nor is there a "Usenet User's Group." You're on your own.

Granted, there are various activities organized by means of Usenet newsgroups. The newsgroup creation process is one such activity. But it would be a mistake to equate Usenet with the organized activities it makes possible. If they were to stop tomorrow, Usenet would go on without them.

2. Usenet is not a democracy.

Since there is no person or group in charge of Usenet as a whole -- i.e. there is no Usenet "government" -- it follows that Usenet cannot be a democracy, autocracy, or any other kind of "-acy." (But see "The Camel's Nose?" below.)

3. Usenet is not fair.

After all, who shall decide what's fair? For that matter, if someone is behaving unfairly, who's going to stop him? Neither you nor I, that's certain.

4. Usenet is not a right.

Some people misunderstand their local right of "freedom of speech" to mean that they have a legal right to use others' computers to say what they wish in whatever way they wish, and the owners of said computers have no right to stop them.

Those people are wrong. Freedom of speech also means freedom not to speak. If I choose not to use my computer to aid your speech, that is my right. Freedom of the press belongs to those who own one.

5. Usenet is not a public utility.

Some Usenet sites are publicly funded or subsidized. Most of them, by plain count, are not. There is no government monopoly on Usenet, and little or no government control.

6. Usenet is not an academic network.

It is no surprise that many Usenet sites are universities, research labs or other academic institutions. Usenet originated with a link between two universities, and the exchange of ideas and information is what such institutions are all about. But the passage of years has changed Usenet's character. Today, by plain count, most Usenet sites are commercial entities.

7. Usenet is not an advertising medium.

Because of Usenet's roots in academia, and because Usenet depends so heavily on cooperation (sometimes among competitors), custom dictates that advertising be kept to a minimum. It is tolerated if it is infrequent, informative, and low-hype.

The "comp.newprod" newsgroup is NOT an exception to this rule: product announcements are screened by a moderator in an attempt to keep the hype-to-information ratio in check.

If you must engage in flackery for your company, use the "biz" hierarchy, which is explicitly "advertising-allowed", and which (like all of Usenet) is carried only by those sites that want it.

8. Usenet is not the Internet.

The Internet is a wide-ranging network, parts of which are subsidized by various governments. It carries many kinds of traffic, of which Usenet is only one. And the Internet is only one of the various networks carrying Usenet traffic.

9. Usenet is not a UUCP network.

UUCP is a protocol (actually a "protocol suite," but that's a technical quibble) for sending data over point-to-point connections, typically using dialup modems. Sites use UUCP to carry many kinds of traffic, of which Usenet is only one. And UUCP is only one of the various transports carrying Usenet traffic.

10. Usenet is not a United States network.

It is true that Usenet originated in the United States, and the fastest growth in Usenet sites has been there. Nowadays, however, Usenet extends worldwide.

The heaviest concentrations of Usenet sites outside the U.S. seem to be in Canada, Europe, Australia and Japan.

Keep Usenet's worldwide nature in mind when you post articles. Even those who can read your language may have a culture wildly different from yours. When your words are read, they might not mean what you think they mean.

11. Usenet is not a UNIX network.

Don't assume that everyone is using "rn" on a UNIX machine. Among the systems used to read and post to Usenet are Vaxen running VMS, IBM mainframes, Amigas, and MS-DOS PCs.

12. Usenet is not an ASCII network.

The A in ASCII stands for "American". Sites in other countries often use character sets better suited to their language(s) of choice; such are typically, though not always, supersets of ASCII. Even in the United States, ASCII is not universally used: IBM mainframes use (shudder) EBCDIC. Ignore non-ASCII sites if you like, but they exist.

13. Usenet is not software.

There are dozens of software packages used at various sites to transport and read Usenet articles. So no one program or package can be called "the Usenet software."

Software designed to support Usenet traffic can be (and is) use for other kinds of communication, usually without risk of mixing the two. Such private communication networks are typically kept distinct from Usenet by the invention of newsgroup names different from the universally-recognized ones.

Well, enough negativity.

4.2 What USENET is

Usenet is the set of people who exchange articles tagged with one or more universally-recognized labels, called "newsgroups" (or "groups" for short).

(Note that the term "newsgroup" is correct, while "area," "base," "board," "bboard," "conference," "round table," "SIG," etc. are incorrect. If you want to be understood, be accurate.)

4.3 Diversity

If the above definition of Usenet sounds vague, that's because it is.

It is almost impossible to generalize over all Usenet sites in any non-trivial way. Usenet encompasses government agencies, large universities, high schools, businesses of all sizes, home computers of all descriptions, etc, etc.

(In response to the above paragraphs, it has been written that there is nothing vague about a network that carries megabytes of traffic per day. I agree. But at the fringes of Usenet, traffic is not so heavy. In the shadowy world of news-mail gateways and mailing lists, the line between Usenet and not-Usenet becomes very hard to draw.)

4.4 Control

Every administrator controls his own site. No one has any real control over any site but his own.

The administrator gets her power from the owner of the system she administers. As long as her job performance pleases the owner, she can do whatever she pleases, up to and including cutting off Usenet entirely. Them's the breaks.

Sites are not entirely without influence on their neighbours, however. There is a vague notion of "upstream" and "downstream" related to the direction of high-volume news flow. To the extent that "upstream sites decide what traffic they will carry for their "downstream"
neighbors, those "upstream" sites have some influence on their neighbors' participation in Usenet. But such influence is usually easy to circumvent; and heavy-handed manipulation typically results in a backlash of resentment.

4.5 Periodic Postings

To help hold Usenet together, various articles (including this one) are periodically posted in newsgroups in the "news" hierarchy. These articles are provided as a public service by various volunteers. They are few but valuable. Learn them well.

Among the periodic postings are lists of active newsgroups, both "standard" (for lack of a better term) and "alternative." These lists, maintained by Gene Spafford, reflect his personal view of Usenet, and as such are not "official" in any sense of the word. However, if you're looking for a description of subjects discussed on Usenet, or if you're starting up a new Usenet site, Gene's lists are an eminently reasonable place to start.

4.6 Propagation

In the old days, when UUCP over long-distance dialup lines was the dominant means of article transmission, a few well-connected sites had real influence in determining which newsgroups would be carried where. Those sites called themselves "the backbone."

But things have changed. Nowadays, even the smallest Internet site has connectivity the likes of which the backbone admin of yesteryear could only dream. In addition, in the U.S., the advent of cheaper long-distance calls and high-speed modems has made long-distance Usenet feeds thinkable for smaller companies.

There is only one pre-eminent site for UUCP transport of Usenet in the U.S., namely UUNET. But UUNET isn't a player in the propagation wars, because it never refuses any traffic. UUNET charges by the minute, after all; and besides, to refuse based on content might jeopardize its legal status as an enhanced service provider.

All of the above applies to the U..S. In Europe, different structures favored the creation of strictly controlled hierarchical organizations with central registries. This is all very unlike the traditional mode of U.S. sites (pick a name, get the software, get a feed, you're on). Europe's "benign monopolies," long uncontested, now face competition from looser organizations patterned after the U.S. model.

4.7 The camel's nose ?

As was observed above in "What Usenet Is Not," Usenet as a whole is not a democracy. However, there is exactly one feature of Usenet that has a form of democracy: newsgroup creation.

A new newsgroup is unlikely to be widely propagated unless its sponsor follows the newsgroup creation guidelines; and the current guidelines require a new newsgroup to pass an open vote.

There are those who consider the newsgroup creation process to be a remarkably powerful form of democracy, since without any coercion, its decisions are almost always carried out. In their view, the democratic aspect of newsgroup creation is the precursor to an organized and democratic Usenet Of The Future.

On the other hand, some consider the democratic aspect of the newsgroup creation process a sham and a fraud, since there is no power of enforcement behind its decisions, and since there appears little likelihood that any such power of enforcement will ever be given it. For them, the appearance of democracy is only a tool used to keep proponents of flawed newsgroup proposals from complaining about their losses.

So, is Usenet on its way to full democracy? Or will property rights and mistrust of central authority win the day? Beats me.

4.8 If you are unhappy...
Property rights being what they are, there is no higher authority on Usenet than the people who own the machines on which Usenet traffic is carried. If the owner of the machine you use says, "We will not carry alt.sex on this machine," and you are not happy with that order, you have no Usenet recourse. What can we outsiders do, after all?

That doesn't mean you are without options. Depending on the nature of your site, you may have some internal political recourse. Or you might find external pressure helpful. Or, with a minimal investment, you can get a feed of your own from somewhere else. Computers capable of taking Usenet feeds are down in the $500 range now, and UNIX-capable boxes are going for under $2000, and there are at least two UNIX lookalikes in the $100 price range.

No matter what, though, appealing to "Usenet" won't help. Even if those who read such an appeal are sympathetic to your cause, they will almost certainly have even less influence at your site than you do.

By the same token, if you don't like what some user at another site is doing, only the administrator and owner of that site have any authority to do anything about it. Persuade them that the user in question is a problem for them, and they might do something -- if they feel like it, that is.

If the user in question is the administrator or owner of the site from which she posts, forget it; you can't win. If you can, arrange for your newsreading software to ignore articles from her; and chalk one up to experience.

5. What need to be known about USENET

5.1 History
Usenet came into being in late 1979, shortly after the release of V7 Unix with UUCP. Two Duke University grad students in North Carolina, Tom Truscott and Jim Ellis, thought of hooking computers together to exchange information with the Unix community. Steve Bellovin, a grad student at the University of North Carolina, put together the first version of the news softwre using shell scripts and installed it on the first two sites: "unc" and "duke". At the beginning of the 1980 the network consisted of those two sites and "phs" (another machine at Duke), and was described at the January Usenix Conference. Steve Bellovin later rewrote the scripts into C programs, but they were never release beyond "unc" and "duke". Shortly thereafter, Steve Daniel did another implementation in C for public distribution. Tom Truscott made further modifications, and this became the "A" news release.

In 1981 at U.C. Berkeley, grad student Mark Horton and high school student Matt Glickman rewrote the news software to add functionality and to cope with the ever increasing volume of news -- "A" news was intended for only a few articles per group per day. This rewrite was the "B" news version. The first public release was version 2.1 in 1982; the 1.* versions were all beta test. As the net grew, the news software was expanded and modified. The last version maintained and released primarily by Mark was 2.10.1

Rick Adams, at the Center for Seismic Studies, took over coordination of the maintenance and enhancement of the news software with 2.10.2 release in 1984. By this time, the increasing volume of news was becoming a concern, and the mechanism for moderated groups was added to the software at 2.10.2. Moderated groups were inspired by ARPA mailing lists and experience with other bulletin board systems. In late 1986, version 2.11 of news was released, including a number of changes to support a new naming structure for newsgroups, enhanced batching and compression, enhanced ihave/sendme control messages, and other features.

5.2 News propagation and management software

The current release of news is 2.11, patchlevel 19. Article format is specified in RFC 1036 (see below). B news versions are considered "obsolete." The various maintainers of the Usenet software have never expressed any commitment to maintain backwards compatibility with "foreign" or obsolete news systems and are unlikely to do so; it is the responsibility of the users to maintain compatibility of such software if they wish to continue to interact with the Usenet.
B News has been declared "dead" by a number of people, including Rick Adams, and is unlikely to be upgraded further; most new UUCP sites are using C News (see next paragraph).

A new version of news, becoming known as "C" news, has been developed at the University of Toronto by Geoff Collyer and Henry Spencer. This version is a rewrite of the lowest levels of news to increase article processing speed, decrease article expiration processing and improve the reliability of the news system through better locking, etc. The package was released to the net in the autumn of 1987. For more information, see the paper "News Need Not Be Slow", published in the Winter 1987 Usenix Technical Conference proceedings. The most recent version of C News is the 2 May 1992 "performance release." C News can be obtained from its official archive site, cs.toronto.edu (128.100.1.105), using FTP.

Another Usenet system, known as InterNetNews, or INN, was written by Rich Salz (rsalz@uunet.uu.net). INN is designed to run on Unix hosts that have a socket interface. It is optimized for larger hosts where most traffic uses NNTP, but it does provide full UUCP support. INN is very fast, and since it integrates NNTP many people find it easier to administer only one package. The package was publicly released on August 20, 1992. For more information, see the paper "InterNetNews: Usenet Transport for Internet Sites" published in the June 1992 Usenix Technical Conference Proceedings. INN can be obtained from many places; its official archive site is ftp.uu.net (192.48.96.9) in the directory networking/news/nntp/inn.

In March 1986 a package was released implementing news transmission, posting and reading using the Network News Transfer Protocol (NNTP) (as specified in RFC 977). This protocol allows hosts to exchange article via TCP/IP connections rather than using the traditional uucp. It also permits users to read and post news (using a modified version of "rn" or other user agents) from machines which cannot or choose not to install the USENET news software. Reading and posting are done using TCP/IP messages to a server host which does run the USENET software. Sites which have many workstations products find this a convenient way to allow workstation users to read news without having to store articles on each system. Many of the Usenet hosts that are also on the Internet exchange news articles using NNTP because the load impact of NNTP is much lower than uucp (and NNTP ensures much faster propagation).

NNTP grew out of independent work in 1984-1985 by Brian Kantor at U.C. San Diego and Phil Lapsley at U.C. Berkeley. It is now in release 1.5.11 dated 11 february 1991, with the next planned release at 1.6 NNTP includes support for System V UNIX and DECNET under Ultrix. NNTP was developed at U.C. Berkeley by Phil Lapsley with help from Erik Fair, Steven Grady, and Mike Meyer, among others. The NNTP package is distributed on the 4.3BSD release tape (although that is version 1.2a and out-of-date) and is also available from the various authors, many major hosts, and by anonymous FTP from lib.tmc.edu [129.106.5.1], ftp.uu.net[192.48.96.9].

5.3 Newsreader software

Several popular screen-oriented news reading interfaces have been developed in the last few years to replace the traditional "readnews" interface. The first of these was "vnews" and it was written by Kenneth Almquist. "vnews" provides a "readnews"-like command interface but displays articles using direct screen positioning. It appears to have been inspired , to some extent by the "notes" system. "vnews" is currently distributed with the standard 2.11 news source.

A second more versatile interface, "rn", was developed by Larry Wall (the author of Perl, patch, metaconfig) now of Netlabs (ftp.netlabs.com 192.94.48.152) and release in 1984. This interface also uses full-screen display with direct positioning, but it includes many other useful features and is very popular with many net readers. The interface includes reading, discarding, and/or processing of articles based on user-definable patterns, the ability to follow "threads of discussions" in newsgroup, and the ability of the user to develop customized macros for display and keyboard interaction. "rn" is currently at release 4.4. It is being maintained by Stan Barber of the Baylor College of Medicine. "rn" is not provided with the standard news software release, but is very widely available because of its popularity. The software can be obtained from its official archive site, lib.tmc.edu (129.106.5.1), using FTP, and via mail from archive-server@bcm.tmc.edu, and via mail from archive-server@bcm.tmc.edu.

A variant of "rn" is "trn" by Wayne Davison. Trn adds the ability to follow "threads of discussions" in newsgroups; its latest version 2.2 is based on rn 4.4. It uses a Reference-line database to allow the user to take advantage of the "discussion tree" formed by an article and its replies. This results in a true reply-ordered reading of the articles, complete with a small ascii representation of the current article's position in the discussion tree. Trn can be obtained from ftp.coe.montana.edu (192.31.215.240) in the /pub/trn directory, from uunet in the news subdirectory, and from many other archive servers world-wide.

xrn is an X11-based interface to NNTP that was written by Rick Spickelmier ands Ellen Sentovich (UC Berkeley). The current version is 6.17. xrn supports many features, including sorting by subject, user-settable key bindings, graceful handling of NNTP server crashes, and many of the features of rn (including KILL files and key bindings similar to rn). xrn is actively supported by the authors with bug fixing and feature addition support from many of the users. xrn can be retrieved from most of the popular FTP sites (gatekeeper.dec.com [16.1.0.2], ftp.uu.net[192.48.96.9], export.lcs.mit.edu [18.24.0.12]).

Another X11-based newsreader is xvnews. This is a news reader designed primarily for Sun workstations running OpenWindows. It runs with NNTP and is compatible with rn style commands. It is available from export.lcs.mit.edu (18.24.0.12) in the contrib directory.

There are two popular macro packages named "GNUS" and "Gnews" that can be used with the GNU Emacs text editor. These allow reading, replying, and posting interaction with the news from inside Emacs text editor. Client code exists to get the articles using NNTP rather than a local disk. Copies can be found on most archive sites that carry the GNU archives.

"nn" is yet another reader interface, developed by Kim F. Storm of Texas Instruments A/S, Denmark, and released in 1989. nn differs from the traditional readnews and vnews by presenting a menu of article subject and sender-name lines, allowing you to preselect articles to read. nn is also a very fast newsreader, as it keeps a database of article headers on-line (I.e. it trades space for time. A good rule of thumb is that the nn database size is 5%-10% of your news spool. So up to 110% of your news spool is the amount of space news and the nn database will take). The current version of nn is 6.4.16 nn can be obtained via anonymous FTP from dkuug.dk (193.88.44.141), uop.uop.edu (138.9.200.1), or various other sites;

Yet another newsreader is the "tin" reader. It operates with threads, has different article organization methods, and is full-screen oriented. tin works on a local news spool or over an NNTP connection. It has been posted to alt.sources, and further information is available from Iain Lea (iain%anl433.uucp@Germany.EU.net). The current release of tin is 1.1 PL6. Tin is based more on the Notes and tass systems than "rn". There is an extensive list of features, including interfaces to batch modes and auto unpacking mechanisms.

5.4 Software versions and availability

You can obtain the version number of your news software by issuing some form of "v" command to show the current version -- consult the man page for details. Current software is obtainable from almost any major Usenet site as well as the sites noted in the body of the article, above.

The following sites have allegedly sources to the current news software available for anyone needing a copy :

Site	Contact
munnari	usenet@munnari.oz.au
osu-cis	ostmaster@tut.cis.ohio-state.edu
philabs	usenet@philabs.philips.com
pyramid	usenet@pyramid.com
rutgers	usenet@rutgers.edu
tektronix	news@tektronix.tek.com
watmath	usenet@watmath.waterloo.edu

Sources for news 2.11, C News, "rn", and "trn" are also available in the comp.sources.unix archives.

5.5 Standards

News program communicate with each other according to standard protocols, some of which are described by RFC's. An RFC is a Request For Comment, a de facto standard in the Internet Community. It is a form of published software standard in the Internet Community. It is a form published software standard , done through the Network Information Center (NIC) at SRI. Copies of RFCs are often posted to the net and obtainable from archive sites.
Current news-related RFCs include the following :

RFC 822	specifies the format of messages;
RFC 977	specifies NNTP, the Network News Transfer Protocol
RFC 1036	specifies the format of Usenet articles
RFC 1123	amends RFC 822
RFC 1153	specifies the digest format some moderated groups use.

There is also a subset of RFCs called FYIs (For your Information); they are written in a language much more informal than taht used in the other, standard RFC's. Topics range from answers to common questions fro new and experienced users to a suggested bibliography.
Both RFC's and FYI's are available by FTP from nic.ddn.mil (192.112.36.5) in rfc/rfc-xxxx.txt and ftp.uu.net ftp.uu.net (192.48.96.9) in the directory /RFC.

The preceding history and descriptions of Usenet and newsreader software have been reproduced from the document "usenet-software" and "what-is-usenet (original from Chip Salzenberg)"
with the permission of
Mr Gene Spafford,
Software Engineering Research Center & Dept. of Computer Sciences
Purdue University, W. Lafayette IN 47907-1398
Internet: spaf@cs.purdue.edu phone: (317) 494-7825

EXPERIENCE OF THE INSTALLATION AND OPTIMIZATION OF THE VICTORIA CODE

Harald Funke, Manfred Kunicke
Forschungszentrum Rossendorf e.V., Dresden, Germany

VICTORIA-92 - Code Scope and Models

The consequences of a severe reactor accident depend on the quantity, characteristics and timing of the release of radionuclides into the containment and from there into the environment. VICTORIA[1] is a mechanistic code designed to model the release, transport and deposition of fission products inside the reactor coolant system during a severe reactor accident. The code development and validation is an international effort, involving three countries (US, UK, Canada) and five laboratories (Sandia, Argonne, Oak Ridge, Winfrith, Chalk River). About 20 user-groups exists all over the world.

The version VICTORIA-92 follows the evolution of 25 elements and 288 species. It requires an initial element distribution, and the variation of the thermohydraulic conditions and geometry of the problem under consideration must be given by the user. The geometry in VICTORIA is discretized into a two dimensional cylindrical (r,z) Eulerian mesh. This allows investigations in the interior part of a reactor vessel as well as e. g. in thermogradient tubes. Boundary conditions must be prescribed at the outer surfaces of the mesh. The calculational framework consists of five main parts (submodules), which are explicitly coupled:

- Equilibrium chemistry calculations for all species located within the fuel geometry and the bulk gas and aerosol particles, and on all structural surfaces.
- An aerosol formation, size evolution and transport calculation in the bulk gas medium and deposition onto structure surfaces.
- A decay-heat-induced structural film and surface heatup.
- A behaviour and transport calculation of the species within the geometry.
- A convective transport calculation of the species in the bulk gas medium.

Chemistry is modelled in VICTORIA in the thermodynamic limit, i. m. local equilibria are calculated by minimization of free energies. The aerosol behavior and transport calculations consider all established agglomeration- and deposition processes, including phoretic and turbulent effects. The superiority of VICTORIA in comparison with other related codes is the coupling of the aerosol and chemistry models under a wide variety of thermohydraulic and geometric conditions.

The Installation of VICTORIA-92

The aim of installation is to build a VICTORIA code executable on the user compute platform. The installation procedure is controlled by an install script being one component of the VICTORIA source package. The other components are makefiles for the code assembling and the compilation process, the Fortran source package of VICTORIA-92, and test data sets. Some subroutines (with system calls) and statements (depending on the precision) of the Fortran source code are different for the different compute platforms. If the Fortran source package contains a version of those subroutines and statements for the user's compute platform, the installation can be carried out automatically. For a list of certain UNIX operating systems the install procedure automatically assembles the operating system dependent source code and produces the VICTORIA-92 executable. That list of UNIX operating systems contains the following in the 93' distribution of VICTORIA-92:

SunOS
AIX
HP-UX
UNICOS.

The installation on any other UNIX system requires modifications of several makefiles of the install procedure and changes in some Fortran subroutines using system calls. For a typical user of VICTORIA this work could be too hard; therefore an automatic installation of VICTORIA-92 on ConvexOS is desirable.

To include the ConvexOS in the list of UNIX operating systems allowing an automatic installation of Victoria-92 the install procedure has been modified, and Fortran code containing system calls, as well as Fortran code tuned to Convex vector facility (see the section The optimization of VICTORIA-92) has been added to the Fortran source package of the VICTORIA-92 distribution. At the next VICTORIA meeting the version modified for ConvexOS will be provided for the VICTORIA community.

To make the right choice the **install script** at first asks for the name of the UNIX system, on which the installation is running. It uses the UNIX command **uname** existing on SunOS, AIX, and HP-UX, but unfortunately missing on ConvexOS. To run the install script on ConvexOS a improvised version of **uname** has been written. This improvisation of **uname** doesn't recognize any option. It makes use of the system call uname and puts into the standard output a string like

```
ConvexOS rhea C3832 10.2 convex.
```

A complete command **uname** recognizing all common options should be included in the standard command set of ConvexOS.

To extract and to assemble the Fortran source code needed for the user's compute platform the install procedure uses an emulator of the CDC update processor. The install script generates the control statement for this update processor. The corresponding control sequences for ConvexOS have been added to the install script. Because each Fortran compiler of a UNIX system has its own name and its own set of options the install script activates an operating system specific group of makefiles carrying out several compilations for the recognized UNIX system. The added ConvexOS specific makefile group contains the following definitions

```
COMPILE= fc
FFLAGS = -f1 -72 -O2
```

for several compilations, and the loader option

```
-lveclib
```

to use the VECLIB for Fortran code optimization.

There were two kinds of changes in the **Fortran source package**. The first concerns subroutines with system calls, the other concerns the Fortran code optimization. Three of the ninety-five subroutines of the Fortran code have system calls:

- the main program **viki92** calls **getarg** and **open** to define I/O files;
- the subroutine **now** calls **etime** and **fdate** to get either the CPU time in seconds or a date/time string;
- the subroutine **rundet** calls **getpid**, **hostname**, **getcwd**, and **getlog** to put information about the current run of VICTORIA-92 into the output.

Convex specific code segments have been included in the decks VIKI92 and NOW of the Fortran source pacckage. The Convex specific deck RUNDET has been completely rewritten and added to the Fortran source package. The code segment included in the deck VIKI92 is, for instance,

```
*IF DEF,CONVEX
   c *** ConvexOS specific coding
   c
         call getarg(0,nmexec)
         call getarg(1,vikin)
         call getarg(2,vikout)
         call getarg(3,tty)
         call getarg(4,vikgrf)
   c
         open (nivi,file=vikin)
         open (nouv,file=vikout)
         open (nttyv,file=tty)
         open (ngrf,file=vikgrf,form='unformatted')
         call page('viki92 ')
         lines = 15
         call rundet
    *ENDIF
```

(The lines beginning with * are declarations for the CDC update processor).

The described above modifications of the install script, of several makefiles, and of the Fortran source package have allowed an automatic installation. The correctness of the Convex specific VICTORIA code has been verified using test input decks and output listings of the VICTORIA source package. As the next step an attempt was made to optimize the Convex specific VICTORIA code.

The Optimization of the VICTORIA code

The VICTORIA code is a big program. Its Convex specific version consists of about 17.000 statements in 95 subroutines. Several authors have made the choice of the algorithms and worked out the sections of the code. Without getting into a close contact with the developers an efficient optimization is not possible. But time and manpower resources weren't enough to carry out a low layer optimization. Should VICTORIA find a wide application in our Research Center an algorithm based optimization will be conceivable. Thus the optimization was restricted to a code tuning, no changes of algorithms were made.

The foundation for a code tuning is the run time measuring. The VICTORIA code itself contains subroutines for run time measuring of the submoduls of the calculation. A run time summary like

total	fuel	chem	gas tran	aero	decay aer tran	
596.7	40.2	258.2	60.7	183.6	14.1	13.6

is put into the output listing. These time data are rather rough. They have served for an overview of which submodul takes which share of the run time, and for the measuring the results of the optimization.

In a first step of the code tuning the run times for nooptimized, -O1optimized, and -O2optimzide code have been compared:

nooptimized:

total	fuel	chem	gas tran	aero	decay aer tran	
596.7	40.2	258.2	60.7	183.6	14.1	13.6

-O1optimized:

total	fuel	chem	gas tran	aero	decay aer tran	
403.5	26.5	192.5	26.2	117.2	11.9	7.9

-O2optimized:

total	fuel	chem	gas tran	aero	decay aer tran	
391.6	40.4	143.1	22.0	152.7	10.6	7.9

The measurements show that in the submoduls fuel and aero there are subroutines with DO-cycles unsuited to be vectorized. To find them out more detailed run time information is needed. The Convex performance analyzer cxpa was used to obtain this information. All subroutines of the VICTORIA code have been recompiled with the -pa -O1 and -pa -O2 options and have run under the cxpa. The analysis of the routine monitoring report has shown two results important for the code tuning:

1. The Victoria program calls BLAS and LINPACK routines not from the VECLIB, but from a library being a component of the VICTORIA code.
2. Eleven subroutines of the VICTORIA program take more CPU time in the case of the optimization level -O2 than in the case of the optimization level -O1. The share of the run time is considerable for five subroutines, for the others it is less than one percent.

That are hints for two easy code tuning activities. In the first place the calls of VICTORIA BLAS and LINPACK routines should be changed by calls of VECLIB routines. The dynamic call graph of the cxpa monitoring report has helped to find out the upper calls of the BLAS and LINPACK routines: **sgtsl**, **sgefa**, **sgesl**, **spoco**, **sposl**, and **ssort**. Because all real variables of the VICTORIA code are declared as implicit double precision variables these calls have been replaced by **dgtsl**, **dgefa**, **dgesl**, **dpoco**, **dposl**, and **dsort**. Besides, the loader option **-lveclib** has been added to the makefile statements describing loader options for the executable VICTORIA code. In the second place the optimization level has been set to -O1 by an **OPTIONS** statement for those subroutines that take more CPU time in the case of optimization level -O2 and have a share of less than one percent of the whole run time. The following run time summary shows the results of this work:

total	fuel	chem	gas tran	aero	decay	aer tran
289.4	26.6	103.3	22.2	104.1	10.5	7.9

As a next step all subroutines that take more than three percent of the whole run time have been analysed for code tuning, regarding a vectorization. The effect of tuning the other routines was expected to be insignificant, and the work to be spent would have led to an unfavourable "price/performance" relation. Eight subroutines have been examined. Only two of them have a structure allowing a manual DO-loop optimization. The other subroutines are not tunable because of the chosen algorithm and numerical methods. For the purpose of DO-loop analysis cxpa runs have been undertaken with loop monitoring for the two mentioned subroutines. A common problem has been the times of DO-loop executions. The mostly used compiler directives have been **C$DIR SCALAR** - because of execution times less than five - and **C$DIR NO_STRIPS** - because of execution times of the order of ten. In some cases the body of a loop could be rearranged for full vectorizing.

The final result of all code tuning activities is

total	fuel	chem	gas tran	aero	decay	aer tran
275.7	25.7	104.6	22.5	89.6	10.4	7.9.

The total run time of the VICTORIA program has been reduced by 30 percent in the described above manual optimization. This value is not a top value, but it economizes CPU time.

[1] Heames, T. J. et al. (1992) VICTORIA : A Mechanistic Model of Radionuclide Behavior in the Reactor Coolant System Under Severe Accident Conditions.
Report NUREG/CR-5545, SAND 90-0756,R3,R4,Rev1; Sandia Nat Lab., Alberquerque, Sept.1992

info-convex AND THE USER GROUP LIBRARY

Karen Birkelbach
Southwest Research Institute

1. Introduction

Pemrac (pemrac.space.swri.edu) is a Convex C120 with 48 Mbytes memory and 4.5 GBytes disk space across both Multibus and VME. It has two ethernet controllers, one Multibus and one VME bus. It was purchased as part of NASA's Upper Atmosphere Research Satellite project. Pemrac is home to the info-convex mail list and the Convex User Group Software Library.

2. What is info-convex?

info-convex is an unmoderated mail reflector based at Pemrac. It is set up using an alias to an include file which is edited manually as requests are received. Its companion aliases are info-convex-request and new-convex. New-convex is a strictly internal alias used for list maintenance. As of September, there are 184 addresses subscribed to info-convex and approximately 12 are also mail reflectors.

2.1. How Does One Join info-convex?

To join info-convex, simply send e-mail to info-convex-request@pemrac.space.swri.edu. Requests to be added or deleted that are sent to info-convex-request are then forwarded to karen@pemrac.space.swri.edu. The aliases are listed as follows:

```
info-convex: "|/usr/lib/sendmail -oi -finfo-convex-request info-convex-include"
info-convex-include: :include:/we/info-con/convex.dis
info-convex-request: karen
new-convex: :include:/we/info-con/newsubscribers.dis
```

As requests are received, the mail message is saved to both "/we/info-con/newsubscribers.dis" and "/we/karen/.rolo.cug." The new subscribers file is then edited down to just valid e-mail addresses and an e-mail with the welcome message is sent to the new-convex mail aliases with verification turned on to make sure that the e-mail addresses are reachable. The .rolo.cug file is to ensure there is a complete mail header should one of the addresses fail and also for record keeping. The welcome message includes instructions to unsubscribe and how to reach both the user group software library and back digests of info-convex mail messages.

When messages are sent to the info-convex alias at Pemrac, each message is piped into the sendmail program to change the sender to info-convex-request to prevent looping. The account "info-con" on Pemrac is also on the mail list and it is used as the mechanism for archiving messages into a digest. Mail messages to info-con are saved to date-named files on a monthly basis and all mail-header information is removed, except the "From:," "Date:," and "Subject:" fields. These files are then placed in the Convex anonymous ftp area on Pemrac to be retrieved at a later time by non-subscribers and new subscribers.

2.2. What Has Happened With info-convex Since comp.sys.convex?

Before comp.sys.convex existed, one site set up a temporary gateway that caused some confusion around Christmas-time, 1992. A user sent e-mail to a non-existent newsgroup that caused every NEWS router to kindly forward the message to info-convex, which generated an untold number of messages over nearly a week. Such things still tend to happen occasionally, but I have thus far not found a way to prevent it.

When comp.sys.convex was first established, a gateway did not exist between info-convex and comp.sys.convex. At this time, there is only a gateway from comp.sys.convex to info-convex, which means

that mail messages sent to info-convex are not readable on comp.sys.convex. However postings to comp.sys.convex are forwarded to info-convex. The gateway software being use is the news2mail portion of Rich Salz's newsgate package and has kindly been installed at convex.com by Eric Schnoebelen. Getting the info-convex to comp.sys.convex portion of newsgate installed is less trivial than news2mail, hence it has been delayed.

Before the newsgroup began, the ratio of requests to join info-convex versus requests to be removed had been approximately 8-10 to 1. Since January, 1993, there have been 30 requests to join and 16 to be removed from the list. Of those 16 requests, four were users who said they were going to read comp.sys.convex instead and 12 were for other or unknown reasons. Of the eight users I was able to ask why they were leaving, most were because the user was changing employment to a place that did not have a Convex, one was because their Convex machines were being phased out, and one was for maternity leave.

2.3. The Future of info-convex

The future of info-convex is not in jeopardy, at least for the near term because there will probably be a a small subset of Convex sites that will not take news. However, I expect the size of the mail reflector to shrink from its current size when the info-convex to comp.sys.convex gateway becomes available.

3. The User Group Library

The Convex User Group Library archives are housed at Pemrac.space.swri.edu. The library contains software written or adapted specifically for Convex computers. Included are source code and binaries. In some cases, patches to make public-domain software build properly on a Convex are provided. All software has been donated by users of Convex computers.

3.1. What is in the Library?

There are many programs kept in the archives, including programming aids, FORTRAN mathematical code, UNIX shells, filters, printing software, and GNU software. Each item in the archives is contained in a compressed tar file and each tar file is listed, by its contents, in an index file. The current contents of the index is shown below.

acctrunc: truncate accounting records
amd: Berkeley replacement for Sun's 'automount'; automatically mounts and unmounts NFS directories on demand
ansimount: front end to tape mount utility to verify that the requested tape actually has the same ANSI label written on it
ansiread: ansi tape reader (similar to ansitar)
arc: file archival software (RCS files, uses Lempel-Zev compression)
arithcomp: arithemetic data compression routines
autodump: script for saving more than 1 file system per tape
availplot: analyzes the uptime of your machine(s)
basic: U. of British Columbia BASIC
calls: finds out who calls who in C source files
calstuff: calendar generator ?
cchk: check C source for errors that lint may not find
cdctape: read various CDC tape formats

cedist: simple text editor
chart: draws charts using nroff (and possibly troff)
cmake: looks obvious, but isn't (??)
cmedit: line editor from Convex
compress: 8->7 bit compression for modem data xfer
compressdir: Lempel-Zev file compression software (this is good stuff!)
con: virtual terminal program (seems a bit esoteric)
copytape: a specialized program to put and fetch files from tape
cparen: adds parenthesis to C expressions for clarity
critdaemon: facility to mail critical daemon events to "root" after they occur
csubs: C subroutine library from H. Diamond labs
cut: program to extract selected columns from a file
c_directory_examples: example C programs that manipulate directories
c_out: H. Diamond labs C pretty printer utility
data_general_tapes: utilities to read Data General AOS & RDOS tapes
diffmark: adds difference marks to nroff files
dig: searches the network for networking information on a given host
dirstack: directory stack functions; replaces pushd & popd; simplifies use and provides a history function.
diskmail: utility to mail a disk usage message to each user
ditrev: reverses pages of ditroff files
dtree: prints a visual directory tree (nice utility)
email_answering_machine: automatically answers your email when you can't
fasplot: pen plotter routines for the Talaris & QMS laser printers
ff: fast text formatter similar to nroff
fortran_unix_calls: FORTRAN callable routines to do raw I/O, etc
ftn_utilities: Diamond labs FORTRAN utilities library
games: here are all your favorites
getindex: ??
get_serial_number: returns the serial number of the machine running it
gdb: GNU debugger binary for Convex
gnu_emacs: loads of GNU emacs stuff, lisp, et al
gnutar-1.10: GNU tar facility binaries for Convex
help_sys: U. of Paderborn distributed help system (some documentation translated from German)
hex: ascii hex dump program
history: history program (?)
imagen-acct: accounting for a laser printer
imagen-plt: ??
imcal2: more calendar programs
iprx: imagen printer emulator (on which devices, I don't know)
imPRESS: imagen printer software
kermit: the kermit file xfer software
laminate_files: script to paste files side by side (also, see paste)
less: variation on more & pg
lplot: plot(3) type filter for the Printronix P300 series printers
m3: makes makefiles for text processing (meta makefile?)
Makefile: a makefile for many of the programs in this library
makediskchart: charts of disk layout and usage (awk scripts & C programs)
makedisklabels: variation on makediskchart
mflops: Convex FORTRAN optimization analyzer (vector/parallel code)

mh: RAND mail & message handling system
modemlast: collect modem use statistics
ncar3.0_convex: Patches for NCAR Graphics 3.0 to make for Convex
nqs: Network Queueing System (source and documentation)
paste: concatonate files side by side
patch: applies diffs to original sources
pathalias: uucp maps and maintenance software (see note below)
pcnfsd: provides authentication and remote print support for PC-NFS
prnfiltr: output filters for Epson SQ2500, HP7550A, and DEC-LN03
progs: mucho good utilities, such as pack, xmodem (most have a man page)
prxplot: Printronix plot filter
psp: prints ASCII files on PostScript printers
ptr: NIH problem tracking system
qm: Quine-McCluskey reduction program
qmprof: variation of qm
queues: queue management routines (lpq, etc)
reserve: reserves a conference room
rim: the RIM dbase system (??)
rmstuff: rm related scripts
rolo: online "rolodex"
runindex: ??
sc: spreadsheet calculator
sched: maintains and prints schedules
schedule: ??
scripts: scripts & man pages (you can find "yell" here)
snmp: network management routines from cisco public ftp area
solve: interactive equation solver
stat2: compute and display CPU usage statistics
symbols: ??
sysadm: System Administration tools in PERL provided by Mark Shoemaker
termcap: new termcap entries
tekstuff: source to tekcolor & tekkey utilities for the 4110/4120 CRT
tgraph: graph maker
tm: time management software
top: neat utility from Rice U. that displays info on interactive processes
tparch: tape archival program
tplot: a plot(5) style filter for Tektronix 4110/4120 CRTs
tpp: text pretty printer
tree: similar to dtree
tcsh-6.00.03: tcsh binary for Convex
util: some Convex supplied FORTRAN utilities for making load maps, etc.
unmbasic: U. of New Mexico BASIC
users: new account management programs
vax_binary_io: routines for accessing VMS RMS binary files from a Convex
vms_rwho: rwho daemon for VMS systems with the Excelan TCP/IP products
vmul-tricyc: fast vector multiplication, tridiagonal matrix solution assembly and FORTRAN code
voyeur: Parallel code analyzer
vplot: Stanford versatek plot library
windows: U. of Maryland window library (does not require a graphics CRT)

xref: cross reference generator
x_windows: Project Athena X_windows (ver. 10.4)
UNCtools: Set of tools provided by Michael Padrick from his talk "Who is Using My Machine and What Are They Doing With It?", 1991 CWWUG Conf.

It is obvious that a large number of programs have been submitted with little or no documentation, which is reflected in the index as question marks.

3.2. How to Contribute

Contributions are always welcome. To make a contribution to the library, the following items are requested:

1) A copyright or disclaimer notice;
2) The submission with 1) included in a compressed tar file;
3) The submission placed on Pemrac via anonymous ftp to the Convex/INCOMING directory;
4) A mail message notification that the submission is in the incoming directory sent to the librarian with a one-line description of the submission suitable for use in the index file.

Following these procedures will ensure that each submission will be included in the library in a timely manner. Contributions are also accepted on 9-track tape at any conference or they can be mailed by land/air mail to the librarian.

3.3. How to Get Software

Data can be retrieved from the library either over the network or through land/air mail.

3.3.1. Anonymous ftp

To retrieve selected programs from the archives, anonymous ftp is enabled on Pemrac. The following instructions will retrieve psp, a postscript printing program:

```
% ftp pemrac.space.swri.edu
Connected to pemrac.
220 pemrac FTP server (Version 6.32 Tue Jan 12 13:07:32 CST 1993) ready.
Name (pemrac:karen): anonymous
331 Guest login ok, send e-mail address as password.
Password: your_username@host.domain
230-
230-You have connected to a U.S. government computer system.
230-Unauthorized connection is subject to federal prosecution.
230-
230-If you have any problems, try using a dash (-) as the first character
230-of your password -- this will turn off the continuation messages that
230-may be confusing your ftp client.
230-
230 Guest login ok, access restrictions apply.
ftp> cd Convex/convexug
250 CWD command successful.
ftp> bin
200 Type set to I.
```

```
ftp> get psp.tar.Z
200 PORT command successful.
150 Opening BINARY mode data connection for psp.tar.Z (8099 bytes).
226 Transfer complete.
local: psp.tar.Z remote: psp.tar.Z
8099 bytes received in 0.0065 seconds (1.2e+03 Kbytes/s)
ftp> quit
221 Goodbye.
%
```

The ftp daemon running at Pemrac is not part of the ConvexOS release, but the modified ftp by Chris Myers, which allows us to disable some usual ftp functions. The MKDIR, RMDIR, DELE, and RNTO functions have been disabled and username logging have been enabled because of some security problems. A cracker was able to steal the password file because of a hole in ftp and was trying to plant trojan horses in some user accounts. On other occasions, network thieves were using Pemrac as a site to transfer and exchange illegal software. For these reasons, and the fact that Pemrac is a well-known host on the Internet, the decision was made to tighten security.

3.3.2. Magnetic Tape

The entire user group library can be received on 9-track magnetic tape at a cost of $60.00US by sending a request to

Karen Birkelbach
Division 15
Southwest Research Institute
P.O. Drawer 28510
San Antonio, TX 78228-0510
USA

A mirror site for the archives is located at the University of London's pluto.ulcc.ac.uk. The files can be found in convex_ug/us_archive_1992.

3.4. The Future of the Library

Plans for the Convex User Group Library include automated updating of the mirror of the archives by either checking the contents periodically or to just send new entries automatically. It would be useful to add other mirror sites to bring the archives closer to user "as the data flows." In addition, use of GOPHER for software retrieval and WAIS indexing for info-convex mail messages would provide enhanced access to these important sources of information.

MINISTÈRE DE LA DÉFENSE

DGA

DIRECTION
DES MISSILES
ET DE L'ESPACE

LABORATOIRE DE
RECHERCHES
BALISTIQUES ET
AÉRODYNAMIQUES

Vernon, July 21 1993

Information Processing
Department

Author : IETA BENNE-DARSES

MEMORANDUM PROJECT N° 368/93/STI/IS/NC

Subject : LRBA PRESENTATION TO CONFERENCE OF EUROPEAN CONVEX USERS
OCTOBER 28 1993.

Top copy to :

CONVEX -Ms AUPETIT

cc :

D - SD/T
STI/CS
STI/IS/GG
STI/IS/LSA/BS
STI/IS/SIM/ER-MA-CA
BT/MM MARTIN-BONADEO
SME (2 copies)
STI/Sec

HEISER
Traitement de l'information

LRBA - B.P. N° 914 - 27207 VERNON CEDEX - TÉL. 32.21.42.00
TÉLEX : 770 817 - TÉLÉCOPIE : 32.21.44.45

CONTENTS

1. GENERAL DESCRIPTION OF THE DGA

1.1. Structure of the **DGA** (*Délégation Générale pour l'Armement* / Central Weapons Directorate) ..... -3-

1.2. Structure of the **DME** (*Direction des Missiles et de l'Espace* / Missiles and Space Division) ..... -3-

2. DESCRIPTION OF THE LRBA

2.1. The LRBA - a Missiles/Space Technical Facility ..... -5-

2.1.1. The Laboratory's technical activity ..... -5-

2.1.2. Current programmes ..... -5-

2.2. The structure of LRBA technical departments ..... -6-

3. ACQUISITION OF A SERVER FOR REAL-TIME COMPUTATION ..... -7-

3.1. General background : system specifications and selection

3.1.1. General description of project ..... -7-

3.1.2. The OSAACAS simulation tool ..... -7-

3.1.3. Required performance levels ..... -7-

3.1.4. System prospection: choosing a CONVEX configuration ..... -8-

3.2. Description of chosen equipment ..... -8-

3.3. Processor capability ..... -9-

4. DESCRIPTION OF APPLICATION

4.1. Hybrid extensions required to enable system to include real components ..... -9-

4.1.1. General remarks ..... -9-

4.1.2. Extensions to the simulation tool ..... -9-

4.2. Development of a Hybrid/Real-Time Demonstration System

4.2.1. Hybrid OSAACAS simulation: system diagram ..... -10-

4.2.2. Planned versions of the Hybrid/Real-Time Demonstration System ..... -11-

APPENDIX ONE ..... -15-

APPENDIX TWO ..... -16-

1. GENERAL DESCRIPTION OF THE DGA

1.1. Structure of the DGA (*Délégation Générale pour l'Armement* / Central Weapons Directorate)

The French Ministry of Defence is divided into two main organisations:

- The Armed Forces: Land
  Air
  Navy
  Domestic policing (*Gendarmerie*)

- The DGA - *Délégation Générale pour l'Armement* or Central Weapons Directorate

This latter organisation comprises the following:

Operational Divisions:

DAT *Direction des Armements Terrestres* - Land Weapons Division
DCN *Direction des Constructions Navales* - Marine Engineering Division
DCAé *Direction des Construction Aéronautiques* - Aeronautical Engineering Division
DME *Direction des Missiles et Espace* - Missiles and Space Division
DRET *Direction des Recherches Etudes et Techniques* - Technological and Project Research Division
DEI *Direction de l'Electronique et de l'Informatique* - Electronics and Information Technology Division

Departments:

SCAI *Service Central des Affaires Industrielles* - Central Department for Industrial Liaison
SIAR *Service de la Surveillance Industrielle des ARmements* - Department for monitoring of the weapons industry

and various other Divisions:

The "*Gendarmerie de l'Armement*" - Weapons Police
DPAG *Direction des Personnels et Affaires Générales* - Personnel and General Affairs Division

1.2. Structure of the DME (*Direction des Missiles et de l'Espace* / Missiles and Space Division)

The **LRBA**, the *Laboratoire de Recherches Ballistiques et Aérodynamiques*, or Ballistics and Aerodynamics Research Laboratory, is a facility within the DME and has four main areas of activity:

- missiles or weapons systems using strategic or pre-strategic weapons carrying nuclear payloads
- missiles or weapons systems using strategic or pre-strategic weapons carrying conventional payloads
- explosive materials and powders
- space

The DME is organised along the following lines:

- central administrative departments in Paris (in the Cité de l'Air)

- three technical departments in Paris:

- **ST3S** (*Service Technique des Systèmes Stratégiques et Spatiaux* - Technical Department for Strategic and Space Systems), which handles military space observation programmes in addition to programmes involving strategic and pre-strategic missiles with nuclear payloads,

- **STSMT** (*Service Technique des Systèmes de Missiles Tactiques* - Technical Department for Tactical Missile Systems)

- **STPE** (*Service Technique des Poudres et Explosifs* - Technical Department for Explosive Materials and Powders)

- four outlying facilities:

  - the **LRBA** (*Laboratoire de Recherches Ballistiques et Aérodynamiques* - Ballistics and Aerodynamics Research Laboratory) located at Vernon in the French territorial department of the Eure. Inertial systems, missile guidance and control, aerodynamics, environment and systems hardening. Staff of 520.

  - the **CAEPE** (*Centre d'Achèvement et d'Essais des Propulseurs et Engins* - Project Completion and System Trial Facility for Drive Units and Engines) located in Saint-Médard en Jalles (French territorial department of the Gironde). Ground and simulated altitude testing of solid propellant drive units, notably for ballistic missiles. Management of production resources for the assembly and completion of ballistic and pre-strategic missiles. Staff of 520.

  - the **CEL** (*Centre d'Essais des Landes* - Landes Test Facility) located in Biscarosse. Flight trials of strategic and pre-strategic missiles and tactical missiles. Staff of 2,370.

  - the **CEM** (*Centre d'Essais de la Méditerrannée* - Mediterranean Test Facility) in Toulon and the Île du Levant (French territorial department of the Var). Flight trials of tactical missiles, medium transfer missiles and submarine systems. Space surveillance. Staff of 740.

2. DESCRIPTION OF THE LRBA

2.1. The LRBA - a Missiles/Space Technical Facility

2.1.1. The Laboratory's technical activity

There are several sides to the activity of the Laboratory of which the most important are listed below:

Programme Upstream Role

The Laboratory's activity as a Technical Facility begins **upstream of programme implementation**.

This takes the practical form of LRBA involvement in the initial studies preceding the project itself (some of this research is in fact given to the LRBA) and prior testing to validate a concept, choose a technology, approve an architecture, and so on. Such tests are generally conducted by the LRBA.

The same applies to certain types of mandatory reference documentation (GAM EG13, GAM T17 etc.)

Programme Back-up Role

Once a programme has been decided upon and actually launched, the LRBA plays a "checks and balances" role in relation to industrial partners by monitoring engineering design work and product development and by providing assistance and advice for both hardware and software.

This active support continues throughout the development phase and may even be prolonged beyond this into the commissioning stage where the programmes involved present particular difficulties or where they have major strategic importance.

Technical Overviews

As a complement to this role in programme development, the LRBA also produces, from the standpoint of its knowledge of the current state-of-the-art and taking due account of the needs or requirements of the various Divisions, **technical overview reports** detailing the capabilities of any given **technology or method** to satisfy a need, and on the limitations of a technology, its capacity for further development etc.

In order to carry out this work, the LRBA conducts system or sub-system studies (on questions of accuracy, guidance etc.) using system behaviour simulations and by modelling complete systems or sub-assemblies in laboratory conditions.

Evaluations and Qualification Trials

Evaluation work, involving the whole range of possible studies right from the single component up to and including entire systems and at all stages of their operational life, also forms part of the activity of the LRBA and are the foundation of its institutional skills. The same holds true for **qualification** trials.

2.1.2. Current Programmes

The LRBA is currently contributing to the development work on the major weapons programmes:

- **Inertial navigation systems:**
  - Naval SNLE, surface craft, nuclear aircraft carriers, SNA
  - Land Tanks (LECLERC) and terrestrial navigation generally
  - Air Aircraft (ACT/ACM, Mirage 2000) helicopters

- **Strategic missiles:**

  - MSBS SM4, M45, M5
  - SSBS S3 and possible successors

- **Pre-strategic missiles:**

  - ASMP Air-to-ground
  - HADES Ground-to-ground
  - ASLP Air-to-ground

- **Tactical missiles:**

  - MAGIC 2 Air-to-air
  - MICA Air-to-air
  - APACHE Air-to-surface
  - ROLAND Ground-to-air
  - FSAF Ground/sea-to-air
  - MILAS Sea-to-Sea
  - AC3G 3rd Generation anti-tank weapon
  - SUPER 530 Air-to-air
  - AS30 LASER Air-to-ground
  - CROTALE Ground-to-air
  - MISTRAL Ground-to-air
  - EXOCET Sea-to-Sea
  - ACCP Short-range anti-tank weapon
  - BREVEL Drone

- **Space:**

  - HELIOS 1 and 2
  - Other satellites
  - HERMES

Miscellaneous activities

Private industrial companies (e.g. **MATRA, AEROSPATIALE, SEP** etc.) often make use of the LRBA's **test resources** in order to conduct their own system and equipment evaluation work in the aerodynamics, inertial guidance and environmental fields.

The **CNES** (*Centre National des Etudes Spatiales* - French National Centre for Space Studies) calls more and more frequently on the experience built up by the LRBA in the areas of inertial systems, celestial navigation sensors and the coupling of these two types of detector. The CNES commissions from the LRBA not only new concept feasibility studies for (e.g. in-orbit calibration) but also evaluation testing.

2.2. The structure of LRBA technical departments

Technical work is carried out in **six separate departments**, each covering a specific field:

- Aerodynamics
- Electronics
- Environments, Electromechanical Components and Metrology
- Inertial sensors, Navigation and GPS
- Systems Studies for Missiles and Space
- Information Processing

3. ACQUISITION OF A SERVER FOR REAL-TIME COMPUTATION

3.1. General background: system specifications and selection

3.1.1. General description of project

The needs to be filled are the following:

- Response to the requirements for implementation of simulation including real system components and factors
- Provision for the departmental laboratories of the LRBA of a server for scientific computation running under UNIX.

The priority need here remains that of providing a simulation capability processing real system components and factors, this being a resource used by the LRBA in the context of its technical overview work on missile programmes, not only where the general behaviour of the missile in its operational environment is concerned but also for the detailed analysis of its component sub-systems.

The main objectives of these simulation studies are:

- Prediction of missile behaviour
- Analysis of results obtained from test firing
- Greater knowledge of a missile's sub-systems ("hardware" and "physical entities"), of the influence of the missile's environment on its behaviour and the effect of that real behaviour on the performance of the entire missile system.
- Synthesis of the body of knowledge existing at a given moment on a given missile considered as a "system" and on its various "component systems".

3.1.2. The OSAACAS simulation tool

OSAACAS (*Outil Structurel et Automatisé d'Aide à la Conception et à l'Analyse des Simulations* / Resource for structured and automated aid in the designing and analysis of simulation programmes) is intended to help LRBA design personnel in the designing and updating of simulation programmes to meet the above-mentioned requirements. This system currently meets the whole range of criteria in so far as simulations based entirely on computer models are concerned.

Since simulations relying on computer models do not allow all aspects of the general control/guidance problem to be handled, it is necessary to enable interaction between the computer processor and real system components or factors, thereby creating "hybrid" extensions to the basic OSAACAS resource. The frequency of such interchange of data between the OSAACAS main processor and the real system factors integrated into the simulation loop constitutes the main constraint governing the "real-time" performance of the computer system.

3.1.3. Required performance levels

The objective was to acquire a computer configuration offering a high level of computational power plus the requisite performance in real-time processing.

The broad lines of the specification for this configuration can be set out as follows:

- A development environment with FORTRAN and C as languages and UNIX as the Operating System.
- Multi-processor architecture giving a minimum capacity of 50 MFlops DP Linpack per processor.

- A VME bus I/O system capable of two-way interchange of 80 32-bit words of data within a 390µs window contained in an overall processing cycle of 1ms (the application's repetition rate being 1000Hz).

3.1.4. System prospection: choosing a CONVEX configuration

Submissions from 19 companies were considered during the tender period.

The selection criteria were the following:

- Processor power (i.e. the power of each processor)
- VME bus I/O performance
- Real-time processing system quality
- Number and quality of tools provided for system fine-tuning and development
- Configuration upgradeability
- Credibility and quality of the services offered

An initial series of tests involving the intrinsic performance of the machines reduced to submissions to a short-list of 8 companies.

Only CONVEX was able to meet the criteria of the second series of tests specific to the target application both in terms of computational power and real-time capability.

3.2. Description of chosen equipment

Equipment supply was divided into two phases since the RTK, the Real Time Kernel, was not available in late 1992.

Phase One

CONVEX offered to place at the disposal of the LRBA a P5430 configuration of intermediate performance level in order to develop and validate system links with real-time missile system factors. This machine possesses three processors for computation addressing 256Mb of live memory and uses a THEMIS card to carry out I/O transfers of data in 80-word blocks in less than 390µs.

Two of the machine's processors run under UXE Real Time Kernel 1.1 and carry out real-time tasks.

One processor runs under UNIX and carries out tasks other than real-time.

Phase Two

As soon as it became available, the chosen P5620 was installed at the LRBA. This machine has two processors for computation and 512Mb of memory. It can carry out the same tasks as the P5430 but with all computation times reduced by half. Both processors operate under the RTK.

NB: A detailed description and diagram of the selected configuration are given here in an Appendix.

3.3. Processor capability

The P5430 is capable of 50 MFlops and 46 MIPS per 64-bit processor (the configuration possesses three such processors).

The P5620 offers 120 MFlops and 115 MIPS on each of its two 64-bit processors. This machine can accommodate a maximum of 8 processors.

4. DESCRIPTION OF APPLICATION

4.1. Hybrid extensions required to enable system to include real components

4.1.1. General remarks

The version of the OSACAAS simulation tool which uses mathematical modelling exclusively is restricted by certain limitations where Real-Time/Hybrid simulations are concerned:

- Firstly, certain computer models of missile sub-systems are not sufficiently faithful to the behaviour of their real counterparts,

- and, secondly, the simulation environment is based on a single-processor UNIX architecture (SUN-OS) which produces simulation times which are far longer than those involved in a real missile firing.

The "Real-Time" and "Hybrid" versions of the OSAACAS tool enable these objections to be met while retaining the general usefulness and flexibility of the basic resource.

4.1.2. Extensions to the simulation tool

The Real-Time and Hybrid extensions to the basic system amount to the addition of the following capabilities:

Real-Time Task Execution

The Real-Time version of the simulation tool had as its objective the addition of an environment for task execution based on a multi-task and multi-processor architecture which would make it possible to achieve simulation times near those involved in a real flight.

Task Execution using Real Missile Components and Factors

In the Hybrid version, the task execution monitor is equipped with extensions enabling it to exchange data with real missile systems according to a protocol specific to each such component: this capability is termed the REAL-TIME/HYBRID task execution monitor.

In addition, the real components introduced into the missile simulation will be included right from the specification and design stage of the simulation programme: this in turn implies further development of the OSAACAS data structure.

The real components currently involved here are:

- parts of the automatic guidance system using the STRADI simulation testbench.
- plus the Inertial Reference system.

NB: For full implementation, some of the above-mentioned components need in addition the use of the Laboratory's Test Resources (two- and three-dimensional movement tables) and the inclusion of the On-board Computer in the simulation.

4.2. Development of a Hybrid/Real-Time Demonstration System

4.2.1. Hybrid OSAACAS simulation: system diagram

The functional validation and final qualification of the simulation tool cannot be carried out without the implementation of a typical simulation or Demonstration representative of the simulations intended for OSAACAS.

A tactical missile with an automatic guidance system and its environment can be represented by OSAACAS using the following general models:

- the Flight Mechanics model integrating more or less complex aerodynamic factors
- Inertial Reference (IR)
- Information Processing Unit (IPU)
- Automatic Guidance System (AGS)
- Environment – incorporating target simulation and the various defence systems

Each of the real missile components or test resources integrated into the simulation programme converses with the computer processor via the simulation bus and through two interfaces:

- Real Component Interface RCI (software)
- Real Component Adapter RCA (hardware)

4.2.2. Planned versions of the Hybrid/Real-Time Demonstration System

The ultimate goal is the simulation of the entire missile, but since Flight Mechanics is the only mathematical "model", three consecutive versions of the Demonstration System are planned. These are:

Version 2.0, implementing:

- The complete model with simplified Aerodynamics and IR. This model can be run either on the STAR or as OSAACAS on the CONVEX. This comparative stage enables validation of this version of the Demonstration System and subsequent abandonment of the STAR.

- STRADI and the automatic guidance and missile environment front-end.

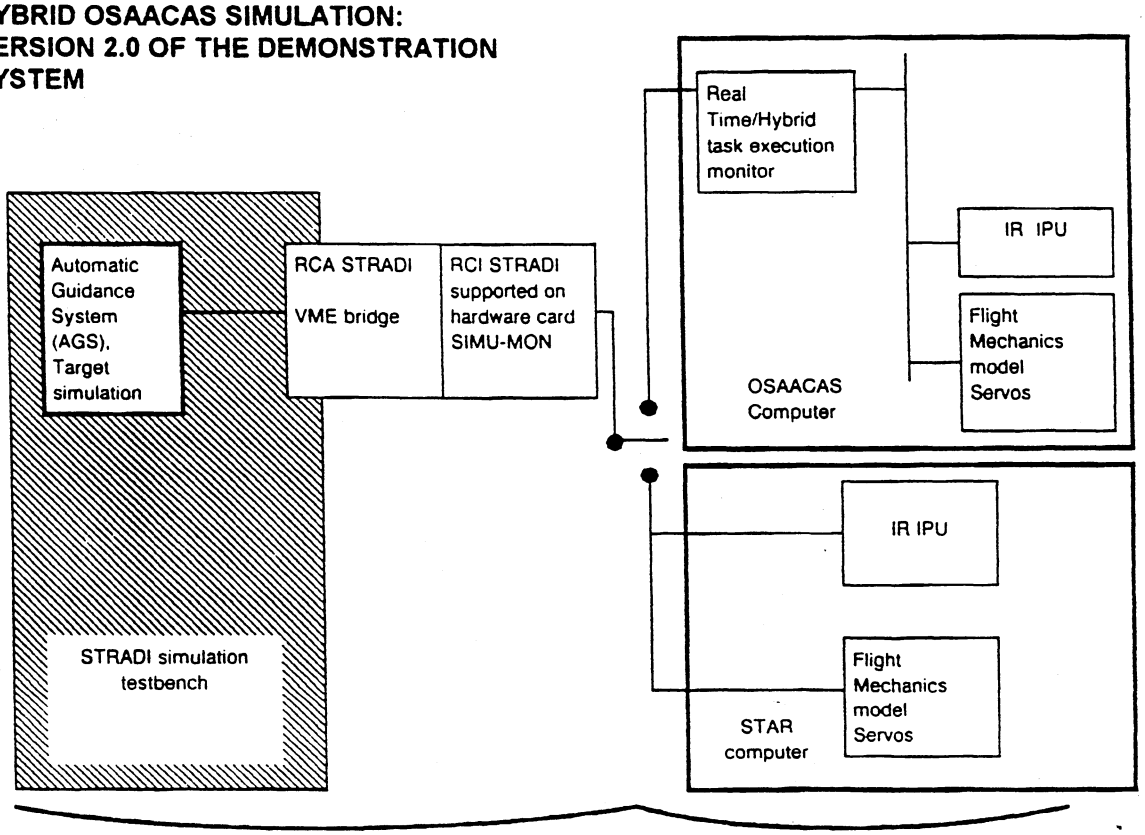
Version 2.1

- Introduces the automatic guidance and missile environment front-end installed on the three-dimensional movement testbench into the simulation programme (not shown in the diagram below).

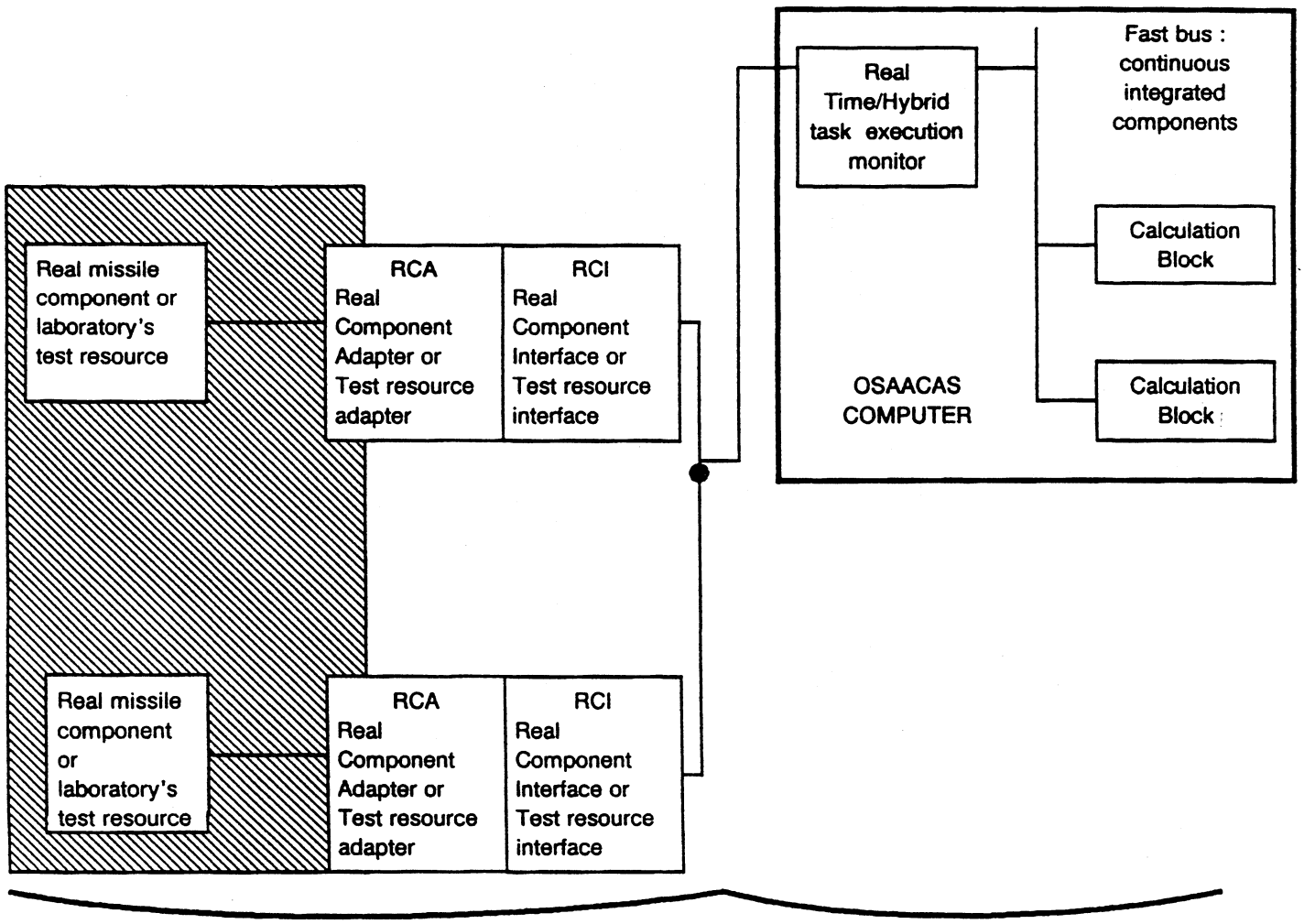
Version 2.2

- Integrates IPU and IR enabling the input of detailed aerodynamics to Flight Mechanics.

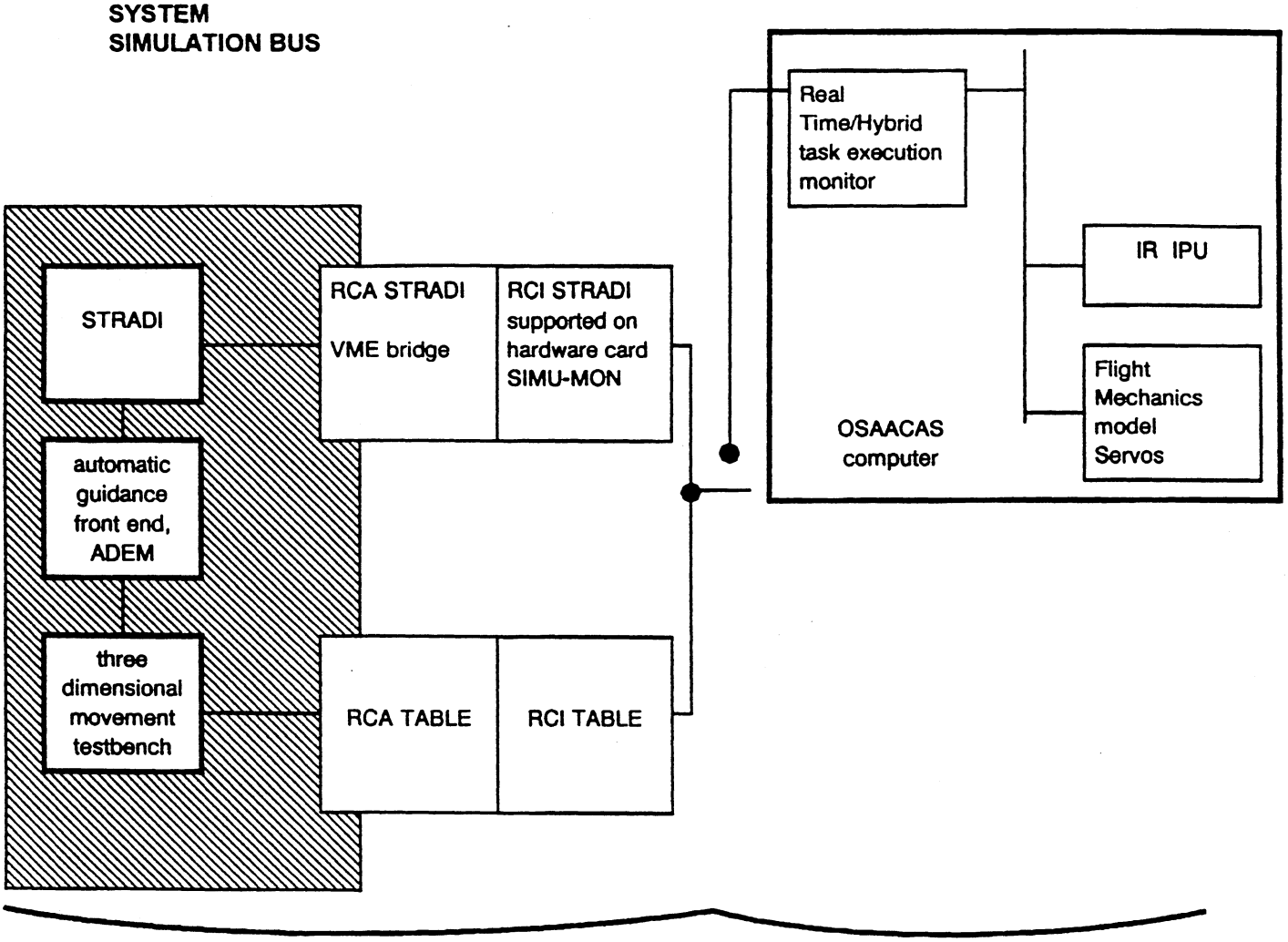
HYBRID OSAACAS SIMULATION: VERSION 2.0 OF THE DEMONSTRATION SYSTEM



Hybrid OSAACAS simulation can be described using the following system diagram:



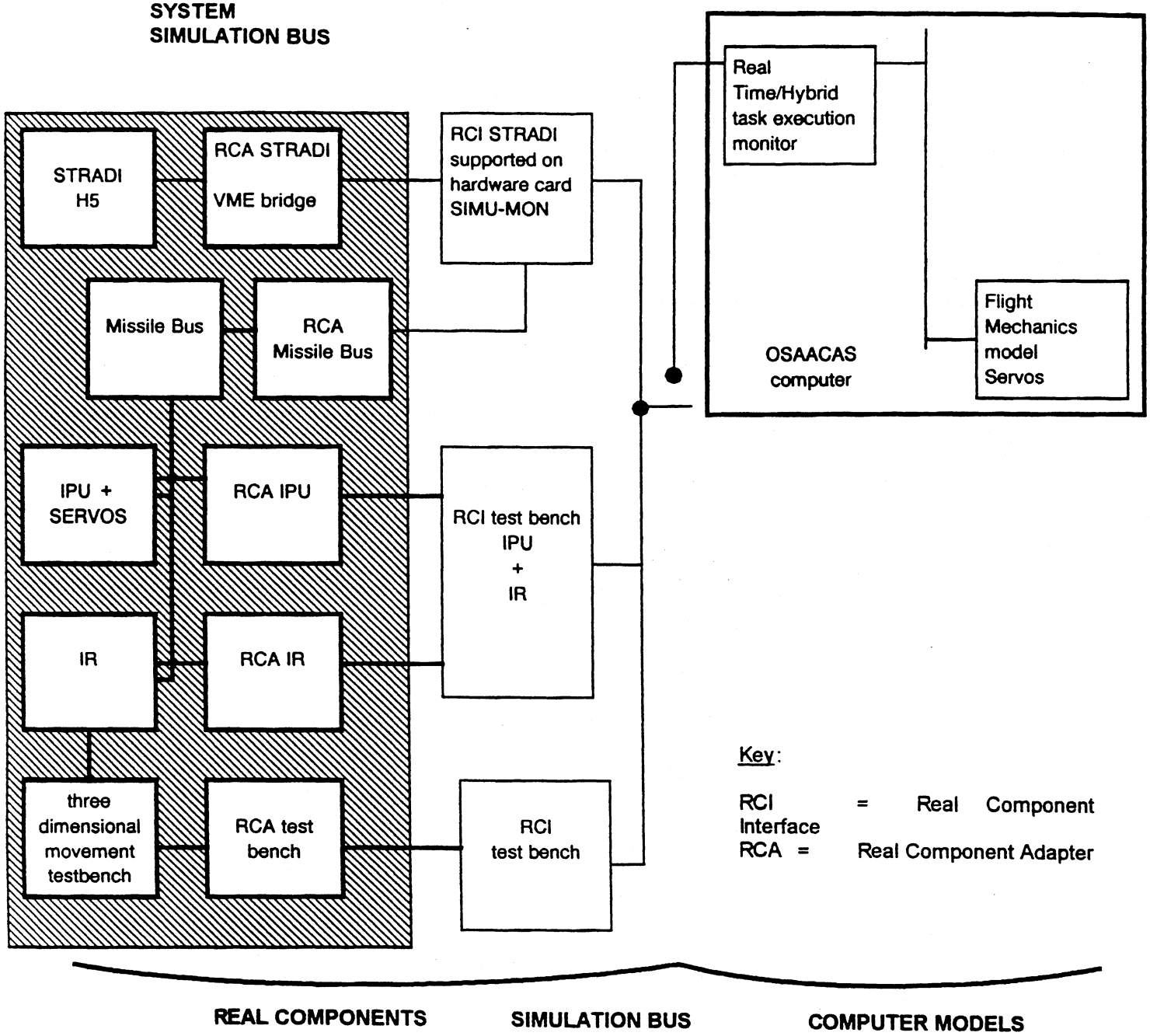
HYBRID OSAACAS SIMULATION:
VERSION 2.1 OF THE DEMONSTRATION SYSTEM
SIMULATION BUS



Key:

RCI = Real Component Interface
RCA = Real Component Adapter

HYBRID OSAACAS SIMULATION:
VERSION 2.2 OF THE DEMONSTRATION SYSTEM
SIMULATION BUS



APPENDIX ONE

DESCRIPTION OF THE CONVEX CONFIGURATIONS

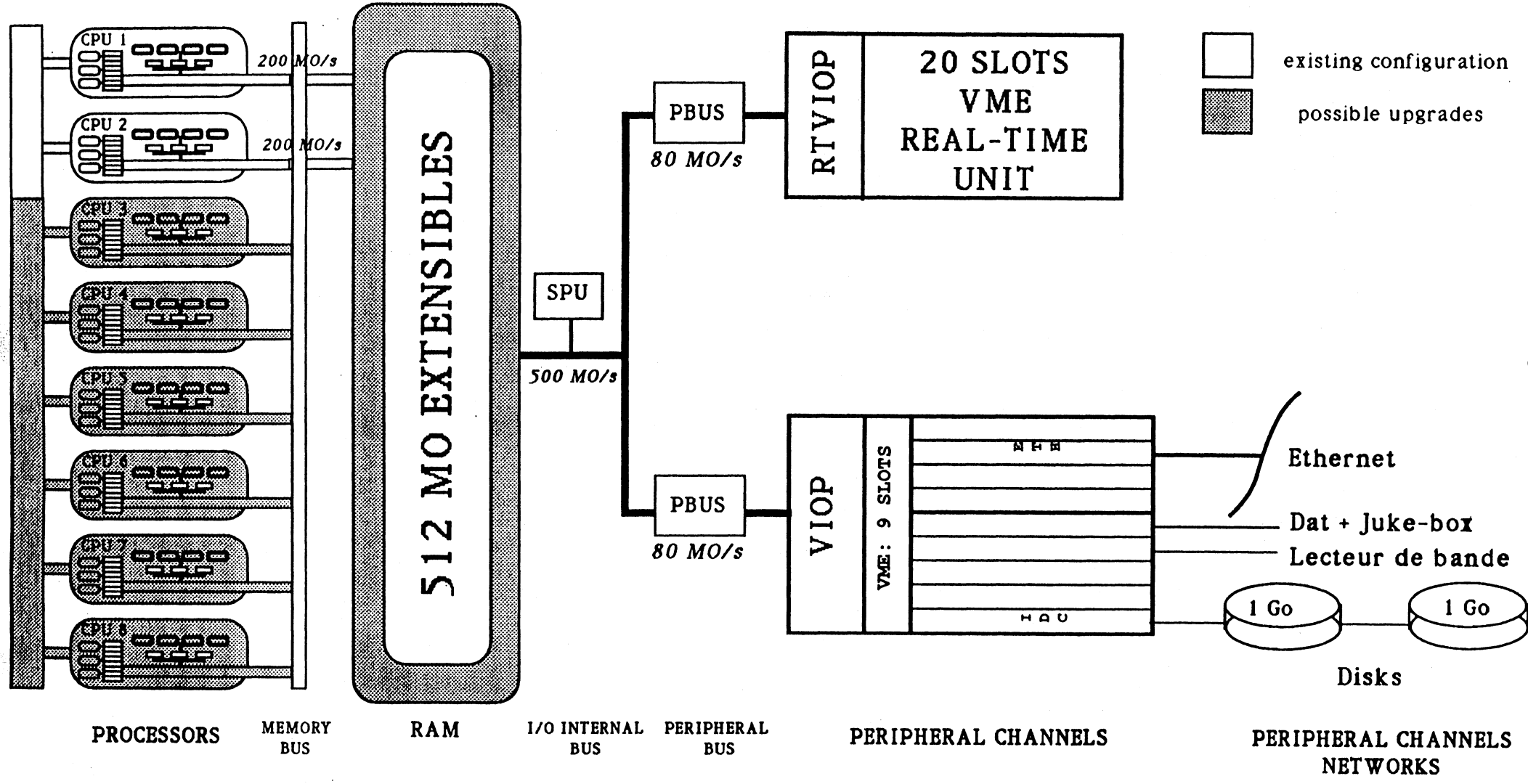
PHASE ONE

The first phase, up to the end of 1992 comprises the following:

- 1 main unit (5430) with 3 processors each capable of 46 MIPS and 50 MFlops.
- 1 VME real-time unit with 20 slots free for real-time I/O controllers (THEMIS card).
- 1 I/O processor controlling the real-time VME bus (RTVIOP).
- 1 VME non-real-time unit holding the Ethernet and disk controllers (IDC).
- 1 non-real-time I/O processor (VIOP) (68030).
- 1 Ethernet controller.
- 1 IDC fast disk controller.
- 2 fast IDC 1Gb disks

PHASE TWO

The peripheral components remain unchanged, but the main computation unit is upgraded to type P5620 with RAM increased to 512Mb, this unit being capable of 115 MIPS and 120 MFlops per processor.



APPENDIX TWO

CUSTOMER SATISFACTION ISSUES

Larry Abernathy
Convex Computer Corporation, Tx, USA

The presentation will discuss the new Customer Satisfaction Organization structure. This structure is designed with a view from the customer back through customer support, manufacturing and into development. The presentation will also discuss the approach being taken to make information more available to Convex users. This information will include problem report status, bug report status, TAC patch descriptions, technical tips as well as other information. Hopefully there may be another surprise or two.

EUROPEAN CONVEX USERS CONFERENCE

Friday, 29 October, 1993

The PARMACS Message-Passing Model – Features and Relations to other Concepts

Hans-Christian Hoppe
Karl Solchenbach

PALLAS GmbH
Hermuelheimer Str. 10
D-50321 Bruehl
Fax: +49-2232-189629

Abstract

The parallel computer market is characterized by large variety of architectures. Using standard message-passing libraries like PARMACS, EXPRESS or PVM, PALLAS has sucessfully demonstrated portability across vector computers, MPP architectures, and workstation clusters.

PARMACS are implemented on (nearly) all parallel systems including Convex C2/C3 and are used by many European application development or porting activities. Software development is supported by graphical visualization tools and numerical libraries basd on PARMACS. The relations of PARMACS to other message-passing interfaces, in particular to PVM, are described.

1 Parallel Programming Models

Parallel processing is the key technology for attaining the high computational performance needed for the effective solution of many compute–intensive technical and scientific problems. During the past decade, a variety of different parallel computer architectures have been introduced, and parallel computing has been firmly established in the science community and is beginning to be seriously used for industrial applications.

Application programs, however, which have been developed for conventional sequential computers are typically not (or not efficiently) executable on scalable parallel systems without modifications. In order to exploit the properties of parallel architectures, the parallelism inherent to the application has to be formulated more or less explicitely.

Corresponding to the predominating hardware architectures (shared-memory MIMD, distributed-memory MIMD, and SIMD) three programming models (for Fortran) have evolved:

1. **Shared memory model.**
   This programming model was developed for multiprocessor systems with shared memory and a limited degree of parallelism (e.g. Cray Y-MP, Convex C2/C3, Alliant, IBM 3090/9000 etc.). It is based on loop parallelism but contains also functional parallelism. The so-called PCF-Fortran was an attempt to standardize the shared-memory programming model.

   Future results will show whether and to which degree the shared-memory programming model is able to achieve scalability on large parallel systems with non-uniform memory access (e.g. KSR-1, Convex SPP-1).

2. **Data parallel model.**
   Data parallelism means that each instruction is applied to many (or all) data in parallel. SIMD machines are operated under data parallel models (e.g. CM-Fortran). High Performance Fortran (HPF) is a coming standard for data parallelism. It is based on the array syntax and operations of Fortran 90, compiler directives for the distribution of arrays, and the FORALL statement.

3. **Message-passing model.**
   The oldest and lowest-level model to program highly and scalable parallel systems is message-passing. The message-passing model is characterized by

   - processes with local address space
   - few basic primitives (SEND/RECEIVE)
   - explicit coding by the code developer / porter
   - no need for special compiler and OS technology
   - full control on data locality and exchange.

2 Message-passing interfaces and portability

The message-passing paradigm has been implemented by each hardware vendor as a machine-specific message-passing library. Subroutines in these libraries (Intel: NX/2, nCUBE: Vertex, Parsytec: PARIX, Meiko: CSTools, IBM: EUI, TMC: CMMD) are semantically quite similar (but not identical) and syntactically different. Applications ported for one parallel systems do not run on a different system.

The importance of portability, of course, is obvious and needs no additional explanation. Commercial codes as well as large industrial in-house codes will never be ported to parallel systems if they depend on non-portable software components.

Portability of message-passing-based applications can only be achieved if hardware independent message-passing libraries are used which are available for a wide range of parallel computers. Typically such vendor independent message-passing libraries have been defined by research laboratories. The origin and the status of the some of these message-passing libraries is described in the following table

Name	**Origin**	**Status**
PARMACS	Argonne Natl. Lab. and GMD	Product (PALLAS)
EXPRESS	CalTech	Product (ParaSoft)
PVM	Oak Ridge Natl. Lab.	public domain
P4	Argonne Natl. Lab.	public domain
TCGMSG	Argonne Natl. Lab.	public domain
CHIMP	Univ. Edinburgh	public domain

Using these vendor independent message-passing libraries guarantees a certain level of portability of application codes across various parallel systems. Nevertheless, there is a strong need to have *one* common message-passing standard which is agreed by all important hardware vendors. Such an attempt is currently ongoing, the MPI (message passing interface) initiative. After many meetings (with a similar procedure as developed for the HPF definition) the MPI definition will be published and released for comments in November 1993.

In the following, this paper gives a comparison of the two most important message passing interfaces used in Europe, namely of PVM and PARMACS. The former is a development of Oak Ridge National Labs tailored towards networks of Unix computers; it has quickly attained a lot of popularity mainly among universities and is now migrating to dedicated high–performance parallel systems.

PARMACS, on the other hand, grew out of a cooperation of Argonne Labs and GMD, is available for all major parallel architectures, and has proven its capabilities in a number of Esprit projects and the current RAPS and SERC benchmarking initiatives.

3 PARMACS vs. PVM

3.1 Availability, Supported Systems

In contrast to the HPF language and the upcoming MPI message passing standard, PARMACS and PVM implementations currently exist for all major MPP systems and clusters of Unix workstations.

PVM is oriented towards distributed computing on networks of loosely coupled Unix systems, implementing message passing on top of standard Unix interprocess communication mechanisms. Traditionally, communication with the node processing elements of those parallel systems that don't execute a full Unix kernel on the nodes was not supported. While for instance an Intel iPSC/860 could be part of a PVM virtual machine, a PVM application could neither start processes on the hypercube nodes nor communicate with them using PVM mechanisms. Only recently, PVM implementations for MPP systems start to appear (see table 1).

PARMACS, on the other hand, was designed from the outset for MPP systems, and consequently supports the use of the node processing elements in a uniform way. In addition to the classical distributed–memory parallel systems, some important shared–memory systems and networks of Unix workstations are supported.

Both interfaces support Fortran and C (PARMACS from version 6.0).

Table 1 lists the PVM and PARMACS implementations for the most important parallel platforms.

3.2 Portability and Compatibility

Both message passing interfaces are currently undergoing a significant change: the PVM interface changed drastically from version 2.4 to version 3.0, and PARMACS switched from the macro–based version 5.1 to the version 6.0 library. For the continued support of PARMACS 5.1 applications, a conversion tool is available that performs a source transformation to PARMACS 6.0 standard, whereas the conversion of PVM 2.4 applications in large parts has to be done by hand because of the principal change in the way processes are addressed.

The PARMACS user can either continue to program using the traditional macro interface (the transformation tool transparently takes care of macro expansion) or at any time transform his/her applications to PARMACS 6.0 and use that interface in the future.

The full PARMACS functionality is available across all platforms, while some implementations of PVM 3.2 (especially the MPP versions) enforce restrictions of the traditional dynamic PVM process model. Furthermore, many of the advanced features introduced with PVM 3.0 are not implemented in the current version (3.2) and will be added lateron.

Platform	PARMACS		PVM	
	Version	Distributor	Version	Distributor
Distributed–memory Systems				
Cray T3D	6.0	PALLAS	3.0 extended	Cray
IBM SP–1	6.0	PALLAS, IBM	3.2	PD
IBM SP–1			2.3 (PVMe)	IBM
Intel iPSC/860	6.0	PALLAS	3.2 MPP	PD
Intel Paragon	6.0	PALLAS	3.2 MPP	PD
Meiko CS–1	5.1	PALLAS	–	
Meiko CS–2	5.1	PALLAS	3.2	PD
nCUBE–2	5.1	PALLAS	3.2*	PD
Parsytec (GC & GCel)	5.1	PALLAS, Parsytec	?	
TMC CM–5	6.0	PALLAS, TMC	3.2 MPP	PD
Transtech (Paramid)	6.0	Transtech	–	
Shared–memory Systems				
Convex (C2 & C3 series)	5.1	Convex	3.2	PD
Convex (SPP-1)	?	?	3.?	Convex
Cray (Y/MP, C–90)	5.1	PALLAS	3.2	PD
Fujitsu VPP–500	6.0	PALLAS	–	
KSR–1	6.0	PALLAS, KSR	3.2	
Workstation Clusters (possibly heterogeneous)				
DEC 3100/5000	6.0	PALLAS	3.2	PD
HP 7xx	6.0	PALLAS	3.2	PD
IBM RS/6000	6.0	PALLAS	3.2	PD
SGI	6.0	PALLAS	3.2	PD
Sun	6.0	PALLAS	3.2	PD

Table 1: Overview of existing or announced PVM and PARMACS implementations.
Remarks: * PVM does not run on nodes

Some of the vendor–supplied PVM implementations (notably the Cray T3D) both enforce some restrictions and offer vendor–specific extensions to PVM for high–performance communication.

Currently the PALLAS implementations are brought to PARMACS 6.0 standard, so that an identical interface will be available across all PARMACS platforms. On the PVM side, the interface seems to be still under active development, and changes are to be expected in the future. For many MPP systems, the parallel OS enforces restrictions of the PVM model, so that portability obstacles will remain. Furthermore, the future of high–performance PVM implementations like PVMe is unclear.

3.3 Support

While the PVM development is sponsored by Oak Ridge National Labs, the resulting implementations are in the public domain, and industry–strength support and maintenance are not available. The MPP ports, in particular, are done by different developers not affiliated to ORNL. Bug reports can be mailed to the tool developers or posted in a newsgroup dedicated to PVM, and often result in source code patches being distributed, but in general a PVM user cannot rely on his complaints being recognized and bugs being fixed in a timely manner. Furthermore, the PVM user has to keep track of the different patches and versions and must perform the updates himself.

The only exceptions to that scheme will be the vendor–supported implementations: Cray and Convex have announced to offer a PVM 3.x interface as the standard message passing programming interface for their upcoming MPP products and will provide support, while IBM up to now distributes PVMe to customers without maintaining or supporting it.

PARMACS, on the other hand, is mainly distributed by PALLAS as a commercial product, and customers can rely on proper support and maintenance. Swift reaction to bug reports is assured, and maintenance contracts are available offering help for installation and use and automatic distribution of updates. The Parsytec, Convex and Transtech implementations are distributed by the respective vendors, who will provide support according to their normal procedures.

3.4 Application Distribution

A PARMACS application typically is distributed as a standalone executable image with the PARMACS libraries linked in, whereas a PVM application needs a compatible PVM system installed and running. Therefore, PVM itself either must be distributed along with the application or has to be acquired, compiled and installed by the customer. The user of a PARMACS application only interacts with the application itself, while in the PVM case the user must be familiar with PVM as well.

3.5 Standards

The PVM interface has attracted a sizeable user community in the US and in Europe, with the emphasis on universities and research institutions. The main attraction of PVM (at least up to version 2.4.2) was the relative ease of installation and use, that enabled a lot of people new to the field of parallel computing to write distributed applications and to run them on their workstations. However, in the field of high–performance parallel computing PVM has problems with performance and portability on MPP systems, and consequently has not attained the position of a "de–facto–standard".

A large variety of application programs has been written using PVM, including some very large numerical simulation codes. Most of these codes, however, are tailored for a loosely coupled network: they either perform a limited amount of communication only and therefore can cope with the high latency and small bandwidth imposed by the network environment or use PVM as a means to connect application components running on specialized systems (graphics computers for visualization and vector supercomputers for computation, for instance). For many of the applications, the question of portability and performance across a variety of parallel computers does not arise.

For software developers in an industrial MPP context, PVM is much less attractive because of the lack of assured support, proper documentation and stability of the interface.

The PARMACS interface has been developed in the Esprit projects GENESIS and PPPE in close collaboration with leading European application software developers to make sure it serves the needs of high–performance parallel computing. PARMACS has been selected as the programming model for the RAPS benchmark initiative as well as for the current SERC benchmark activity. In the Esprit projects PPPE and GP–MIMD–II, PARMACS is used as a low–level portable interface for both tool and application development. In each of these projects, portability and performance of the PARMACS codes was established.

The well–known GENESIS benchmark package uses PARMACS in its distributed–memory version. Furthermore, the current SERC benchmark activity is mainly based on large PARMACS applications. The ECMWF decided to parallelize their weather–forecasting code using PARMACS, and first results show portability and good performance on all major MPP platforms.

ectionPerformance

On MPP platforms, the PARMACS implementations go to great lengths to exploit the communication performance of the hardware, whereas the generic PVM implementation uses standard Unix interprocess communication mechanisms that impose a considerable performance penalty. The new MPP PVM implementations try to reduce that performance disadvantage by using the native communication mechanisms where possible, but still are at a clear disadvantage performance–wise.

For many applications, the PVM message buffer handling by packing routines incurs a considerable overhead over the PARMACS method of sending a buffer packed by the application programmer using the facilities offered by the programming language.

Platform	Criterion	PARMACS	Version	PVM	Version
Cray C–90	Latency	0.04 msec	5.1	3.0 msec	3.1
	Bandwidth	480 MBytes/sec		12 MBytes/sec	
KSR–1	Latency	0.07 msec	6.0	14 msec	3.1
	Bandwidth	8 MBytes/sec		0.3 MBytes/sec	
Intel iPSC/860	Latency	0.09 msec	6.0	0.34 msec	3.1 MPP
	Bandwidth	2.8 MBytes/sec		2.1 MBytes/sec	
TMC CM–5*	Latency	0.04 msec	6.0	0.4 msec	3.1 MPP
	Bandwidth	8.8 MBytes/sec		3.0 MBytes/sec	

Table 2: Performance of various PARMACS & PVM implementations.
Remarks: *: measured on a beta–version of the CM–5 OS

Table 2 shows a performance comparison for PVM and PARMACS.

3.6 Differences in Functionality

1. Process model:

   Both interfaces employ a master (or "host") process that starts up the application worker (or "node") processes. PVM supports the MIMD model and allows the dynamic creation of processes during runtime, while PARMACS implements a static SPMD–type process model.

   For many of the currently supported MPP platforms, the PVM process model had be restricted to SPMD.

2. Process topologies/groups:

   PVM 3.x has added a process group extension: processes can dynamically join and leave groups identified by a character string, and broadcasts and barriers can be restricted to one process group. This feature, however, is implemented in a non–scalable way and requires additional processes to be run. Furthermore, the application program must provide synchronization for any changes to the process groups.

   The PARMACS node processes are arranged according to a logical topology defined by the application (a torus/grid or graph); processes are addressed using the logical coordinate in the topology and an efficient process–to–processor mapping is performed. For workstation clusters, the initial process placement tries to level the load across the available processing nodes.

3. Communication modes:

   PVM and PARMACS both allow the point–to–point transmission of arbitrary–length messages identified by an integer tag value. PARMACS allows synchronous (sender and receiver perform a rendezvous) as well as asynchronous message transfer with blocking send and receive operations, while PVM supports asynchronous communication with blocking send and blocking or nonblocking receives.

For PVM, the message contents is specified by calling "packing"–routines that copy the data into a system buffer. On the receiving side, the data has to be unpacked in the same way. The PARMACS send routines accept a Fortran or C vector containing the message data and an optional specification of the data types, and the receive routines analogously fill a user–supplied buffer. Both models support heterogenous networks with differing data representations.

4. Synchronization:

   Both interfaces provide a global synchronization operation; the PVM barrier needs a process count argument and synchronizes the given number of processes in a group, while the PARMACS barrier always synchronizes all active node processes.

5. Collective communication/global operations:

   PVM supports a broadcast operation that sends a message to all processes of the specified group. PARMACS administrates a global tree across all processes that can be used to efficiently implement collective communication routines and global operations in a portable manner.

3.7 Support Tools

For PARMACS, an instrumentation library producing a log file of communication events is currently available. This library was optimized with respect to with minimal interference with the running parallel application. After the program run, the event logs can be viewed and analysed by a selection of graphical performance analysis tools including the well–known ParaGraph.

Furthermore, within the PPPE Esprit Project an integrated toolset consisting of a much improved graphical performance analysis tool, a scalable parallel debugger and a transformation tool for subset HPF is being developed.

While a graphical PVM programming environment (xPVM) has been promised for years, the only tool currently available is the xab online monitor program for PVM 2.4 applications. This tool produces an event log that can be analyzed by ParaGraph; in contrast to the PARMACS instrumentaton library it influences the parallel program execution noticeably. The Hence programming environment based on PVM is poorly suited for high performance parallel programming because of its programming model.

3.8 Future Developments

The existing PARMACS implementations are currently being updated to PARMACS 6.0 standard, while full backwards compatibility to PARMACS 5.1 is assured. When the MPI standardization process is completed, PALLAS will offer a conversion tool for the transformation of PARMACS codes into MPI compliant programs. Furthermore, a PARMACS library based on top of MPI will be available.

The further development of PVM, especially with regard to MPP platforms, is not very clear. Only Cray and Convex are committed to PVM 3.0 as their premier message passing programming interface, with at least Cray providing a large set of proprietary extensions. The transformation of PVM programs to MPI standard, while in principle possible, will be a much harder problem than for PARMACS.

4 Conclusion

Summarizing the above, PVM has advantages in the following fields:

- Price: freely available public–domain software
- Process model: dynamic process creation on many systems, limited support for process groups

PARMACS is leading for these aspects:

- Support and maintenance: by PALLAS or the system vendor
- Performance on MPP systems: low overhead by careful interface design and optimized implementations
- Stability of the interface: assured backwards–compatibility
- Mapping support: automatic process–to–processor mapping or load leveling
- Program development tools: instrumentation library and visualization tools available
- Better support for productization: no need to distribute and maintain a parallel runtime–environment

THE EUROPEAN PARALLEL COMPETENCE CENTRE

Ruud van der Pas

Convex Computer

1. Introduction

Beginning of May 1993 Convex has opened the European Parallel Competence Centre (EPCC) located in Utrecht, The Netherlands. Its major goal is to offer support in the area of parallel processing, focussing on distributed memory systems. This support does not only consist of offering hardware, but also provides consultancy services and trainings.

In this paper we will start with a global overview of the Convex Meta Series, followed by a description of the current configuration installed at the EPCC. In the second part we concentrate on a PVM example solving the Poisson equation for a model problem. This program reflects many of the issues explicit message passing programs have to deal with. Performance results obtained on a Convex Meta Series system will be given.

2. The Convex Meta Series

The Convex Meta Series hardware consists of a C-series machine and a collection of HP PA-RISC processors. Any C2 or C3 system can be combined with HP730 or HP735 systems. Also a mixture of such HP processors is possible.

The processor components can be connected in three different ways: Ethernet, FDDI or the Convex developed Shared Memory Interconnect (SMI). The SMI combines low latency with a high bandwidth and is primarily meant for parallel PVM applications.

At the moment of writing, the Meta Series software Convex has added to the already available respective C-series and PA-RISC software consists of the following products:

- NQS+

The familiar NQS based CXbatch system has been ported to the HP environment. The joined batch environment is now referred to as NQS+. An important new feature are the so-called pull queues, where the least loaded processor will execute the job. The decision which processor to run on is not made when the job is submitted, but just before execution is to begin. This provides for transparent load balancing.

- Cluster tools

A suite of commands is available for both the system administrator and end-user to operate on the Meta Series as a whole. An example of such a useful command is "clsh". This will execute the specified command on the least loaded processor.

- MLIB

The well-known Convex VECLIB library has not only been ported to the PA-RISC platform, but also tuned for maximum performance. A growing set of modules is written in assembly language in order to fully exploit the potential of these processors.

- ConvexPVM

This is the Convex implementation of the popular PVM message passing system. It is available on both the C-series and PA-RISC systems.

Several Meta Series software projects have been initiated. Not only tools like profilers and debuggers for PVM applications are under development, but also load balancing is looked into.

3. The European Parallel Competence Centre

The European Parallel Competence Centre (EPCC) is located in Utrecht, The Netherlands and provides facilities to:

- Get familiar with the concept of the Meta Series and available software;
- Port and optimize an application;
- Obtain stand-alone performance results;
- Experiment with early releases of Convex products.

These services are not only available for Convex internal use, but also to customers in coordination with the local Convex support and sales staff.

At the EPCC a Meta Series system consisting of a C3220 plus 8 HP735 nodes connected over FDDI has been installed. In Figure 1 the configuration is shown.

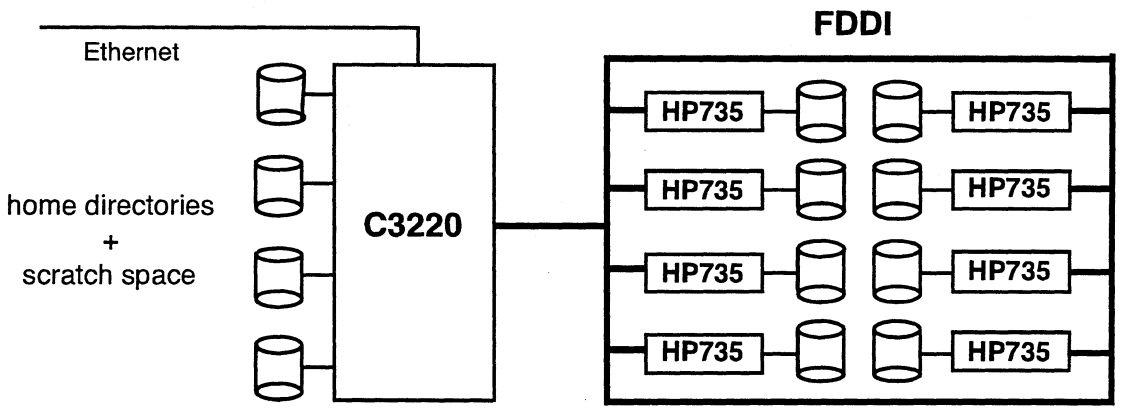


Figure 1. The EPCC Meta Series system.

As can be seen in Figure 1, the cluster is connected to the C3220 with a seperate FDDI network. The Ethernet network used is for office purposes.

Through NFS the home directories are made accessible from any processor in the Meta Series. The same holds for a scratch area. Note that each HP processor also has local diskspace. This is not only used for the system, but a part of it is available for local disk I/O performed by user applications.

4. PVM

PVM stands for Parallel Virtual Machine and is available through the public domain ([1]). PVM provides for a means to implement explicit message passing in Fortran and C programs meant to be run on a (heterogeneous) cluster of computer systems.

With PVM comes a controlling daemon program plus a set of libraries. Those libraries contain the modules supplying the functionality required to implement the message passing.

Important functions are of course those to be used for sending and receiving data. With PVM the approach taken is that on the sending side the programmer first has to fill a (PVM) buffer. Next the actual send function is called, where, amongst others, the identification of the target has to be given. The receiver will wait for the message to arrive. After having received this, the data has to be taken out of a buffer again. Multiple messages are distinguished by a so-called "message type". This can be any user-defined integer number. Also functions to perform broadcasts and explicit processor synchronization are provided. For a more detailed description we refer to [1].

5. A PVM application example

The program we will discuss in the remainder of this paper uses a so-called Generalized Red Black algorithm (or GRB for short) to solve the Poisson equation on the unit square.

The first parallel version of the GRB algorithm can be found in [2]. In [4] a slightly different variant is derived, leading to a higher level of parallelization. In [5] a faster implementation is discussed. Evans and Van der Pas et. al. ([3], [4], [5]) have reported performance results obtained on both shared and distributed memory systems.

Our first distributed memory implementation ([4]) was coded in the DPP84 assembly language and therefore not portable and also not very flexible. In this paper we will describe a new implementation coded in ANSI C, making use of the PVM message passing library, version 2.4 ([1]). Usage of PVM will not only enhance portability, but also make it easier to incorporate more features into the program.

In the next subsection of this paper the GRB algorithm will be derived for a model problem: the Laplace equation on the unit square with Dirichlet boundary conditions. In subsection 2 parallelization with PVM for distributed memory systems is discussed, followed by some performance results obtained on a Convex Meta Series system.

5.1. The GRB algorithm

In this subsection we will briefly outline the algorithm. More details can be found in [2], [4] and [5]. It should be noted that all formulas hold for the model problem only. At the end of this subsection it is indicated in what way the method might be generalized.

The model problem we will use is the Laplace equation on the unit square with Dirichlet boundary conditions:

$$\begin{aligned} -\Delta U &= 0 \quad \text{on } R := [0,1]x[0,1] \\ U &= g \quad \text{on } \delta R \end{aligned}$$

Using a standard 5-point discretization scheme for the Laplace operator with equidistant meshsizes $h_x = 1/N_x$ and $h_y = 1/N_y$ results in the following relationship for the unknown discretized solution value $U_{i,j}$ at grid point $x_i = (i-1)/N_x$ and $y_j = (j-1)/N_y$:

$$U_{i,j} - \alpha * (U_{i+1,j} + U_{i-1,j}) - \beta * (U_{i,j-1} + U_{i,j+1}) = 0 \qquad \begin{matrix} i = 2, ..., N_x \\ j = 2, ..., N_y \end{matrix} \qquad (1)$$

The coefficients α and β depend on the meshsizes only.

If we apply equation (1) at grid points (x_i , y_{j+1}), (x_{i+1} , y_{j+1}), (x_i , y_j), (x_{i+1} , y_j) and solve for the corresponding solution values $U_{i,j+1}$, $U_{i+1,j+1}$, $U_{i,j}$, $U_{i+1,j}$ we arrive at the following set of formulas:

$$\begin{aligned} U_{i,j+1} &= m1*R1 + m2*R2 + m3*R3 + m4*R4 \\ U_{i+1,j+1} &= m2*R1 + m1*R2 + m4*R3 + m3*R4 \\ U_{i,j} &= m3*R1 + m4*R2 + m1*R3 + m2*R4 \\ U_{i+1,j} &= m4*R1 + m3*R2 + m2*R3 + m1*R4 \end{aligned} \qquad (2)$$

The values m1,..,m4 depend on the meshsizes only. The values of R1,..,R4 are defined as follows:

$$\begin{aligned} R1 &= \alpha * U_{i-1,j+1} + \beta * U_{i,j+2} \\ R2 &= \alpha * U_{i+2,j+1} + \beta * U_{i+1,j+2} \\ R3 &= \alpha * U_{i-1,j} + \beta * U_{i,j-1} \\ R4 &= \alpha * U_{i+2,j} + \beta * U_{i+1,j-1} \end{aligned} \qquad (3)$$

It is easy to see that the unknown solution at the four grid points is expressed as a linear combination of the (also unknown) surrounding grid points. In Figure 2 formula (3) is depicted graphically.

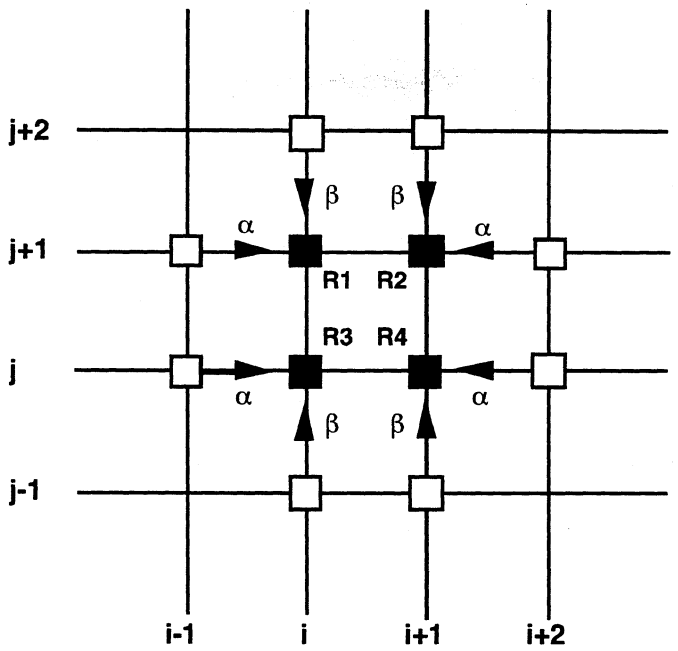


Figure 2. The calculation of R1,...,R4.

The next step we take is to apply an imaginary colouring scheme to the computational grid. The grid is divided into blocks consisting of four grid values each. Two different colours, say black and white, are applied to these blocks in a checkerboard manner. In Figure 3 an example is given.

This colouring scheme leads to a decoupling of the computations. Due to the way the solution is calculated in (2) and (3), computations in a block are independent of the calculations in the other blocks. This leads to the following algorithm:

1. Choose an initial solution U(0), defined on the internal grid points. Set k to 1.

2. Calculate new values at the black points, using the values at the white points. The update scheme is defined by (2) and (3).

3. Calculate new values at the white points, using the values at the black points. The update scheme is defined by (2) and (3). We now have a new solution, say U(k).

4. Test whether the stationary state has been reached by comparing the values of the previous iterand U(k-1) and the new value U(k). The algorithm is considered to have converged if U(k-1)-U(k) falls below a threshold value, measured in the l2-norm. If the algorithm is not yet converged, increase k by 1 and repeat steps 2 to 4.

5. The final solution is contained in U(k).

The GRB algorithm is highly parallel because the block updates within step 2 and 3 can be done in parallel.

The algorithm bears a strong ressemblance to the classical red-black scheme. The difference is that instead of a single point update, a block of four solution values is updated. Hence the name Generalized Red Black.

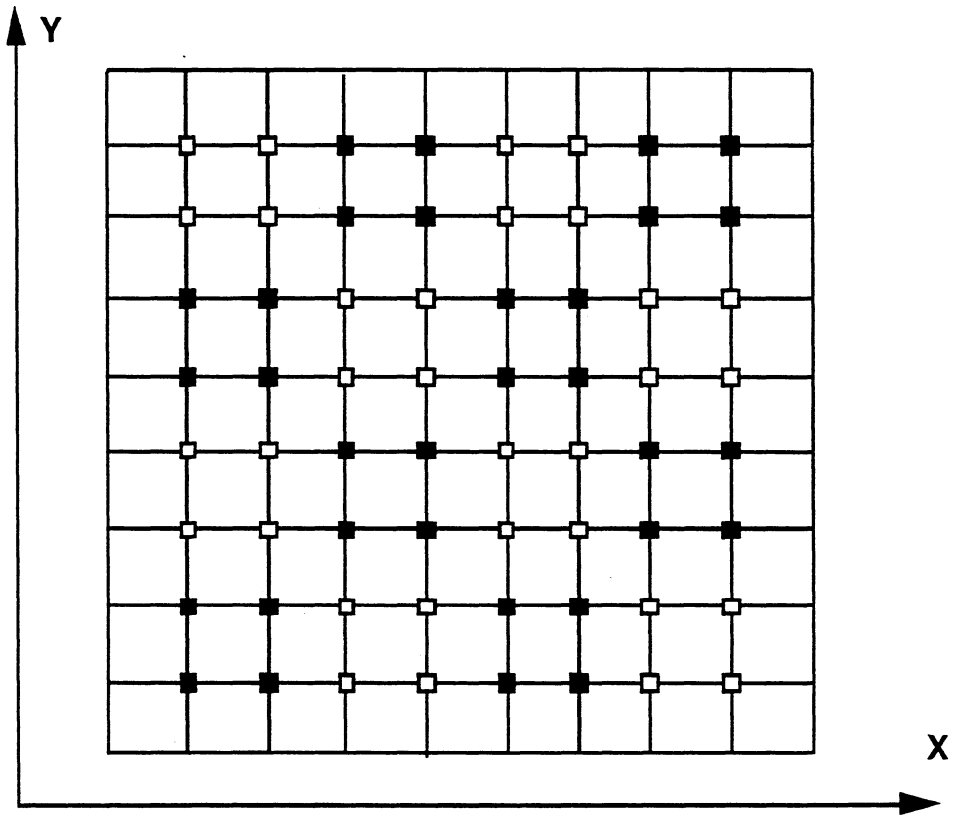


Figure 3. An example of the colouring of the grid points.

There are several obvious generalizations:

- The choice of 2x2 blocks is motivated by the results presented in [2], but of course the algorithm can be extended to blocks of size nxm.

- The algorithm has been derived for a simple model problem. It is straightforward to derive similar formulas for a more general problem. A major difference in the non-constant coefficient case would be to solve the linear subsystem numerically, as opposed to the analytical approach used for this model problem.

5.2 A PVM implementation

In the past we have reported on performance results obtained with two Fortran versions ([5]). Those implementations were easily vectorized and parallelized automatically and perform very well.

For this project it was decided to make an explicit distributed memory version under PVM. ANSI C was chosen as the programming language. The opportunity was also used to incorporate many features not available in the original version:

- Flexible with respect to the number of processors and physical configuration
- Support of options
- Support for continuation runs
- Timing overview of subparts of the program
- Data transfer overview
- Instrumented to run in verbose mode

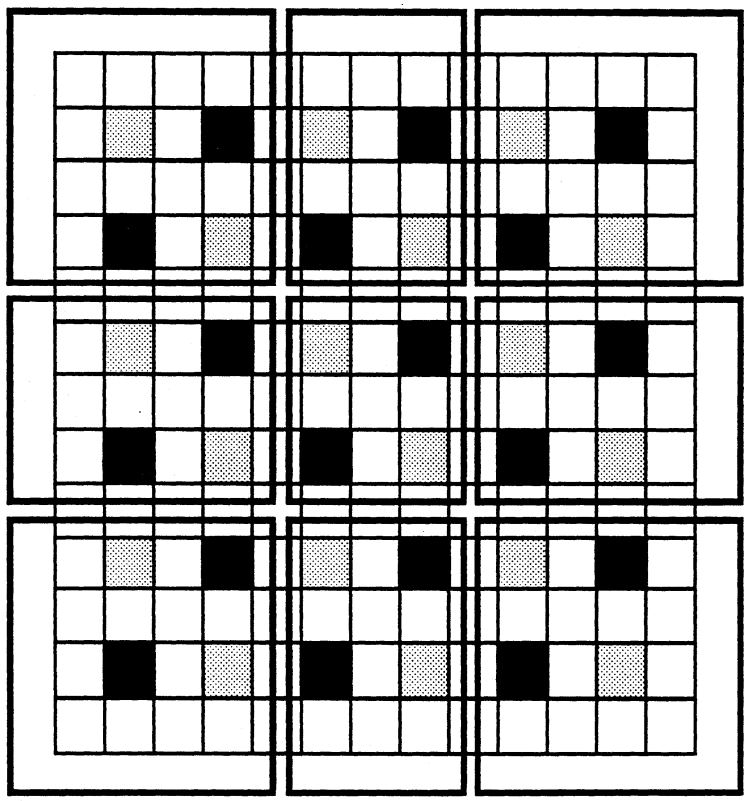


Figure 4. Example of domain decomposition.

In the previous subsection it was seen that each equally coloured block can be updated in parallel. Implementing parallelism at this level on a distributed memory system would probably result in a poor efficiency because it would require data transfer of 8 solution values present in the local memory of the neighbouring processors to obtain 4 new values only.

Instead we have applied domain decomposition. This means that an imaginary grid of processors has been placed over the computational grid. Each processor updates the solution corresponding to its part of the grid. At the boundaries of such local domains, processors have to exchange solution values before they proceed with the next iteration. In Figure 4 an example of domain decomposition is given.

The advantage of this approach is clear. The amount of data transferred is proportional to the perimeter of the local domain, whereas the number of points involved in the computations is a function of the area. This leads to a much more favourable computation to communication ratio.

We will conclude this subsection with a discussion of the more interesting aspects of the PVM implementation. In Figure 5 we have outlined the structure of the application.

During the initialization the host program has to send the following to each processor:

- The user options;
- A processor specific list of its neighbours;
- The relevant part of the initial solution.

Step	HOST part	NODE part
1.	User interaction	
2.	Preparation of data	
3.	Distribution of data	Receive data from host
4.		Update black blocks
5.		Exchange black boundary values
6.		Update white blocks
7.		Exchange white boundary values
8.	Receive local residuals	Send partial residual to host
9.	Test for convergence	
10.	Send flag to nodes	Receive flag from host
		if not finished repeat steps 4-10
11.	Receive local solutions	if finished send local data to host
12.	Merge local solutions	
13.	Store solution on file	

Figure 5. Outline of the structure of the application.

The first two sets of data are small and independent of the problem size, but the partial solution depends on the size of the problem solved and so can easily become quite large. We have chosen to use the synchronous "vsnd" function to send out the first two sets of data, but use the asynchronous "snd" to distribute the initial solution. This has the advantage that the host is available to start preparing the data package for the next processor as soon as the previous solution has been shipped. It does not have to wait for an acknowledgement from the receiving processor.

As soon as the nodes have received their initial data they can start with the first update of, say, the values at the black blocks. Next, they need to interchange boundary values with the neighbouring processors. This has been implemented with the asynchronous "snd" function. In order to ensure that the next half step (update of the solution at the white blocks) has the right values, we have used the "barrier" function to place an explicit synchronization point in the algorithm. After the white points have been updated and the boundary values have been exchanged, a similar synchronization point is inserted to ensure correct data is used for the next iteration.

Each processor then sends it local residual back to the host. This is done with the "vsnd". After the host has received all such local residuals it will determine the global residual and decide whether the algorithm has converged or not. If not, it will send a signal back to the nodes so that they can proceed with the next iteration. In case the final solution has been obtained a flag is sent to the processors so that they know they need to send back their part of the local solution. Sending this flag is performed synchronously i.e. with the "vsnd". Also the actual transfer of the local solution makes use of the "vsnd".

Upon arrival of the partial solutions, the host program merges them into the global solution. This solution is then written on disk for postprocessing purposes or a continuation run. Finally, each node sends some statistics back to the host, like timings for several subparts and number of bytes transferred and received. This small package is transferred using the "vsnd" function.

6. Performance results on a Convex Meta Series

In this section we will present and discuss some early performance results obtained.

It should be noted that for this first version no attempt has been made to optimize CPU or data communication performance. We have some ideas on how to obtain better absolute performance and will test this as soon as time permits.

All the experiments have been carried out on the above described EPCC configuration. The relevant characteristics at the time of testing were:

- A Convex C3220 with 256 MB of memory, 16 way interleaved;
- DKD-504 IDC disks;
- ConvexOS operating system, version 10.1.201;
- Convex "cc" C compiler, version 5.0;
- ConvexPVM version 1.0;
- 8 HP735 nodes, each with 144 MB of memory;
- Fast and Wide SCSI disks;
- HP "c89" C compiler, version HP92453-01 A.09.19;
- HP-UX operating system, version 9.0;
- FDDI interconnect.

All runs have been performed in September 1993. Because of practical reasons the C-series system was only a launcher and has not been involved in the computations.

With parallel programs of the type just described, the number of choices is overwhelming. Grid sizes, number of iterations, number of processors and processor lay-out are just some of them. We have tried to make a sensible choice without generating too much performance numbers.

It was decided to use the same number of grid points in both the X and Y direction. The number of processors in the Y direction has been set to 1, whereas for each problem size we have taken 2, 4 and 8 processors in the X direction. This means that we have used vertical processor strips. In Figure 6 an example is shown for 4 processors.

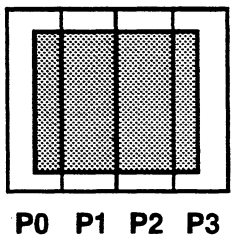


Figure 6. Example of vertical processor strips.

The following grid sizes have been selected: 256, 512 and 1024. For each grid size we have performed 50 and 100 iterations.

One might be tempted to measure the elapsed time of the algorithm only. This will in general show an incomplete performance picture because in production oriented programs not only the actual algorithm determines the time, but typically also pre- and postprocessing.

We do not pretend to have a complete program in that respect, but it does more than just raw computations. So, all the performance tables contain six columns with elapsed times, five of them representing different parts of the program:

PVM. This is the elapsed time it takes to enroll all the processors.

INIT. This is the time needed to prepare the local initial solutions and to send this, plus options and neighbour list, to the processors.

GRB. This is the actual time to solve the problem. Data transfer between the processors is included in this.

BACK. This is similar to the INIT column, only here we measure how long it takes to receive all data (final solution plus some statistics) from the processors.

STORE. Measures how long it takes to write the data on disk.

SUM. Sum of the previous five values.

In Figure 7 the results for a 256x256 grid are given. The times are always elapsed times and are given in seconds. The first four columns contain the grid size, the number of iterations and the number of processors in the X and Y direction respectively.

Size	Iter	Px	Py	PVM	INIT	GRB	BACK	STORE	SUM
256	50	1	1	0.21	0.89	7.67	0.46	0.31	9.52
		2	1	0.16	0.78	5.52	0.37	0.31	7.14
		4	1	0.23	1.03	4.25	0.28	0.31	6.10
		8	1	0.38	10.23	3.31	0.31	0.32	14.55
256	100	1	1	0.17	0.49	9.57	0.46	0.32	11.00
		2	1	0.18	0.62	7.36	0.37	0.31	8.84
		4	1	0.21	0.56	5.94	0.33	0.32	7.36
		8	1	0.37	10.25	4.72	0.31	0.32	15.96

Figure 7. Meta Series performance for a 256x256 grid.

The values in column "GRB" demonstrate that even for this relatively small problem a speed-up is realized.

As can be expected a slight increase in the time to enroll the processes is observed when the number of processors increases. The sharp increase in column "INIT" for 8 processors is not yet explained. A detailed study of the all the timings given revealed that it is the part where we send the options to all processors. Although not implemented as such, this is a broadcast operation and it seems to slow down dramatically.
Additional experiments (see also below) indicate that this increase appears more or less randomly, but always the in the part where we send the options to all processors. At this moment we do not have a reasonable explanation for this, but will certainly do more research to find out what is happening.

One would expect that the timings in columns "INIT" and "BACK" would be roughly the same. Figure 7 shows this is not observed. Perhaps the different send functions ("snd" and "vsnd" respectively) are responsible for this.

Another observation that made is that the actual time spent in the solver (column "GRB") does not scale linearly with the number of processors. This can partially be ascribed to a flaw in the current implementation that does not check for the actual number of processors involved. For a run on one processor this means that also the boundary exchange parts in the code are visited. Only just before actually preparing and sending the data it is detected there is no need to do communication. For the runs on more than one processor the behaviour is more complicated and not explained yet. Also for this behaviour further research will be required.

In Figure 8 the results for a 512x512 grid are listed.

Size	Iter	Px	Py	PVM	INIT	GRB	BACK	STORE	SUM
512	50	1	1	0.12	1.85	61.27	1.80	1.30	66.34
		2	1	0.22	1.78	33.32	1.44	1.28	38.04
		4	1	0.27	6.49	19.80	1.25	1.28	29.07
		8	1	0.56	6.04	16.30	1.20	1.28	25.38
512	100	1	1	0.13	1.86	166.73	1.80	1.29	171.81
		2	1	0.29	1.76	83.30	1.44	1.29	88.09
		4	1	0.38	6.48	61.73	1.27	1.27	71.13
		8	1	0.50	6.22	43.58	1.20	1.29	52.77

Figure 8. Meta Series performance for a 512x512 grid.

Note the speed-up on two and four processors. With 8 (or more) processors the serial part starts to dominate and the algorithm no longer scales.

Also for this problemsize an increase in column "INIT" is observed, but this time not as sharp as in Figure 7. The biggest mystery however is that when doubling the number of iterations, it takes more than two times longer to solve the problem ! This and the results from Figure 7 suggest there is an intricate coupling between perhaps solver time, communication time and cache usage.

Additional tests performed with the Fortran version ([5]) on a single HP735 processor reveal the same performance pattern. It looks as if cache usage plays a rather critical role in this and we will certainly do more investigations.

We conclude this section with results for a 1024x1024 grid. The numbers are given in Figure 9.

Size	Iter	Px	Py	PVM	INIT	GRB	BACK	STORE	SUM
1024	50	1	1	0.12	5.50	182.58	7.22	5.17	200.60
		2	1	0.18	6.87	92.42	5.75	5.15	110.37
		4	1	0.39	10.80	59.74	4.99	5.27	81.19
		8	1	0.46	10.30	42.82	4.65	5.16	63.39
1024	100	1	1	0.11	6.46	548.27	7.23	5.13	567.20
		2	1	0.18	6.89	295.43	5.76	5.15	313.41
		4	1	0.28	6.24	173.28	4.98	5.27	190.04
		8	1	0.52	6.64	142.54	4.66	5.14	159.50

Figure 9. Meta Series performance for a 1024x1024 grid.

The performance pattern for this problem size is quite similar to the one given in Figure 8.

The conclusion of this section is that the distributed memory version of the GRB algorithm, as discussed in this paper, is running faster if the number of processors increases, but that there are still some performance mysteries to be solved.

7. References

[1] Adam Beguelin, Jack Dongarra, Al Geist, Bob Manchek, Vaidy Sunderam, "A User's Guide to PVM Parallel Virtual Machine", ORNL/TM-11826,1992, Oak Ridge National Laboratory, Oak Ridge, Tennessee.

[2] D.J. Evans, "Parallel S.O.R. iterative methods", Parallel Computing, 1(1) 3-18, 1984.

[3] D.J. Evans and W.S. Yousif, "The implementation of the explicit block iterative methods on the Balance 8000 parallel computer", Parallel Computing, 16(1) 81-97, 1990.

[4] R.J. van der Pas and S.W. Brok, "A generalized Red Black algorithm and its implementation on the parallel processors DPP84 and Alliant FX/8", Report TR-28, ACCU, Utrecht, August 1988.

[5] R.J. van der Pas, "On the vectorization and parallelization of a finite difference scheme", in J.T. Devreese and P.E. van Camp, ed., Scientific Computing on Supercomputers III (Plenum Press, New York, 1991).

Parallel Distributed Computing in Quantum Chemistry

J.Fernández Sanz[(1)], H.P.Lüthi[(2)], A.Márquez[(1)], and M.Sierra[(3)].
(1) Dpt. de Química Física. Univ. de Sevilla. Sevilla. Spain. (2) ETH Zurich. Switzerland. (3) CICA Sevilla. Spain

A parallel distributed implementation of the RHF self-consistent field method, widely used in quantum chemistry, is presented. Details of this method, its parallelization, and performance, based on a quantum chemistry program running on a network of Convex computers, are also discussed.

1.Introduction.

The starting point for essentially all methods of *ab initio* molecular electronic structure theory is the Hartree-Fock or self-consistent-field method. Although it has long been recognized that, in many instances, it is necessary to go beyond the Hartree-Fock approximation and include the effects of the electronic correlation [1]; the SCF method continues to be of immense value in its own right.

Quantum Chemistry applications are the most important consumers of computer resources, including CPU time and input/output time, as well as main memory and disk storage. So they are good candidates for vectorization and/or parallelization.

Parallelization of electronic structure codes began in the earlier eighties with the work of Clementi and co-workers [2] on the loosely coupled array of processors (LCAP) at IBM Kingston. The integral and SCF modules of the IBMOL program were parallelized right at the beginning of the LCAP experiment. Later on, the work of Clementi was extended by Dupuis and co-workers by parallelizing the integral, SCF, and integral derivative modules [3] of the HONDO program [4,5] on the same LCAP system. However, the LCAP system was always an experimental computer, not normally available to the general scientific community.

A network of general purpose computer systems, interconnected by existing networks and support services, is a very attractive and viable alternative to special purpose hardware multiprocessors. Some network-based concurrent computing environments are specialized, in the sense that they are based on distributed operating systems, or that they support special-purpose programming paradigms [6-8]. While highly effective, they impose constraints on applications and end-users that are often difficult to meet.

Another class of distributed computing environments provide general purpose programming environments and require associated hardware and software at levels that can be considered standard. Programming paradigms based on procedure-call access to system facilities, limited inter-process communication in a machine, and network services based on unreliable data delivery are some characteristics of those distributed computing systems [9-13].

For the purpose of programming a parallel distributed implementation of the HONDO program system on a network of computers, early work of one of the authors (A.Márquez) with Dr. Michel Dupuis began in the IBM laboratory at Kingston. The distributed programming environment chosen was PVM, a widely available and general purpose concurrent computing system from ORNL [12-13]. The integral, SCF, integral derivative, and SCF (hyper)polarizabilities codes were parallelized on a network of RS6000 workstations [14].

In this paper we describe the parallelization of the integral and SCF modules of the program using the PVM system and its performance with representative samples in a Convex minisupercomputer.

2.The SCF equations.

Before going to the actual implementation of our serial and parallel SCF programs, let us revise the SCF basic equations and, in doing so, take a look at the major computational steps that can benefit from parallelization.

For our present purposes we will give a brief discussion of the SCF equations and for a detailed discussion we refer the reader elsewhere [15-17]. The following conventions have been adopted. Greek characters (μ, ν, λ, ...) represent atomic (AO) basis functions while latin characters (a, b, ..., i, j, ...) represent molecular orbitals (MO) basis functions.

For closed-shell systems, the RHF energy is given by

$$E_{SCF} = 2 \sum_{\mu\nu}^{n} D_{\mu\nu} (\mu|h|\nu) + \sum_{\mu\nu\lambda\sigma}^{n} D_{\mu\nu} D_{\lambda\sigma} [2 (\mu\nu|\lambda\sigma) - (\mu\lambda|\nu\sigma)] + V_{NUC} \qquad (1)$$

where n is the dimension of the atomic basis set, V_{NUC} is the nuclear repulsion energy, and the density matrix elements $D_{\mu\nu}$ are defined in terms of the molecular orbital coefficients by

$$D_{\mu\nu} = \sum_{i}^{n_{occ}} C_{\mu i} C_{\nu i} \qquad (2)$$

and the $(\mu|h|\nu)$ and $(\mu\nu|\lambda\sigma)$ are the one- and two-electron integrals respectively. In (2) n_{occ} is the number of occupied MOs. The molecular orbital coefficients are determined iteratively by diagonalizing the Fock matrix **F**, whose elements are given in the AO basis by

$$F_{\mu\nu} = (\mu|h|\nu) + \sum_{\lambda\sigma}^{n} D_{\lambda\sigma} [2(\mu\nu|\lambda\sigma) - (\mu\lambda|\nu\sigma)] \quad (3)$$

We should emphasize that (3) is expressed in terms of spatial orbitals rather than spin-orbitals and applies to closed-shell wave functions. The equations for UHF wave functions are similar and may be derived from the general spin-orbital expression[16,17].

3. The parallel algorithm.

In this section we shall describe the parallel implementation of the two most computationally demanding steps in the calculation of the SCF energy: the two-electron integral evaluation and the Fock matrix construction. Our discussion must be brief since this has been much discussed by Clementi and co-workers [2] and Dupuis and coworkers [3].

3.1 Two-electron integral evaluation.

In HONDO, as in other quantum chemical programs, the two-electron integral program is driven by loops over shells, irrespective of the details of how the blocks are computed. The evaluation of the integrals on one block is independent of that of other blocks and the program can be easily parallelized by distributing the loop over shell blocks over the different processors (Figure 1).

```
      SUBROUTINE INTGRL

      IJKLSH = 0
      DO 9000 I=1,NSHELL
      DO 8000 J=1,I
      DO 7000 K=1,I
      MAXL = K
      IF(K.EQ.I) MAXL = J
      DO 6000 L=1,MAXL
      ...
      IJKLSH = IJKLSH + 1
      IF(MOD(IJKLSH,NAP)+1.NE.IAP) GOTO 6000
      ...
6000  CONTINUE
7000  CONTINUE
8000  CONTINUE
9000  CONTINUE

      RETURN
      END
```

Figure 1: Distribution of the loop over shell blocks among NAP processors.

This technique produces a statistical distribution of the work over the processors and has been found to give an excellent load balancing and consequently the parallel efficiency is very high [3]. The different processes only need information about basis set, symmetry, and molecular geometry, and can compute the integrals independently for hours.

The output of the standard SCF program is a collection of files (one per processor) with an unordered sublist of AO integrals on the computer disk. In a direct SCF implementation, the integrals are contracted with the elements of the one-electron density matrix (eq. 2) as soon as they are computed, forming partial contributions to the Fock matrix on each of the processors.

3.2 Fock matrix computation.

As can be deduced from examinig equation (3), the Fock matrix elements depend linearly on the two-electron integrals. So, the computation of the Fock matrix can be easily parallelized having each process compute a partial contribution to this matrix from its own sublist of AO integrals.

Then in both the standard and direct SCF approaches, the total Fock matrix is formed by adding the contributions from each process in a global add operation. On the standard implementation of the SCF method, each processor reads from disk its own integral sublist on each iteration and builds from them its partial Fock matrix; on a direct implementation, the computation of the Fock matrix on each node is done on the fly, as the integrals are computed on each iteration.

The computation of the Fock matrix is asymptotically an $O(n^4)$ step, while diagonalization and other steps on the SCF iterations are, at the most, $O(n^3)$ steps. Experience shows that for reasonably large basis sets, that are good candidates for parallelization, the performance of the program is not affected by the serial execution of those $O(n^3)$ steps, and no attempt has been made for futher parallelization.

4. Implementation.

Although not enforced, the programming paradigm implicit in the philosophy of the PVM system is the host/node. In this paradigm, a master program (the host) starts *other small programs* (the nodes) to perform *specific tasks*. While the node programs are running, the host is usually in a idle state, waiting for results from the nodes to execute other, possibly serial, parts on the program. This was also the philosophy underlying the original LCAP system, although the terminology was somewhat different [2].

So, it seems that to implement a parallel version of the HONDO program system it is necessary to break the program into two separate modules (host and node) that will perform different roles in the final implementation. For example, one can think of a

small program whose task will be to compute the two-electron integrals given the necessary data from the host, and contract those integrals with the density matrix to form a contribution to the Fock matrix that is returned to the host. These matrices can, in turn, be added together by the host that also uses the full Fock matrix to compute the next density matrix.

However, one such implementation requires many changes in the serial program, related with the transfer of the necessary data (molecular geometry, basis set, density matrices, ...) between the nodes and the host. The logic of the program would be rather cumbersome and problems associated to maintenance of two different versions of the program will be created.

To avoid this situation, one of the objectives of the parallel implementation was to keep the necessary changes to a minimum, and that the necessary routines added to the

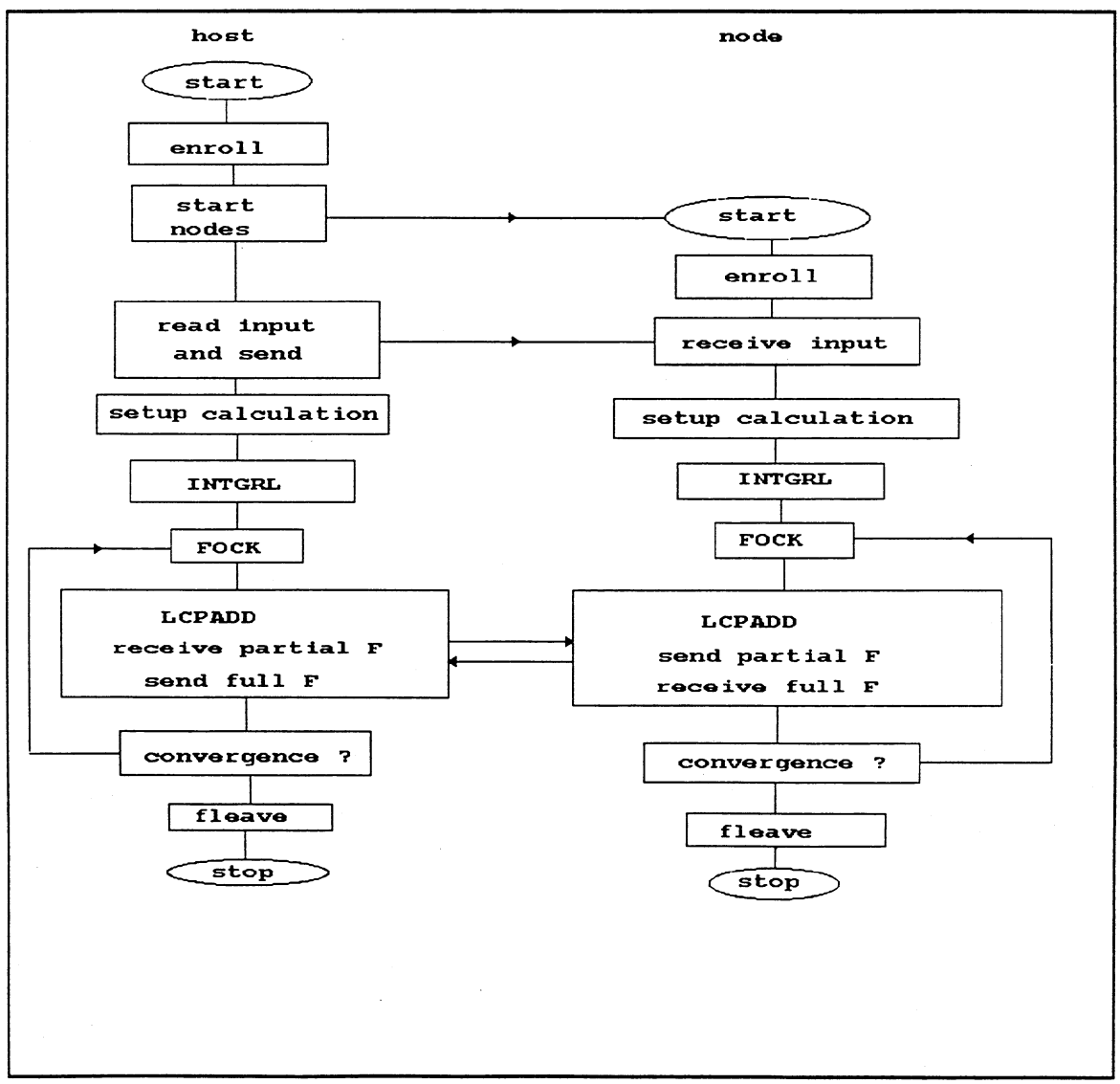


Figure 2. Structure of the parallel computation.

serial program have to be few and compact. This has been accomplished so that changes in the serial parts of the code do not affect its parallel version, and only a small number of subroutines held together in a single fortran module (pvm.f), are necessary to build the parallel PVM version of the HONDO program. Another result of the implementation is that changes in the serial code are independent of the parallel system used, as such details are hidden to the program.

Although we adapt the implementation to the host/node paradigm, there is only a single executable module that is host and node at the same time, and it performs different tasks only in selected points of the program. This way, the implementation is simple and easy to maintain. To help understand how the program works, Figure 2 shows schematically the structure of computation for both, host and node.

When the user starts the host, the program determines by reading the input file, how many processes are going to be used (nap), enrolls into the PVM system and starts nap-1 node programs. The node programs enroll in turn into the PVM system and wait to receive the input file from the host. This (the host) reads the input file and at the same time that it makes a working copy, sends it to the nodes. In this way all processes have the necessary information about the molecule to do the calculation.

At this point both host and nodes begin computation separatedly. They perform the necessary set-up of data (initial guess orbitals, one-electron integrals, etc.) and compute independently their share of the two-electron repulsion integrals on INTGRL. Although some work is repeated in the set-up step, no communication is necessary at this point and the structure of the program is kept simple and we expect that the performance of the parallel implementation is not affected.

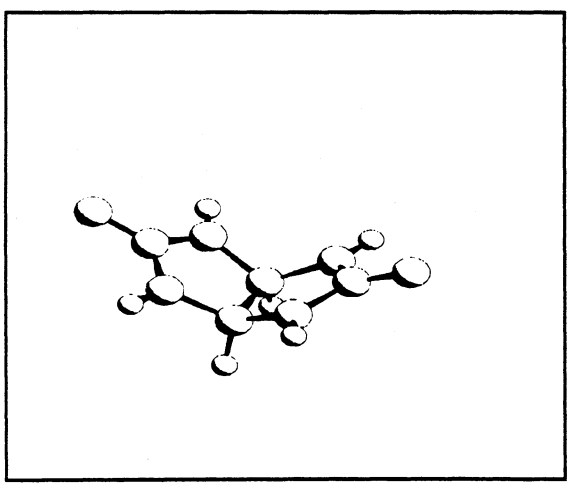


Figure 3. Molecular structure of $C_4N_4O_2H_6$ (sample 1).

After this task is completed the SCF iterations begin. First, the one-electron density matrix of equation (2) is computed and then each process (host and nodes) contracts this density matrix with its own sublist of two-electron integrals as in equation (3) to form an *incomplete Fock matrix* (routine FOCK). Then the node programs send their incomplete Fock matrices to the host, this adds all contributions together and sends the complete Fock matrix to the nodes (routine LCPADD). The SCF energy is computed, convergence is checked and a new SCF iteration begins, if necessary.

5. Results and performance.

The glycoluryl urea monomer, $C_4N_4O_2H_6$, whose structure is depicted in figure 3, was chosen as first sample molecule.

In Table 1, timings for a representative, medium-sized calculation on this molecule are presented. The number of processors used ranged from one to four and the efficiency of parallelization for the computation of the two-electron repulsion integrals, the SCF iterations, and the total calculation are given between parenthesis. A standard basis set of 6-31G** type was used for this test (180 basis set atomic orbitals).

Table I. Timing results for sample1 (standard SCF).

	time/seconds (efficiency %)[1]						
#proc	integrals		scf[2]		total		speed-up
1	1178.2	-	149.5	-	2413.8	-	1.00
2	601.5	(98)	90.8	(82)	1372.3	(87.9)	1.76
3	400.6	(98)	72.5	(69)	1019.3	(78.9)	2.37
4	303.9	(97)	62.2	(60)	842.1	(71.7)	2.87

1) Time in seconds on a C240 Convex computer. 2) Last iteration. Eight iterations were necessary for convergence.

These results show that the efficiency of parallelization in the computation of the two-electron integrals is very high (97-98%), even for this medium sized sample. The speed-up is nearly linear on this step, as can be seen graphically on Figure 4.

As expected, parallel efficiency of the SCF iterations is affected by the serial execution of the $O(n^3)$ steps as previously discussed. The efficiency of parallelization reduces to 60% for four processors, and the speed-up is not as good as on the integrals computation step. The speed-up of the total run is an average of both steps, as they dominate the computation.

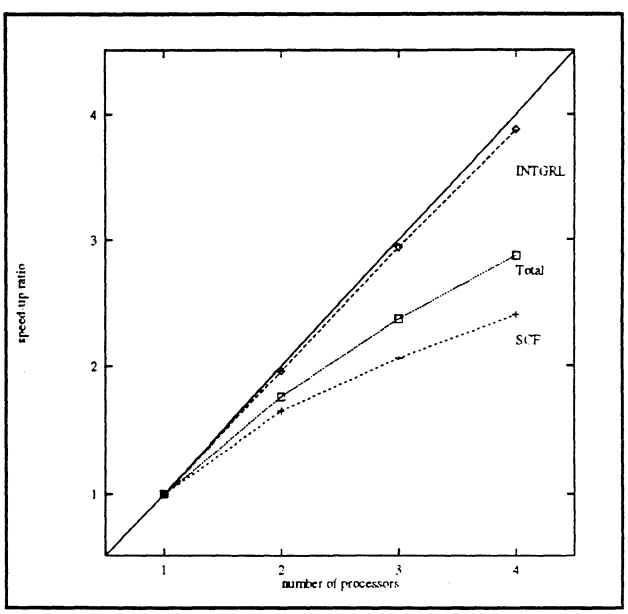


Figure 4. Speed-up for sample 1.

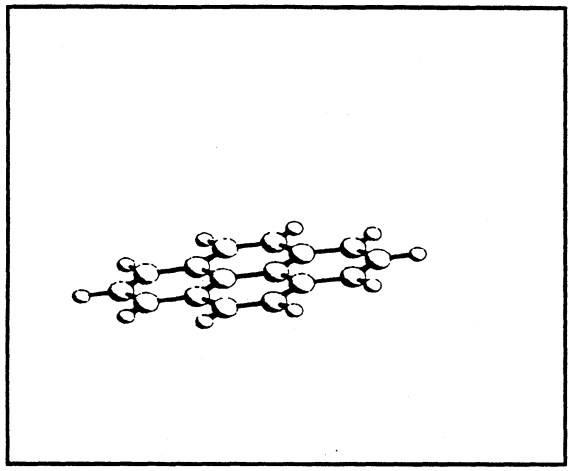


Figure 5. Molecular structure of pirene (sample 2).

To examine the performance of the parallel implementation of the program on bigger cases, we ran a second series of jobs. As second test sample the pirene molecule, whose molecular structure is sketched in Figure 5, was chosen. Using a standad 6-31G* basis, the size of the atomic orbital basis was 260 basis functions. Timing results for a SCF calculation on this sample are collected in Table II.

These results show an increased efficiency of parallelization on both the INTGRL and SCF steps, resulting from the increased basis set size. The total speed-up is, however, a little smaller, due to the incresed number of SCF iterations. For larger basis set it is expected that the performance of the program should be larger and calculation on molecular systems with hundreds of basis set can be performed in a reasonable amount of time.

Table II. Timing results for sample 2 (standard SCF calculation).

	time/second (% efficiency)[1]						
#proc	integrals		scf[2]		total		speed-up
1	2857.4	-	491.2	-	8370.20	-	1.00
2	1443.8	(99)	282.9	(87)	4715.70	(88.7)	1.77
3	998.4	(99)	234.0	(70)	3650.47	(76.4)	2.29
4	718.5	(99)	192.5	(64)	2954.42	(70.8)	2.83

1) Time in seconds on a C240 Convex computer. 2) Last iteration. Eleven iterations were necessary for convergence.

However, for molecular systems of this size, and bigger, the available disk space imposes the limit on the size of the calculation that can be performed using the standard SCF approach and the direct SCF technique should be used instead. In essence, a direct code simply recomputes the two-electron repulsion integrals when needed, avoiding quartic disk space requirements. Despite the greater computational cost associated with these algorithms, direct methods have become highly efficient and competitive due to improvements in integral evaluation codes and the rapid developments in CPU technology.

To check the performance of our direct SCF program, we run a series of tests jobs on

this same molecule. Using the standard 6-31G** basis set, the size of the basis was, in this case, 290 atomic orbitals. This is a more realistic sample of the type of calculation that are to be performed in parallel computers with a small number of processors. Larger jobs are the objective for massively parallel systems.

Table III presents timing results for a direct SCF calculation on this system. These results shows a good efficiency of parallelization as the serial steps make no significant contribution to the total time. Figure 6 presents graphically those results and as it can seen an excellent speed-up is obtained.

Table III. Timing results for sample 2 (direct SCF).

	time/minutes (% efficiency)[1]				
#proc	scf[2]		total		speed-up
1	77.3	-	621.06	-	1.00
2	40.7	(95)	326.87	(95)	1.90
3	28.0	(92)	225.02	(92)	2.76
4	21.5	(90)	172.52	(90)	3.60

1) Time in minutes on a C240 Convex computer. 2) Average time per iteration. Eight iterations were necessary for convergence.

6. Concluding remarks.

The preliminary results show that for basis sets, our implementation of the SCF method for computation of the electronic structure of molecular systems in the HONDO program system performs extremely well. Molecular systems of thousands of basis sets will soon become routine jobs using this technique on MPP computers. Progress is being made in the implementation of the system in a network of Convex minisupercomputers and personal workstations.

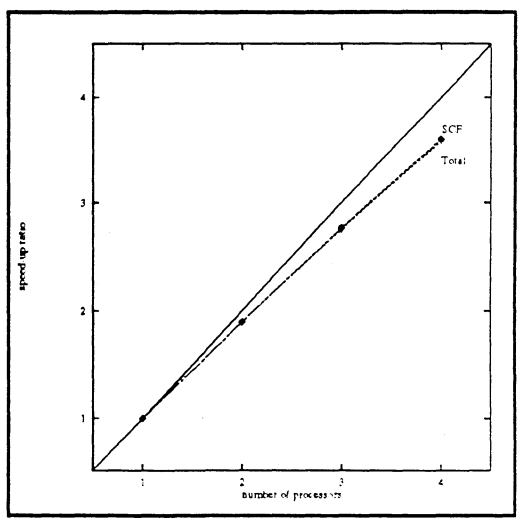


Figure 6. Speed-up for sample2.

7. Acknowledgement.

Part of this work is the result of the post-doctoral research of A.Márquez at IBM

Corporation in Kingston, New York, under supervision of Dr. Michel Dupuis. This work has been supported (in part) by Convex Supercomputer S.A.E. Support from the staff of the Centro Informático Científico de Andalucía (M.Sierra) is gratefully acknowledged.

7. References.

1. H.F.Schaefer, *The Electronic Structure of Atoms and Molecules: A Survey of Rigurous Quantum Mechanical Results*, Addison-Wesley, 1972.

2. E.Clementi, S.Chin, G.Corongiu, J.Detrich, M.Dupuis, L.J.Evans, D.Folsom, D.Frye, G.C.Lie, D.Logan, D.Meck and V.Sonnad. LCAP: Loosely Coupled Array of Processors Parallel Processing Systems, in *Modern Techniques in Computational Chemistry, 1991*, E.Clementi, Ed. ESCOM 1991; and references therein.

3. M.Dupuis and J.D.Watts, *Theor.Chim.Acta,* **71** (1987) 9; see also J.D.Watts, M.Dupuis and H.O.Villar, IBM Technical Report KGN-78, 1986.

4. M.Dupuis, D.Spangler, and J.J.Wendoloski, NRCC Sofware Catalog., Volume 1, Program Number QG01 (1980).

5. M.Dupuis, S.Chin, and A.Márquez. Chem-Station and HONDO: Modern tools for electronic structure studies including electron correlation. in *Relativistic and Electron Correlation Effects in Molecules and Clusters*, G.L.Malli Ed., NATO ASI Series, Plenum Press, NY 1993.

6. G.Popek, B.Walker, *The Locus Distributed System Architecture*, MIT Press, Cambridge, 1985.

7. D.Cheriton, Com. ACM, **31** (1988) 314.

8. M.Arango, D.Berndt, N.Carriero, D.Galernter, and D.Gilmore, Supercomputing Review, **3** (1990).

9. M.Sullivan, D.Anderson, *Proc. 9th Intl. Conf. on Distributed Computing Systems* 1989, p. 181.

10. T.J.Gardner et al. *DPUP: A Distributed Processing Utilities Package*, Computer Science Technical Report, University of Colorado, 1986.

11. A.Bequelin, J.J.Dongarra, G.A.Geist, R.Manchek, V.S.Sunderam; *Heterogeneous Network Supercomputing.* Supercomputing Review, August 1991.

12. V.S.Sunderam; PVM. A Framework for Parallel Distributed Computing. *Concurrency: Practice and Experience*, **2** (1990) 315.

13. V.S.Sunderam; Concurrent Computing with PVM. Cluster Computing Workshorp. Florida State University, 1992.

14. M.Dupuis and A.Márquez; *HONDO 8.4 from CHEM-Station Documentation.* IBM Corporation, Kingston, NY, 1992.

15. C.C.J.Roothaan; *Rev.Mod.Phys.* **23** (1951) 69.

16. A.Szabo and N.S.Ostlund; *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory.* McGraw-Hill, NY 1983.

17. J.A.Pople and R.K.Nesbet; *J.Chem.Phys.* **22** (1954) 571.

A fair view of MPP parallelism

Greg Astfalk
CONVEX Computer Corp.
PO Box 833851
Richardson, TX 75083-3851
email: astfalk@convex.com

September 8, 1993

1. Introduction

MPP is an emerging technology that is quite likely to impact us all at some future date. There is a great deal of expectation for this technology to arrive. Certainly the adjective "massive" is a major part of MPP. The thing that needs clarification is the arithmetical value of "massive." The notion that an MPP must have a very large number of processors to qualify as an MPP is somewhat self-defeating. There have appeared, in the past, several views that stated that 10 or fewer processors is "modestly" parallel, 100 processors is "moderately" parallel, and that 1000's of processors are required to constitute "massive." We don't subscribe to this. Taking an inverse perspective it should be viewed that MPP is a technology that will allow the machine to scale to a size, measured in processors, that is sufficient to minimize the time-to-solution for a given problem.

Consider that your application is too large to be effectively performed on a high-end workstation or perhaps even on a supercomputer. If that is the only class of machine at your disposal then your problem solving ability is constrained. If however you are using an MPP then you could, in principle, increase the size of the MPP to accommodate the problem[1]. Do you need to have tens, hundreds, or thousands of processors? The correct answer is that you need as many as your problem can effectively use. As we will see shortly it is not true that all problems will scale up to run efficiently on a "massive" number of processors. We believe that the precise number that constitutes massive is illusive and not worth the effort to quantify. The reasoning is that the answer is highly dependent on the architecture, application, algorithm, code, and problem size. With such a multi-dimensional space, rules of thumb and simple models will often lead us astray.

2. Speed-up

In the early parts of this paper we will be considering an abstract application executing on a parallel computer. This abstract application has a sequential part and a parallel part to its

[1]This implies that the problem's solution time is reduced by the addition of more processors: we discuss this later.

execution; in this regard it is not at all abstract. Let the time taken by the parallel portion of the program be given by $t_p(n,p)$. Here, and in what follows, p indicates the number of processors that are being used and n is some measure of the size of the problem (*i.e.*, number of particles, number of equations, etc.). There is also a sequential part (all real programs have a sequential part) whose time is given as $t_s(n,p)$. Obviously the time of execution of the code is;

$$t(n,p) = t_s(n,p) + t_p(n,p) \tag{1}$$

Rather than deal with the solution times themselves it is easier and more instructive to focus how much faster our application will run on some number of processors. It is common practice to do this by using the speed-up. The definition of speed-up is simply;

$$S_u = \frac{t(n,1)}{t(n,p)} \tag{2}$$

This is simply the time taken by 1 processor divided by the time taken on p processors. Note that there are a substantial number of alternate ways of defining the speed-up [3]. We will neglect the controversial and confusing issue of super-linear speed-up[(2)]. This constrains our speed-up to $1 \leq S_u \leq p$. We will retain this form rather than normalizing the equation as is often done in the literature. The reason will be apparent later.

If we allow for perfect parallelism in the parallel execution, that is the p threads of parallel execution are not doing redundant work and they are loosely synchronized (in the sense that they are starting and stopping their execution at the same times) then the wall-clock time of the parallel portion is given as t_p/p. Additionally, we further assume that we retain the same size problem when we solve it on p processors. Given these assumptions the speed-up is;

$$S_u^{(f)} = \frac{t_s(n,p) + t_p(n,p)}{t_s(n,p) + t_p(n,p)/p} \tag{3}$$

We use a superscript of (f) to denote this as the fixed problem size speed-up. The only issue is that the parallel execution time is reduced by a factor of p owing to the concurrent execution on the p processors. This is the original, fixed problem size, form of Amdahl's Law [1].

In the previous development we retained the same problem size as we applied an increased number of processors. A more contemporary view of this has been proposed [7]. In this case we scale the size of the problem as we add processors. The graphic shown in Figure 1 should help to make the difference clear. In the scaled variant we increase the size of the problem by a factor of p and then we run it on the p processors. There are some subtle issues about both of these perspectives that we defer to a later section.

In light of this scaled problem size approach the definition of speed-up now becomes

$$S_u^{(s)} = \frac{t_s(n,p) + p \cdot t_p(n,p)}{t_s(n,p) + t_p(n,p)} \tag{4}$$

[(2)]This is often reported but it is an issue of architecture and algorithm synergy rather than the more obvious architecture-only issue that we are concerned with here.

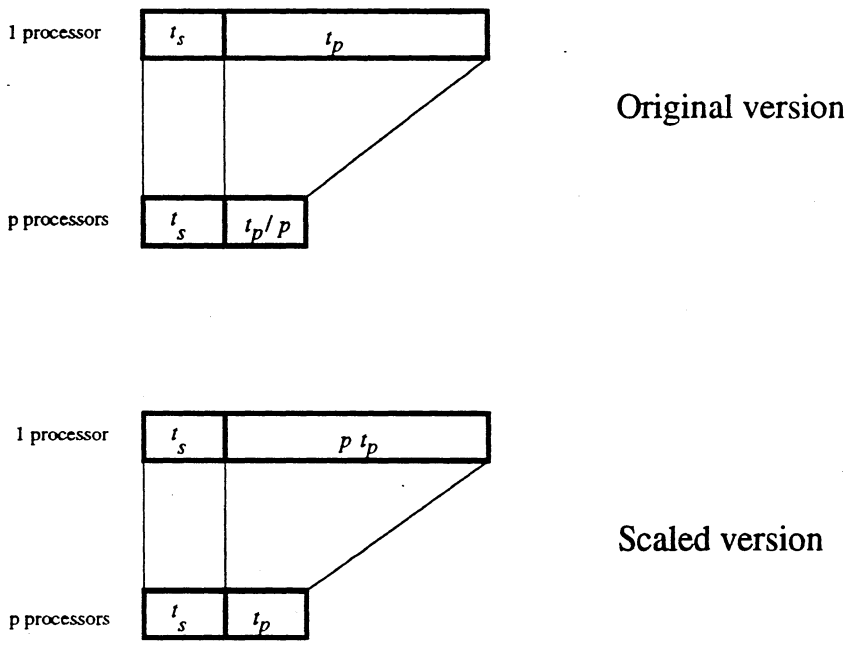


Figure 1: Graphic representation of the two versions of Amdahl's Law.

In a similar vein as before the (s) superscript denotes the scaled problem size speed-up. Note that the numerator is still the time it takes for executing on a single processor; the problem size has been increased by a factor of p. In the denominator this parallel time has been reduced by a factor of p owing to the application of p processors to the problem. The thesis of the scaled problem size version is that when we have access to a larger computer we are more likely to run a larger problem. This is probably closer to reality than the fixed problem size view but even it is flawed in actual practice.

3. Parallel overhead

The parallel execution of virtually all applications will require some communication or synchronization between the parallel threads. This is a direct result of the nature of how problems must be partitioned, and yet cooperate, in order to be solved in parallel. If the parallel threads had no need to communicate with each other or share data then there would be little point in addressing them with an MPP. We need to allow for a certain amount of communication but since all applications are different we can not, in general, state exactly how much will be present. Let's denote this communication time, which is really overhead, as $t_o(n,p)$. This is time that is additive to the parallel execution time. It is understood that $t_o(n,1) \equiv 0$. Folding this into the equation for the fixed problem size speed-up, equation (3), it is now

$$S_u^{(f)} = \frac{t_s(n,p) + t_p(n,p)}{t_s(n,p) + t_p(n,p)/p + t_o(n,p)} \tag{5}$$

Similarly we now have, for the scaled problem size speed-up, equation(4),

$$S_u^{(s)} = \frac{t_s(n,p) + p \cdot t_p(n,p)}{t_s(n,p) + t_p(n,p) + t_o(n,p)} \tag{6}$$

Fixed problem size	
$t_o(p)$	$\lim_{p\to\infty} S_u$
0	$\frac{t_s+t_p}{t_s}$
α	$\frac{t_s+t_p}{t_s+\alpha}$
$\alpha \ln(p)$	0
αp	0
$\alpha p \ln(p)$	0
αp^2	0

Table 1: The limit of the speed-up, $S_u^{(f)}$.

The first thing to note is that $t_o(n,p) > 0\ \forall n,p$. Therefore the speed-up, either $S_u^{(f)}$ or $S_u^{(s)}$, that we achieve is always going to be *reduced* by the communication overhead. This is in fact the focus of this paper; the effect of the communication overhead.

4. Effects of communication

We want to focus on the effect of this communication overhead as we add processors. To get some sense of the behavior of our speed-up we will allow for several different dependencies of $t_o(n,p)$ as a function of p alone. Since it is impossible to be specific about the amount of communication we will consider six functional representations for the overhead. In particular we will treat:

$$t_o(p) = 0$$
$$t_o(p) = \alpha$$
$$t_o(p) = \alpha \ln(p)$$
$$t_o(p) = \alpha p$$
$$t_o(p) = \alpha p \ln(p)$$
$$t_o(p) = \alpha p^2$$

Here α is a constant with units of time. We can consider it to be the time taken for a *single* communication event[3]. Note that each of the six forms for the overhead is increasingly large.

4.1 Fixed problem size

For each of the 6 forms of $t_o(p)$ we can take the limit of $S_u^{(f)}$ as $p \to \infty$. From Table 1 it is apparent that any communication, other than a constant amount, is going to cause problems in the limit of a very large number of processors. Practical considerations will certainly limit us to far less than an infinite number of processors. However the point is that communication can change the prospect of the scalability of an algorithm.

[3] We are careful not to call this a "transaction" nor a "message" since both these terms carry a connotation that is too restrictive.

Fixed problem size	
$t_o(p)$	p_{max}
0	∞
α	∞
$\alpha \ln(p)$	t_p/α
αp	$\sqrt{t_p/\alpha}$
$\alpha p \ln(p)$	$p^2 [1 + \ln(p)] = t_p/\alpha$ †
αp^2	$[t_p/2\alpha]^{1/3}$

Table 2: The maximum number of processors that can be utilized. († This case does not have a closed form solution; it is necessary to solve this nonalgebraic equation.)

Fixed problem size, α = 100 microseconds			
$t_o(p)$	p_{max}		
	t_p = 6 minutes	t_p = 1 hour	t_p = 10 hours
0	∞	∞	∞
α	∞	∞	∞
$\alpha \ln(p)$	3.6×10^6	3.6×10^7	3.6×10^8
αp	1897	6000	18974
$\alpha p \ln(p)$	691	2043	6087
αp^2	122	262	565

Table 3: The maximum number of processors that can be utilized.

We further explore the communication overhead by determining the maximum number of processors that can be effectively used. This is achieved by simply differentiating $S_u^{(f)}$ with respect to p and equating to zero. In determining these numbers we relax the integrality condition on p and assume it is continuous. Each of the six cases are covered in Table 2.

From Table 2 it is clear that for those cases with more than a constant amount of communication overhead there is a definite maximum number of processors that can be efficiently utilized. We see that the maximum number of processors that can be utilized is a function of only t_p and α. Rather than deal with the algebraic forms for the maximum speed-up shown in Table 2 we instead produce a table for a realistic range of values for t_p and α. We assume that the communication event time, α, is 100 microseconds and we allow for t_p to be 6 minutes, 1 hour, and 10 hours.

What is surprising, and perhaps disconcerting, is that the maximum number of processors that can be utilized is quite low, relatively speaking. If we examine each column in Table 3 we note a rapid decrease in the maximum number of usable processors as the communication complexity increases. This is a clear indication that inter-processor communication has a strong effect on the scalability of algorithms on an MPP. A final look across the columns in Table 3 shows that even as the total parallel execution time is increased by a substantial amount, if there is a "large" amount of communication we don't gain much advantage in terms of the maximum number of usable processors.

Fixed problem size			
$t_o(p)$	$\mid S_u^{(f)} \mid_\infty$		
	t_p = 6 minute	t_p = 1 hour	t_p = 10 hours
0	6.96	57.62	378.96
α	6.96	57.62	378.96
$\alpha \ln(p)$	6.96	57.62	378.95
αp	6.95	57.53	378.55
$\alpha p \ln(p)$	6.89	56.99	376.15
αp^2	6.52	45.41	231.69

Table 4: The maximum achievable speed-up, $S_u^{(f)}$.

Scaled problem size	
$t_o(p)$	$\lim_{p\to\infty} S_u^{(s)}$
0	∞
α	∞
$\alpha \ln(p)$	∞
αp	t_p/α
$\alpha p \ln(p)$	0
αp^2	0

Table 5: The limit of the speed-up, $S_u^{(s)}$.

As the next logical step we can examine the maximum speed-up that is achievable under the same circumstances. In order to compute $S_u^{(f)}$ we need to assume a value for t_s and p. So that we don't get completely bogged down in exploring the parameter space let's use the same values of t_p and α as in Table 3 and assume that t_s = 1 minute. Recall that the values of t_p sets the ratio of parallel to sequential time at 6, 60, and 600. It could be argued that other numbers might be more appropriate but that isn't our goal here. We allow for p to be everyone's emotional favorite of 1024. Table 4 thus illustrates that the speed-up is largely determined by the relative values of t_s and t_p, as you would expect. When the disparity between them is large then the added burden of the communication overhead is not significant. However as we increase the parallel time relative to the sequential time we note that the communication begins to show an increasingly detrimental effect on the achievable speed-up.

4.2 Scaled problem size

We repeat the developments from the previous section only now we work from the scaled problem size model. The structure of the communication term, t_o, does not change from the previous development. Therefore we will dealing dealing with Equation (6). First we examine $\lim_{p\to\infty} S_u^{(s)}$ the results of which are in Table 5.

Comparing Table 1 and Table 5 we see the expected better behavior of the scaled problem size model relative to the fixed problem size model. In spite of the generous scaling (proportional to

Scaled problem size	
$t_o(p)$	p_{max}
0	∞
α	∞
$\alpha \ln(p)$	∞
αp	∞
$\alpha p \ln(p)$	$\frac{t_s t_p + t_p^2 - \alpha t_s}{\alpha} = t_s \ln(p) + p t_p$ †
αp^2	$\frac{\sqrt{t_s^2 + t_s t_p^2/\alpha + t_p^3/\alpha} - t_s}{t_p}$

Table 6: The maximum number of processors that can be utilized. († This case does not have a closed form solution; it is necessary to solve this nonalgebraic equation.)

Scaled problem size			
$t_o(p)$	p_{max}		
	t_p = 6 minutes	t_p = 1 hour	t_p = 10 hours
0	∞	∞	∞
α	∞	∞	∞
$\alpha \ln(p)$	∞	∞	∞
αp	∞	∞	∞
$\alpha p \ln(p)$	4.2×10^6	3.6×10^7	3.6×10^8
αp^2	2049	6050	18989

Table 7: The maximum number of processors that can be utilized.

p) of the parallel portion of the execution we still see that the communication effects of higher than linear complexity in the number of processors could still drive the speed-up to zero.

The maximum number of processors that can be used for the scaled problem case is shown in Table 6. As we did before we choose not to dwell on the algebraic forms for p_{max} results except to note that with the scaled problem size model we can incur a large amount of communication and still utilize a large number of processors. One distinct difference here is that, unlike for the fixed problem size model, p_{max} also depends on t_s.

In Table 7 we show the number of processors that can effectively be utilized when we assume (as before) a communication event time, α, of 100 microseconds, sequential execution, t_s, of 1 minute, and parallel execution time, t_p, of 6 minutes, 1 hour and 10 hours. Clearly these numbers are essentially infinite relative to the number of processors that are in the MPPs of today. Thus, if we adopt, and can justify, the scaled problem size model, then the communication overhead for the parameters we have chosen presents little restriction to scalability. It is still an open question if the application can in fact be scaled in proportion to the number of processors that are applied.

Our final consideration in this section is that of the maximum achievable speed-up for the scaled problem size model with communication. We stay with the parameter values we had before[4] and form Table 8. Without any detailed discussion it is apparent that the scaled problem size

[4]These are $t_s = 60$, $p = 1024$, and $\alpha = 10^{-4}$.

Scaled problem size			
$t_o(p)$	$\mid S_u^{(s)} \mid_\infty$		
	$t_p/\alpha = 3.6 \times 10^7$	$t_p/\alpha = 3.6 \times 10^6$	$t_p/\alpha = 3.6 \times 10^5$
0	877.86	1007.22	1022.30
α	877.86	1007.22	1022.30
$\alpha \ln(p)$	877.86	1007.22	1022.30
αp	877.64	1007.20	1022.29
$\alpha p \ln(p)$	876.38	1007.03	1022.28
αp^2	702.48	979.18	1019.33

Table 8: The maximum achievable speed-up, $S_u^{(s)}$.

model allows us to achieve substantial speed-ups. This is true even for relatively small parallel times, relative to the sequential time.

5. Real communication latencies

In the previous developments we used a communication event time, α, equal to 100 microseconds. Clearly this was a very rough approximation. In this section we will be more detailed and specific about the communication times. A somewhat simplistic model for a communication event between processors is

$$t_{comm} = t_l + W \cdot t_w + H \cdot t_h \tag{7}$$

In this equation t_{comm} is the time taken in communicating some data between processors, or processor and memory. t_l is the overhead per communication event. t_w is the time required to move a data item which, for consistency, we define to be a 64-bit floating-point operand. W is the number of such words that are in the communication event. t_h is the "hop" time that is required to transit an intervening node in the message path. H is the number of intervening nodes in the path (note that for a direct path with no intervening nodes $H = 0$). As a very broad characterization of the terms in this equation we can attribute the first as a latency term, the second as a bandwidth-like term, and the last as a "topology" effect.

Again we need to emphasize that this is a simple model. The simplicity of the model is evident in a number of ways. As a specific example, t_w for the CM-5 is not a constant value but rather a function of the message size and the path length. Additionally the bandwidth-like term, t_w, does not directly account for contention on the communication channel(s). We could generalize the equation somewhat by allowing the time terms, t_l, t_w and t_h to be functions of W and H. In spite of this the model can be put to profitable use. We show this in a subsequent section.

From the open literature [4] [5] [6] [9] we can construct a table, Table 9, that indicates some of the numbers for these communication parameters. In Table 9 we have added t_c which is the cycle time for the host processor in each of the listed MPPs. An important thing to notice in this table is the ratio between t_l and t_w. This ratio will indicate to us the size that W must

Communication times (μseconds)					
Machine	CPUs	t_l	t_w	t_h	t_c
iPSC/860	128	136	1.60–2.86	2.00	0.0250
Delta	512	50	0.32–0.47	0.05	0.0200
Paragon	2048	10	0.04	0.03	0.0125
nCube-2	8096	60	1.60	2.00	0.0500
CM5	1024	88–350	1.01–3.60	0.50	0.0330
IBM SP1	64	50–200	1.00–1.33	–	0.0160
KSR	1088	6.7–30	0.24–0.41	?	0.0500
CONVEX	128	< 3	–	–	0.0100

Table 9: Communication times for several existing MPP machines.

attain in order for the latency to be at parity with the bandwidth term. Without proof[5] we state that the size of W is small ($\mathcal{O}(1)$) in the case of most current algorithms.

Let's perform a thought experiment to put some of the preceding discussion into perspective. It is a rather well accepted fact that the shared-memory vector supercomputers can achieve a respectable percentage of peak performance from a high level language. Why is this so? On the shared-memory vector supercomputers the memory subsystem is a very sophisticated part of the machine. The memory is highly interleaved to avoid busy-bank waits and the cross-bar is capable of very high bandwidths and it is generally non-blocking. Under conditions that are, in general, easy to achieve, *i.e.*, vectorized computation, the latency to get a piece of data from the memory into the vector register ready for computation is $\mathcal{O}(1)$ processor clocks. Even if the calculations are such that the $\mathcal{O}(1)$ latency doesn't apply then we still only suffer a latency of a few 10's of processor clocks. If we consider a cache-based RISC engine then the latency is the time to load a cache line from memory into the cache itself. For a contemporary RISC engine this is something like 30 clock cycles. This is clearly not the situation on an MPP as the data in Table 9 indicates (compare t_l and t_c). Attention must be paid to this aspect of an MPP architecture.

6. A specific application

To bring the previous discussion(s) into better focus let's consider a specific problem: a 3-dimensional FFT[6]. We will neglect mathematical rigor since it will suffice for this discussion to note that we need to perform a 1-dimensional FFT in each of the three dimensions on each "stripe" of data for the corresponding directions. That is, if we assume the data array size to be n_x by n_y by n_z then we will need to do $n_x\, n_y$ FFTs of length n_z, $n_x\, n_z$ FFTs of length n_y, and $n_y\, n_z$ FFTs of length n_x.

In the sequel we will assume;

- a 1-d FFT routine operating on contiguous data is available
- we have $p = \max(n_x, n_y, n_z)$ processors

[5]This is detailed in as yet unpublished work [2].

[6]This treatment is similar to [8] but more specific in its findings.

- $\max(n_x, n_y, n_z)\ mod\ p \equiv 0$
- each of the p processors has sufficient memory to hold "plane" of data
- each datum is 64-bits
- the inter-processor communication network is sufficiently rich that it can support one-to-all communication without contention
- the time to "move" a datum within a processor's local memory is equivalent to the time it takes to do a floating-point operation
- the computational complexity of the FFT is given as $\delta\, n \log_2(n)$, where n is the length of the FFT and δ is a constant that depends on the particular FFT implementation; we choose $\delta = 5$.

The obvious layout for the data is in a rank-3 array, say **x(nx,ny,nz)**. A high level pseudo-algorithm to accomplish the task is;

```
c First phase
      for i = 1,nz
         do ny FFTs of length nx
      end
      for i = 1,nz
        transpose data arrays
      end
      for i = 1,nz
         do nx FFTs of length ny
      end
c Second phase
      transpose data arrays
c Third phase
      for i = 1,nx
         do ny FFTs of length nz
      end
```

The obvious partitioning of the data is to take a plane of data, say the (x, y) plane and assign each of the n_z such planes to the memories of the p processors. Now on each processor, that is for each of the n_z (x, y) planes, we need to perform n_y FFTs of length n_x on the rows. Following this we will need to transpose the data associated with the plane in order to bring the current columns into the rows (to insure contiguous data for our FFT engine). This transpose requires $\mathcal{O}(n_x\ n_y)$ moves on each of the n_z planes. With the initial data partitioning having been done by (x, y) planes to the various processors it is obvious that this set of transpositions is done within each of the processor's local memory. Following the transposition we perform n_x FFTs of length n_y on what used to be the columns; this is done on each of the n_z planes.

Now it remains to do the z-direction FFTs. Obviously this will require another transposition of the data. In order to accomplish this transpose, on the parallel machine, it is going to be necessary to use the inter-processor interconnect. Recall that each processor holds a (x, y) plane of data. In order to achieve contiguous z-direction data in the local memory of each processor will require all processors to send some of their data to other processors. This is going to require $\mathcal{O}(n^2)$ "messages" each of length n to accomplish.

We now have sufficient information to quantify the computational and communication complexity of the algorithm. Without loss of generality we set $n_x \equiv n_y \equiv n_z = n$. On a single processor

the time required for the entire algorithm is

$$t(n,1) = t_c \left(3\delta\, n^3 \log_2(n) + 2n^3\right) \tag{8}$$

In this equation the first term is (obviously) the FFTs computational complexity while the second term is that of the transpose operation(s).

On the parallel processor each of the processors will be doing the FFTs on the plane of data that it "owns." This means that on any given processor the time will be, for the first phase of the algorithm;

$$t^{(1)}(n,1) = t_c \left(2\delta\, n^2 \log_2(n) + n^2\right) \tag{9}$$

Since each processor has its own data it is only necessary to note that the overall time-to-solution for the first phase is equal to;

$$t^{(1)}(n,1) = \max(t(n,1)) \;\; \forall p \tag{10}$$

Since n is the same in all three spatial directions there is inherent load-balancing and with our assumption of loose synchronization all processors will have identical times. Note that each of the p processors is only operating on local data during this phase. Therefore the first phase time is the same on the sequential and parallel machines.

Now (unfortunately) we must do the transposition to get the z-direction data onto the processors such that the 1-d FFT engine can be applied. This requires $\mathcal{O}(n^2)$ communication events[7] each of length n. Since the z-direction FFTs cannot started until this happens then all subsequent computation is pending on this *communication* intensive phase being completed[8]. We are lead to the time taken in this second phase as being

$$t^{(2)}(n,1) = n^2(t_l + t_w n) \tag{11}$$

In the third phase, doing the z-direction FFTs, we have all of the $p = n$ processors working concurrently on a (y,z) plane of data resulting in a time of

$$t^{(3)}(n,p) = t_c \left(\delta\, n^2 \log_2(n)\right) \tag{12}$$

Again this work is all done on data that is in the local memory of each processor. Each of the n sets of n FFTs is being done concurrently by each of the p processors so with the same caveats as for the first phase time all processors will take, and finish at, the same time

Combining all of the above into the equation for speed-up yields

[7] This operation requires some local gathers and scatters but we ignore that in this study.

[8] Actually in practice this is too restrictive: some of the computation could begin before all of the communication was completed, however this is beyond the scope of this paper.

3D FFT speed-up				
	S_u			
t_l	$t_w = 0.08$	$t_w = 0.04$	$t_w = 0.016$	$t_w = 0.008$
0.01	14.39	27.25	58.75	95.57
0.10	14.33	27.04	57.77	93.01
0.50	14.07	26.14	53.79	83.11
1.00	13.76	25.09	49.53	73.35
2.00	13.18	23.22	42.75	59.40
5.00	11.70	18.99	30.30	37.82
10.00	9.86	14.56	20.41	23.56
20.00	7.49	9.93	12.34	13.43
25.00	6.69	8.57	10.31	11.05
40.00	5.06	6.07	6.89	7.22
50.00	4.36	5.08	5.65	5.86
100.00	2.57	2.80	2.97	3.02

Table 10: The speed-ups for a 3D FFT with various values for the latency, t_l.

$$S_u = \frac{t_c\,(3\delta\, n^3 \log_2(n) + 2n^3)}{t_c\,(2\delta\, n^2 \log_2(n) + n^2) + n^2\,(t_l + t_w n) + t_c\,(\delta\, n^2 \log_2(n))} \tag{13}$$

In order to quantify this example we assume that $n = 256$. We further assume that the cycle time of the host processor, t_c, is 0.010 microseconds. We will consider several different bandwidths between processors; 100, 200, 500, and 1000 Mbytes/sec for a t_w of 0.08, 0.04, 0.016, and 0.008 microseconds, respectively. Let the latency, t_l, be a parameter since this is one of the key things we wish to explore.

Table 10 has results for this 3d FFT for a sequence of values of t_l and t_w. The immediate observation from this table is the strong effect on speed-up from the inter-processor latency (down the columns) and the weaker dependence on the bandwidth (across the columns). As the latency gets larger its effect tends to overwhelm any increased interconnection bandwidth. Of course the inverse observation also applies; when latencies are low then high bandwidth does have a noticeable effect on performance. However, in Table 10 the low latency numbers shown are equal to and very close to the cycle time of the processor. In practice latencies this low will not be achievable. We need to be mindful that we have *not* included interconnection contention effects with can be related to bandwidth. However, from the analysis doen here it is evident that if the inter-processor latency is high then an increase in the bandwidth is not a major help in increasing the speed-up. Reflecting back on the more abstract developments in the previous section we recall that the p^2 communication complexity was a detrimental factor in achieving performance in that abstract setting. It also proves to be problematic in this real application.

A facet of this problem that we have ignored which can further worsen the performance is the contention among the processors for communication channel bandwidth. With essentially all processors doing a one-to-many broadcast there will be significant contention of any existing topological interconnect. We purposely ignored this aspect in this paper; in practice it cannot be ignored.

7. Conclusion

It used to be that the astute user would look at the memory bandwidth on the vector machine to decide, to zero-th order, how fast the machine would run. On an MPP this is not sufficient but is, in principle, replaced by the inter-processor communication latency and bandwidth.

Owing to physical constraints in computer design it is certain that MPPs will never behave in the manner that the shared-memory multiprocessors do. The vehicle to attenuate the effects of inter-processor latency on an MPP is data locality. This is the approach that will be the determining factor in the amount of communication that is required by the application.

In this paper we have shown that the added burden of communication overhead gives a different perspective on the scalability of algorithms. This leads to the obvious conclusion that the first and foremost issue for the hardware designers is to produce low latency and high bandwidth interconnects. The algorithm designers task is to develop those algorithms that have low communication complexity.

Finally we note that a blind belief in Amdahl's law, either form, can be hazardous to your health! The original form is too pessimistic while the scaled form is too optimistic. Both forms ignore substantial issues such as communication and the growth of the sequential portion of the execution. We feel it essential that all complexity analyses of algorithms for their potential scalability incorporate at least a minimal communication complexity analysis. This we take to mean an inclusion of the communication time without regard to the topology induced contention. Of course if it is practical the topological effects should be considered but that is often more complicated than the simpler type of analysis presented here.

8. References

1. G. AMDAHL, *Validity of the single-processor approach to achieving large scale computing capabilities*, in Proceedings of the AFIPS Conference, Atlantic City, NJ, AFIPS Press, Reston, 1967, pp. 483–485.

2. G. ASTFALK, *Loop based parallelism and MPP*, in preparation, 1993.

3. R.S. BARR AND B.L. HICKMAN, *Reporting computational experiments with parallel algorithms: Issues, measures, and experts' opinions*, ORSA Journal on Computing, 5(1993), pp. 2-18.

4. Z. BOZKUS, S. RANKA, AND G.C. FOX, *Benchmarking the CM-5 multicomputer*, in Fourth Symposium on the Frontiers of Massively Parallel Computation, IEEE Computer Society Press, Los Alamitos, CA, 1992.

5. T.H. DUNIGAN, *Kendall Square multiprocessor: Early experiences and performance*, ORNL/TM-12065, Oak Ridge National Laboratory, Oak Ridge, TN, April, 1992.

6. I. FOSTER, W. GROPP, AND R. STEVENS, *The parallel scalability of the spectral transform method*, Preprint MCS-P215-0291, Argonne National Laboratory, Argonne, IL, 1991.

7. J.L. GUSTAFSON, *Reevaluating Amdahl's Law*, Communications of the ACM, 31(1988), pp. 532–533.

8. D.C. MARINESCU, *The speedup and efficiency of 3D FFT on distributed memory MIMD systems*, in Fourth Symposium on the Frontiers of Massively Parallel Computation, IEEE Computer Society Press, Los Alamitos, CA, 1992, pp. 150–154.

9. R. PONNUSAMY, A. CHOUDHARY, AND G. FOX, *Communication overhead on CM5: An experimental performance evaluation*, in Fourth Symposium on the Frontiers of Massively Parallel Computation, IEEE Computer Society Press, Los Alamitos, CA, 1992, pp. 108–115.

Towards The Implementation of the Linda Model into the Parallel Virtual Machine

V. De Florio, N. Galesi, F. P. Murgolo, V. Spinelli

SASIAM Tecnopolis Novus Ortus
str. prov. per Casamassima km. 3
I-70010 Valenzano (Ba)

September 5th, 1993

Abstract

As remarked by many, a parallel program is conceivable by means of a mixture of three computational methods i.e., message passing, distributed data structures, and live data structures. This paper describes the enhancement of PVM, the Parallel Virtual Machine, so that it will be able to support also the live data structure computational model. This is achieved by implementing Linda (a trademark of Scientific Computing Associates, New Haven) i.e., its Tuple Space and functions as a set of C/PVM codes. The programmer is free to mix Linda and PVM functions so that he gains the best from their features or he can see the system as a pure Linda implementation, knowing nothing about PVM technicalities.

Linda's way to express concurrency is quite elegant and easy, because the programmer does *not* have to worry about communication technicalities like establishing a message passing protocol between involved entities.

The preliminary version of the system is being implemented on a PVM 3.1 cluster made of a CONVEX C220 system and a set of workstations (IBM RS/6000 mod.320 and 530, HP 9000 mod. 750).

1 Introduction

A very important point writing a parallel code is the choice of the computational method i.e., the way parallelism is exploited. Carriero and Gelernter ([1], [2] etc.) remark that this choice is mainly based on three fundamental cases or on a mixture of them. Such cases are:

- **message passing**; in this case parallel entities communicate by sending explicitly messages to each other; this is the only computational method supported by PVM ([3, 4, 5]), Meiko's CS-TOOLS and many others.
- **distributed data structures**, in which a virtual shared memory space is simulated in a distributed environment. DSM is a product which supports this model, while Argonne's P4 [6] supports both message passing and distributed data structures.
- **live data structures**, also known as the Linda model, in which parallel entities communicate by means of an intermediate entity, that we call the tuple space manager ($\mathcal{TSM}$), which behaves like some sort of a database manager, recording and retrieving objects sent to it by the entities. Objects are addressed by their content—or in other words, $\mathcal{TSM}$ manages its database (the so-called *Tuple Space*, from now on referred as $\mathcal{TS}$) as an associative memory. Synchronization is given by success or failure in matching a tuple in $\mathcal{TS}$. Parallel tasks (in our terminology, *eval-servers* or $\mathcal{ES}$) awake, live their lives as Linda processes interacting with $\mathcal{TSM}$, and finally die producing a new object that is inserted in $\mathcal{TS}$.

We feel important to outline some aspects of the PVM/Linda implementation i.e., semantical compatibility and conceptual semplicity.
Indeed, any sequential language could be the host language for PVM/Linda primitives, the only modifications for the system being the PVM/Linda preprocessor. In other words, given any sequential language, we just need a different version of the preprocessor specialized for its grammar; PVM/Linda libreries and $\mathcal{TSM}$ are on the countrary independant of the host language.
For what concerns conceptual semplicity, Linda is a very simple way to express parallelism into user's code, masking him all communication, synchronization and scheduling technicalities. Of course there's a price to pay for such semplicity: the user can't directly control the whole parallel processing as with lower level tools. To overcome this problem we developed the Linda model on top of PVM, giving the user the choice between pure Linda or a mixture of Linda and PVM calls for his/her code.

Some important aspects about PVM/Linda which is is useful to remark are:

- *conceptual gain for expressing an algorithm*:

  The availability of a conceptual tool more suitable to a certain problem often means a better and faster solution. Some parallel algorithm are better conceivable with a message passing scheme, some others fit very well with a live data structure approach; and of course there exist cases in which a mixed model results in the best mapping in terms of ease of use, short-time development, and efficiency.

- *portability*: this software ineherits the high degree of portability that characterizes PVM; a socket-based version of Linda, as an example, could produce non-portability.

- *ease of use*: an important property for any tool is fast training times and ease of use. As compared with message passing techniques, requiring the programmer to:

  - understand the model
  - manage job creation and destruction
  - set up a communication protocol between jobs
  - deal with low-level technicalities like synchronization, scheduling, topology of processors ...

  PVM/Linda requires the sole model understanding. Of course, technicalities are often necessary for optimizing reasons; but the user must be free to chose according to his/her know-how of the system.

  Communication can also occur implicitly through routine calls in the `eval()`s. This is an even more intuitive way to perform communication among two tasks.

- *availability as an open system* the implementation could be modified by anyone so to customize the system to his needs;

- *multiple inner logics*: The same Linda program could result in very different actions according to the particular implemented algorithm; such actions will in turns result in different mappings of the user code on different architectures just changing the $\mathcal{TSM}$ module. In this sense, given the existence of a family of $\mathcal{TSM}$'s, the user can better map its code to its hardware architecture. Note that this is completely independant of the user's codes.

- *load balancing*; in some sense, Linda's processes are some sort of producer-consumer concurrent agents. If properly managed, a Linda program could be structured with a producer which fills $\mathcal{TS}$ with work-slices e.g., subintervals in a numerical integration process, and a family of consumers with different working times due to the different performances characterizing a heterogenous computing environment. In this situation the consumers asynchronously access $\mathcal{TS}$ "eating" work-slices at different rates, proportionally to their speed and stopping when $\mathcal{TS}$ is empty. Of course, a little work-slice both means a reduction of worst case time and an increase in communication. The final chapter, §4, depicts this situation with an example.

In the following section we briefly introduce the PVM/Linda model.

2 The PVM/Linda Environment

2.1 The Linda Model

Linda is implemented by means of six primitives. Using Linda it is possible to enhance a traditional sequential programming language allowing it to express concurrency in quite a straightforward and expressive way.

As well known, Linda primitives operate on *tuples* i.e., lists of objects identified by their cardinality and the type of their components. No limits are imposed by the model on the number of elements in a tuple.

A Linda process starts as a normal sequential process. Other than conventional actions, a Linda process is allowed to perform the following activities:

- it can put a tuple in the public space ($\mathcal{TS}$) by means of `out()`. As an example, the following command:

  ```
  out("DomainDecomposition", a[i], b[i])
  ```

  results in a new tuple in $\mathcal{TS}$. The new tuple is characterized by these properties:

  - cardinality : 3;
  - type sequence (in order): string, real, real;
  - content (in order): `"DomainDecomposition"`, `a[i]` and `b[i]`.

- it can ask for a tuple in the tuple space. $\mathcal{TS}$ is managed as an associative memory, so selection is operated by giving a template tuple. This action can be performed in a variety of ways:

- `rd()`, which blocks the caller up to the availability of a matching tuple. That tuple is *not* removed from $\mathcal{TS}$.
- `in()`, which similarly blocks the caller up to the availability of a matching tuple, finally removing that tuple from $\mathcal{TS}$.
- `rdp()`, which returns immediately with a boolean value meaning the presence/the absence of a matching tuple in $\mathcal{TS}$; if such a tuple exists, its value is copied from $\mathcal{TS}$ but *not* removed from it.
- `inp()`, which acts in a quite similar way as compared to `rdp()`, the only difference being the removal of the eventually matching tuple.

- A Linda process can also execute the `eval()` function, which at first sight resembles very much the `out()` function. Indeed, `eval()` behaves exactly like `out()`. Let's try to depict the differences by means of an example. Both the routines can handle the same objects e.g.,

```
out(a, b, f(a), g(b));        eval(a, b, f(a), g(b));
```

  And they both end issuing the referred new tuple in $\mathcal{TS}$. Where's the difference? It's quite simple, though here lies Linda's power: while in `out()` `f(a)` and `g(b)` are computed on the same machine in which the main Linda process is also run, the `eval()` spawns a different process (eventually on a different machine) whose task is to compute the values of the referred functions. So `eval()` is Linda's way to express concurrency.

Linda is often said to operate on "Live Data Structure". The reason for that lies in `eval()`: if you consider a C language function returning, say, an integer value, you are speaking of a chunk of machine language instructions that eventually will run and, given an input of some sort, produce the corresponding integer value. You can use such function in any place where an integer value is admitted by the C language grammar; in some sense, you are free to refer to the datum that the function will provide before this datum is actually ready to be picked up. So a live data structure is a running function returning some object. Within Linda you can say:

```
eval(a, b, f(x));
```

which means a request to the $\mathcal{TSM}$ so that it records a triple whose contents are `a`, `b` and the value returned by `f(x)`. Of course this means that `f(x)` must be computed somewhere by some PVM component running on a PVM computational element. $\mathcal{TSM}$ will satisfies this request only after the end of the computation of `f(x)`, so there's an automatical synchronization of all other tasks waiting for that tuple: live data structures are running objects which give raise to static, public data when they die; till that moment, they are running, private data.

Note also that functions referred in the `eval()` can in turn execute PVM/Linda calls.

In the following section we will briefly discuss the PVM/Linda system and its implementation.

2.2 The PVM/Linda Architecture

From the user's point of view, a PVM/Linda application consists of a user code containing Linda calls and a library called `libLinda.a`. Our implementation adds to this set a primitive of initialization and a primitive of termination. The user compiles his/her program with the command `sysprep`, a preprocessor whose aims (quite briefly) are:

- to insert proper special code for initializing the parallel environment;
- to convert all references to Linda calls into standard-C function calls. In the translation process every expression supplied as an argument of a Linda call is converted in a quadruple consisting of:
  - the expression itself,
  - its type (a integer number greater than zero),
  - its status (i.e., wether the expression is constant or is a variable),
  - and the dimension of the referred object.

  Every function referred in the `eval()` are identified by an internal integer code that we will call an *icode*.
- to create a file containing all functions that are referred in the `eval()`s, plus the proper code to receive data argument across a PVM connection, compute the specified functions and issue an `out()` with static data plus values returned by the functions. Such code is called *the eval server* (that we will call $\mathcal{ES}$ from now on.)

This module will be discussed in §3.2.1.

Next steps are to distribute copies of the eval server among the hosts configured as Linda computational elements, and to start the distributed compilation process of these copies. As already said, this process builds up a special protocol with a client executing an `eval()`. The client sends static data and function identifiers and then is free to continue; the server executes the associated functions, emits an `out()` with resulting data, and finally dies.
All such processes are PVM components.

3 PVM/LINDA at Application Level

The PVM/Linda system can be seen as a three-layered framework:

- at the lowest level, it's simply C language;
- at the middle level there is the PVM 3.1 library;
- the highest level is `liblinda.a`, the PVM/Linda library.

PVM/Linda allows the user to properly mix PVM's message passing primitives with LINDA's primitives. This section covers the application level interface to the PVM/Linda system.

3.1 The PVM/Linda System

The user can starts up the system in two ways.

In the first way the user calls his/her main procedure `linda_main()`. This routines is in turn called by a system supplied `main()` which also initializes the system and finally terminates it. In this way the user is always in PVM/Linda mode i.e., from the beginning there's a $\mathcal{TSM}$ component somewhere in the system ready to manage all $\mathcal{TS}$ primitives.

In the second way, when the user decides to start a PVM/Linda section of his/her code, he/she calls the function

```
linda_init();
```

This function results in spawning a $\mathcal{TSM}$ on a particular host; the choice of this host relies on PVM and on the way the user arranged his/her hosts in PVM's configuration file. PVM/Linda calls will be now managed up to an explicit call to the routine

```
linda_end();
```

In other words, PVM/Linda statements are corecctly managed only if between these bracket-statements.

If the machine on which to run $\mathcal{TSM}$ is not in the PVM cluster, the PVM/Linda system tries to include it.

When the user is in PVM/Linda mode she/he can use any of the Linda functions (`in()`, `inp()`, `rd()`, `rdp()`, `out()`, and `eval()`) and any of PVM primitives. This is true also for functions referred in the `eval()`'s i.e., the PVM process associated with every single `eval()` can dialogue with the user master program and with the other eval-servers via the $\mathcal{TSM}$ and/or PVM.

3.2 Overview of the System

3.2.1 The PVM/Linda Preprocessor

The PVM/Linda preprocessor is a program which converts a source code written in the C language extended with the special PVM/Linda syntax into a set of standard C codes which consists of

1. a literal translation of the user program in which all PVM/Linda primitives and conventions are converted according to the C language syntax and to our program's needs;
2. a special code whose aims is to compute the functions referred in the *eval()*'s;
3. some special code for managing and recording the types of variables.

For what concerns item **1.**, the translated code converts each reference to a C language expression or function call within a PVM/Linda statement into a list of informations according to the following cases: a variable, a function call, and an expression.

The code produced in 2 is a special PVM component which will be copied on all the computational elements that will take part in the execution of the PVM/Linda process; it plays the role of the server side of the `eval()` function and so it is called the `eval-server` or $\mathcal{ES}$ for brevity. In other words by this code we issue for every `eval()` a PVM process whose task is to execute all functions referred in all `eval()`s, building a new tuple, in which the function call is substituted with the return values of the function and issuing an `out()` of the new tuple.

As a consequence of these needs the preprocessor performs a first analysis phase of the user source file and collect the above mentioned information. Through such information the preprocessor produces a set of source files. In details the preprocessor' actions are:

- analisys of the sources and creation of a new PVM/C source file from the original user source file; such file contains the translation of the Linda primitives into proper calls to the functions in our PVM/Linda client library, `liblinda.a`.
- creation of $\mathcal{ES}$;
- creation of a file describing types; such file will be used during the compilation of $\mathcal{ES}$.
- creation of a file with ancillary routines.

Every linda primitive is translated into a call to a function contained in the library `liblinda.a`. Such function interfaces $\mathcal{TSM}$. This function has a variable number of parameters, the first of which is an opcode which specifies the particular linda function; next it follows a list of informations about single elements of the tuple. As already stated, an element of a tuple can be:

- a *variable* ;
- a *function call*, with or without parameters;
- an *expression*.

As already said, the PVM/Linda preproccessor is made of three conceptual different parts:

- a syntactical C analyzer;
- a PVM/Linda statement recognizer;
- the $\mathcal{ES}$ creator;

The tasks of syntactical analyzer are to parse the declarative part of source file saving informations about variables, types and function prototypes in a binary tree simble table and to scan the C source file recognizing only C-expressions that are elements of a PVM/Linda tuple. The program has been written exploiting **lex** ([9]) and **yacc** ([10]) potentialities.

3.2.2 The Management of the Tuple Space

The $\mathcal{TSM}$ (Tuple Space Manager) is the program which process Linda requests coming from user programs. It is made up of the following four files :

a) `TASK_TSM.c` - contains the main program and high level routines for running the requests.

b) `TSM_ROUTINES.c` - contains all low level routines for runnig $\mathcal{TS}$ and $\mathcal{JS}$ (the Job Space). These two structures are used for storing sent tuples ($\mathcal{TS}$) and waiting requests ($\mathcal{JS}$).

c) `TASK_TRASM.c` - contains all communication routines (using PVM calls) used both by $\mathcal{TSM}$ and by Linda routines (contained in `liblinda.a`).

d) `TASK_PRINT.c` - contains error printing routines and other routines for printing service information when user defines debug flags.

The $\mathcal{TSM}$ essentially uses two very similar date structures, namely $\mathcal{TS}$ and $\mathcal{JS}$, which are characterized as dynamical structures, so their dimensions are proportional to the work load. $\mathcal{TS}$ is a vector of pointers to distinct tuple groups, each of which is characterized of having all (and only) the tuples in $\mathcal{TS}$ with the same number of elements. Every request to $\mathcal{TSM}$ is sent to a particular tuple group and this makes search faster. These tuple groups are implemented as dinamical lists and this, even if could involve an onerous management (for scanning the list), does not set bounds to the dimension of the list itself but the limits of the computer. It is possible to realize three basic operation on $\mathcal{TS}$:

a) *TUPLE INSERTION* - the current tuple is put ahead of the tuple list with the same length.

b) *TUPLE SEARCH AND DELETION* - the current tuple is searched in a particular tuple group and if the search successes, the tuple is read and destroyed from $\mathcal{TS}$.

c) *TUPLE SEARCH WITHOUT DELETION* - likewise b) but tuple is not destroyed.

Each of these operations are realized by particular routines in `TSM_ROUTINES.c`. $\mathcal{JS}$ is a structure quite similar to $\mathcal{TS}$, used when there are requests which can not be executed (in particular requests as `in()`s or `rd()`). Also in $\mathcal{JS}$ the requests are stored in separated groups according to the tuple group they refer to. We have two possible operations on $\mathcal{JS}$:

a) *INSERT A REQUEST* - when a request is waiting for a tuple which can not be found in $\mathcal{TS}$ then it is inserted in $\mathcal{JS}$.

b) *DRAW OUT A REQUEST* - a sleeping request is extracted from $\mathcal{JS}$ and $\mathcal{TSM}$ tries to satisfy it.

In `TASK_TRASM.c` we have all the routines for the communication between $\mathcal{TSM}$ and Linda library. The communication between these two entities is not symmetrical, because $\mathcal{TSM}$'s answer is much more concise than the associated Linda request, so we have two distinct phases:

a) *REQUEST FOR AN OPERATION TO* $\mathcal{TSM}$ - when `liblinda.a` sends a request, $\mathcal{TSM}$ have to record not only the tuple but also all the necessary informations so to send the answer only to the waiting process and not to the other ones. For this reason, $\mathcal{TSM}$ stores user name, user istance and so on.

b) *ANSWER* of the $\mathcal{TSM}$ - when $\mathcal{TSM}$ has to answer, at first it sends a flag that indicates if there were errors or not (printed by $\mathcal{TSM}$ on standard error) during the computation. If there was an error, this is the only information

sent by $\mathcal{TSM}$ otherwise $\mathcal{TSM}$ sends result tuple. It is necessary to notice that for predicative operations (a `inp()` or a `rdp()`) if there is a negative answer, $\mathcal{TSM}$ sends only this signal,while for operation such as `out()` or `eval()` it does not send any tuple. For `in()` or `rd()` requests, $\mathcal{TSM}$ sends back only the non constant elements of the tuple.

Because $\mathcal{TSM}$ manages requests coming from more than one user, it is obvious that $\mathcal{TSM}$ algorithm must contain some mechanisms to discipline requests' arrival, to check request' correctness and so on. Briefly, the logical scheme of the $\mathcal{TSM}$'s algorithm could be described in the following way:

- initialization phase ($\mathcal{TS}$ and $\mathcal{JS}$ allocation),
- the $\mathcal{TSM}$ waits for user's requests and, if correct, it executes the relative actions on $\mathcal{TS}$ and/or $\mathcal{JS}$; then it returns waiting for another user's request.
- if $\mathcal{TSM}$ receives a user `END` request, it returns unresolved requests to the waiting programs with a proper error code; anyway, it frees $\mathcal{TS}$, $\mathcal{JS}$ and stops.

The $\mathcal{TSM}$ system activation involves $\mathcal{TS}$ and $\mathcal{JS}$ allocation and error variable initialization. If $\mathcal{TSM}$ does not execute enroll operation rightly, it prints an error message and exits. $\mathcal{TSM}$ waits for a request by mean of the `RECEIVE_JOB_DESCRIPTOR` routine (in `TASK_TRASM.c`) whose results are the constant values `YES` or `NO`; if there was a right reception (the `YES` answer) the routine returns also a significative job descriptor (described in 1.2) too. Now $\mathcal{TSM}$ checks if the request is correct from a syntactical point of view by mean of `IS_A_RIGHT_JOB` routine (in `TSM_ROUTINES.c`) and if so it processes the request with the `ACTIVATE_PROCESS` routine. At the end of the elaboration request it sends user program the result by `SEND_RESULT` routine (in `TASK_TRASM.c`). Every time $\mathcal{TSM}$ detects some errors, it increments error variable and, when variable gets to a limit value (defined by `MAX_ERROR_NUMBER`), it activates an error process which rejects every following request to $\mathcal{TSM}$. In every case $\mathcal{TSM}$ activity is halted when a user process sends the `END` request.

As already said the preprocessor captures Linda primitives in user programs and changes them to Linda library routines calls. This library transforms user requests into the data structures described in 1.2 and sends them to $\mathcal{TSM}$. Only if there were not errors, significative values are returned. In any case Linda library (before formatting the request) checks if $\mathcal{TSM}$ is already active, and if not it prints an error and stops doing anything. A request is parsed before transmission so to check if there were errors.

Request correctness controls are made more than one time because date structures are transfered from one process to another one by means of PVM processes and there could be accidental modifications. Every Linda function has has three possible return values (`YES,NO,ERROR`). The user process is

always stopped to wait $\mathcal{TSM}$ answer and if user process is killed during this wait, $\mathcal{TSM}$ reports this condition.

4 PVM/Linda Benchmarks

The PVM/Linda package has some utilities for testing its perfomances on the particular user pvm cluster. In particular we would like to outline the `time_test` utility, which realizes a time performances monitor both by showing the results on video (run time) and by memorizing them in a user file. The hereby reported performances are related to a software for the numerical integration of a generic function $f(x)$ on the compact range $[a, b]$, by using the first six Newton-Cotes formulae.

The obtained results relate to a PVM cluster configured as follows: a CONVEX C220, an IBM RS6000/530, and a Hewlett & Packard HP9000/750; such machines are connected in a TCP/IP - Ethernet network. All experiments were performed with the sole CONVEX C220 running the $\mathcal{TSM}$.

The following values were considered for the benchmarks:

- integrating function: $f(x) = \cos(x)e^{\sin(x)}$
- integrating range: [0, 3];
- number of integrating subrange: 100;

while we decided to vary the following parameters:

- number of eval-server;
- number of PVM/Linda hosts;
- number of points in each integrating subrange.

The following graphics display some results obtained, noting that these are wall-clock times taken in idle periods. X-values are the number of $\mathcal{ES}$ considered, while y-values represent the time spent in the particular execution. Note that slopes of all segments constituting the piecewise curves reflects the power of each machine. Note also that the particular algorithm exploits a peculiar feature of PVM/Linda i.e., the fact that the workers in our application asynchronously access the pool of work units according to their actual speed so realizing a sort of dynamical balance of the workload in the PVM cluster.

Other tests have been performed through the utility `time_test`. Such tests measure the performance for each PVM/Linda functions. Note that these results heavily depend on network usage and workload of the overall PVM cluster. In general in our configuration we obtained the following values for all PVM/Linda primitives except `eval()`:

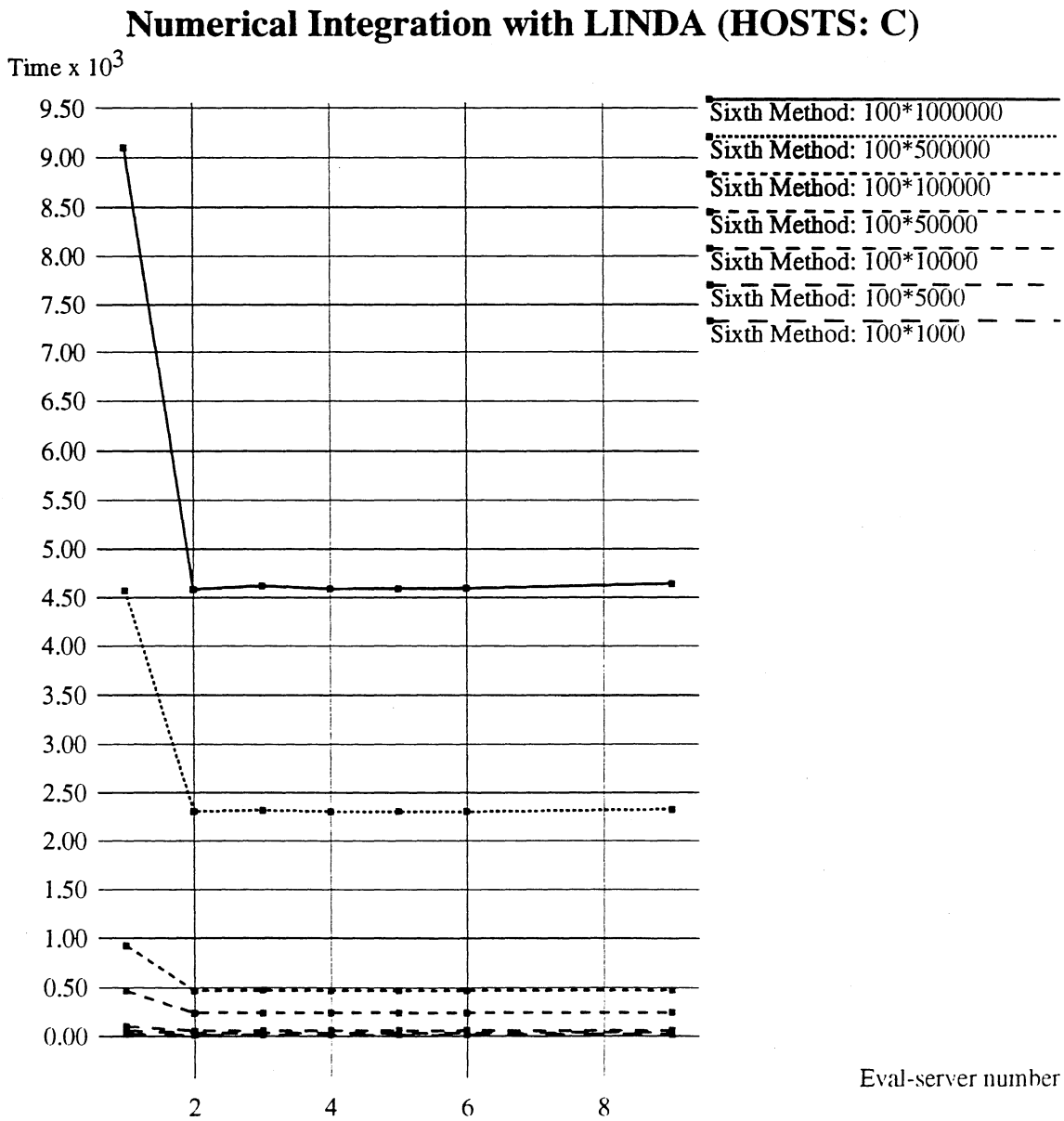


Figure 1: Increasing the workload and the number of `eval-servers`. One computational element (CONVEX C220). As you can see, the highest gain occurs when we use two eval-servers because of the number of cpus of this machine. Note also that angular coefficients for the first segments for each curve converge to the limit value -1 as the workload increases.

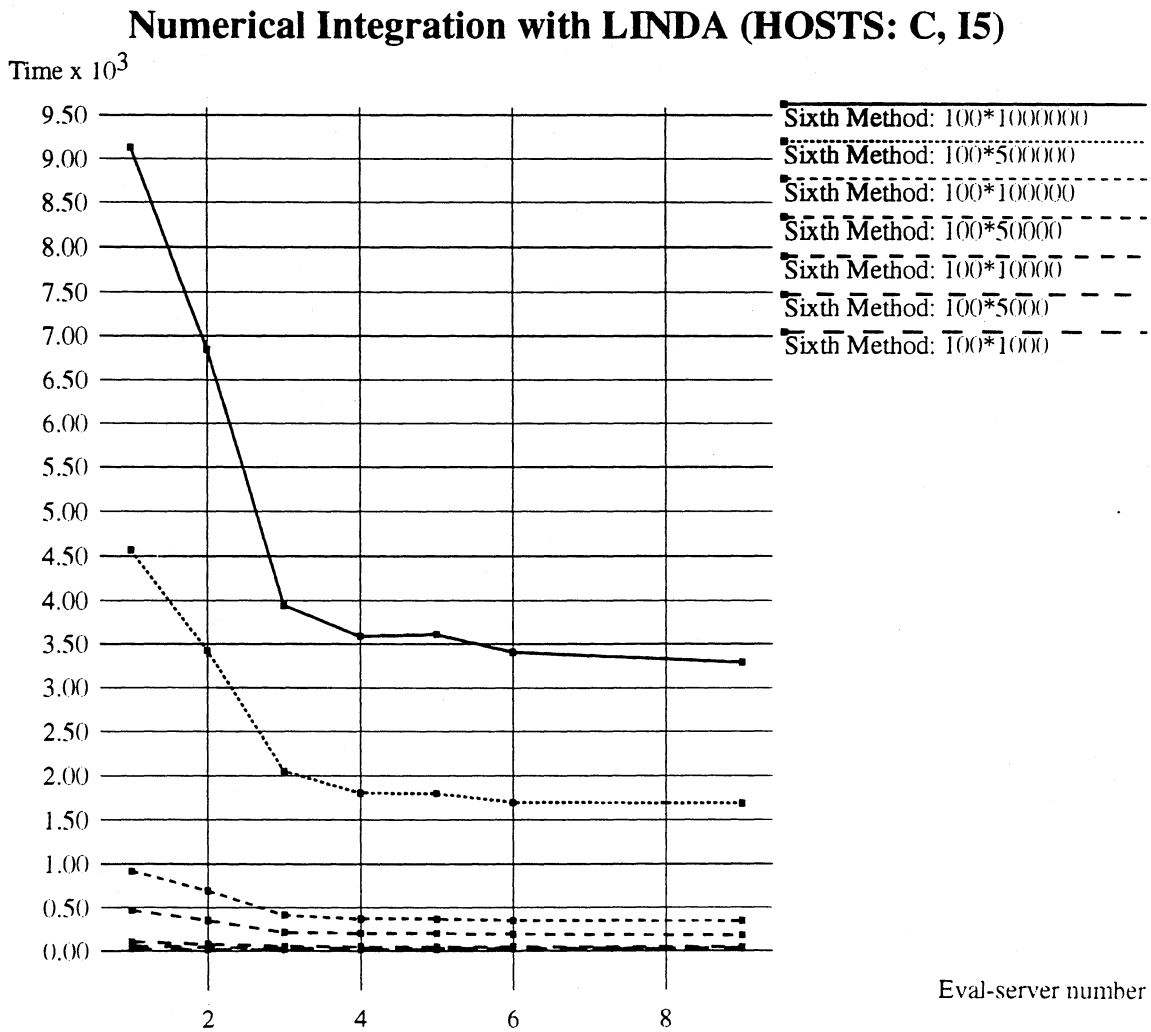


Figure 2: Increasing the workload, the number of `eval-servers`. Two computational elements (a CONVEX C220 plus an IBM RS6000/530 host). As you can see in this case the main gain was obtained using up to four $\mathcal{ES}$s. Subsequent gains are more and more influenced by process overcrowding. For $x = 1$ the only contribution is due to the Convex machine. For $x = 2$ the IBM machine starts contributing, but the highest gain occurs for $x = 3$, when the second cpu of the C220 is re-involved in the computation.

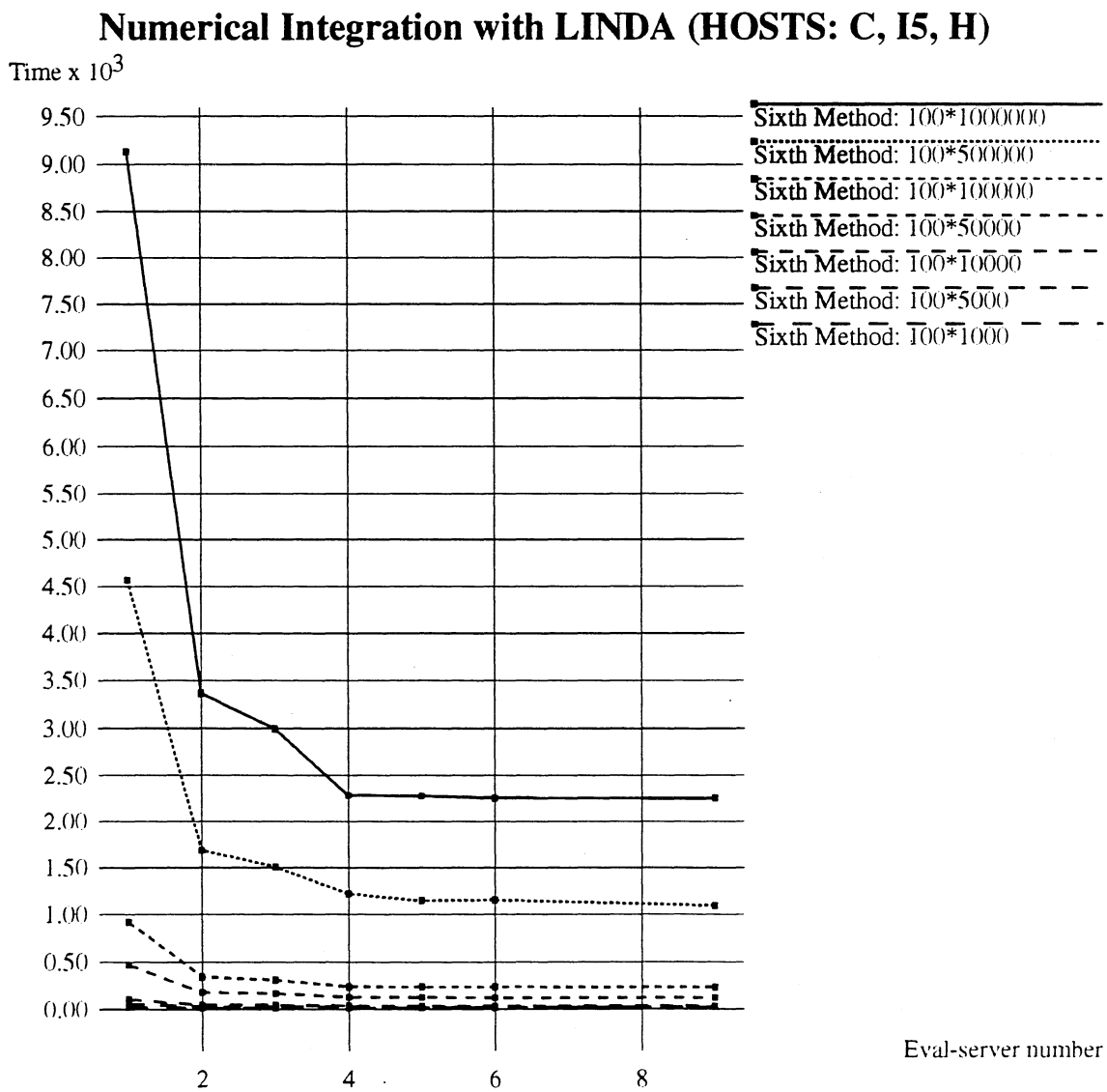


Figure 3: Increasing the workload, the number of `eval-servers`. Three computational elements (C=CONVEX C220, I5=IBM RS6000/530, and H=HP9000/750). Also in this case, for $x = 1$, we exploit the sole Convex machine. The highest gain is for $x = 2$, due to H, the most powerful machine in our cluster. For $2 \leq x \leq 4$ we continue to improve performances but more and more slowly.

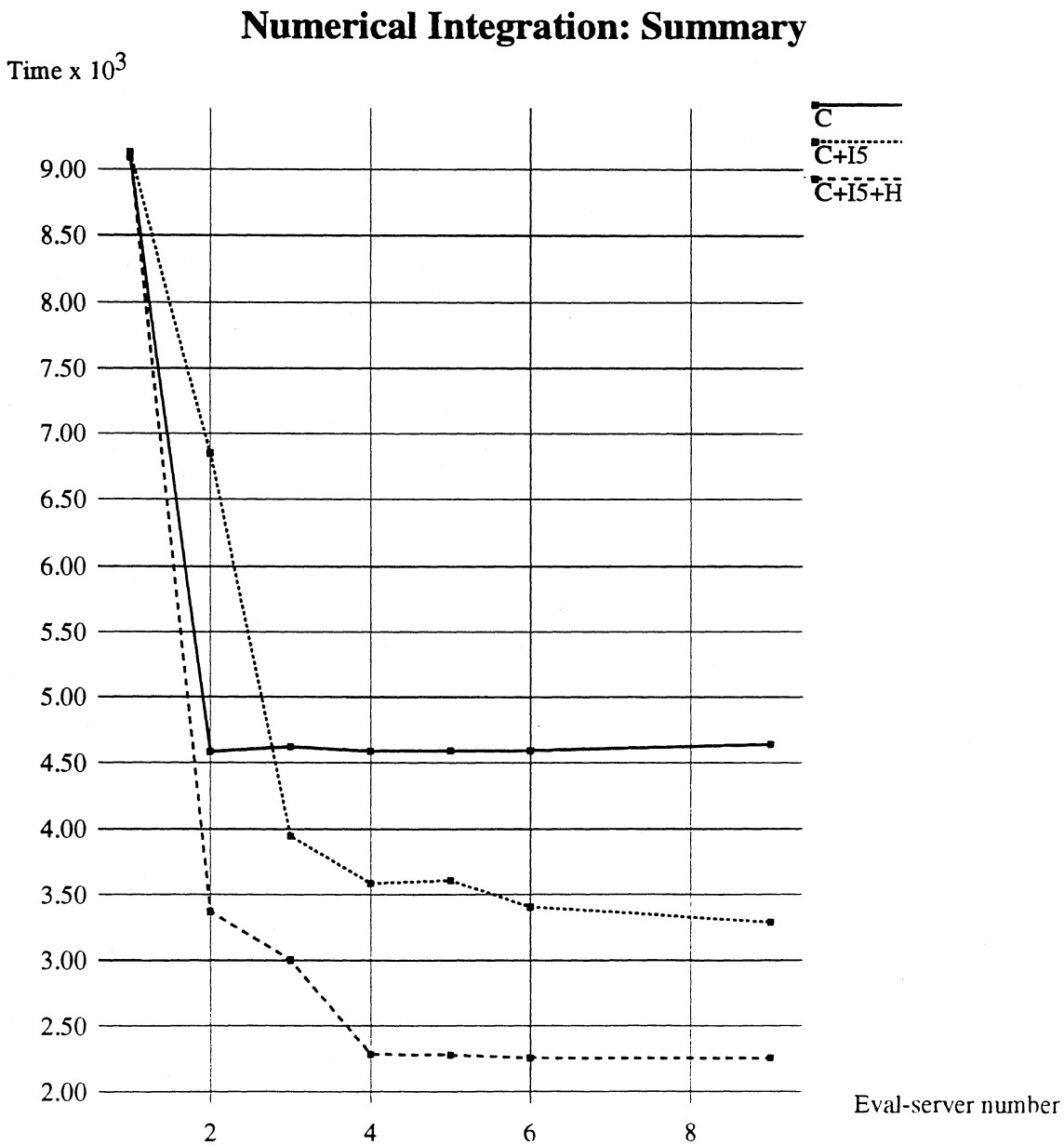


Figure 4: Comparing the 1000000 point execution for the three previously shown graphics. In each case the contribution of each machine is evident even though computational power is not uniformely distributed into the PVM/Linda cluster.

	min	mean
local	30/40 msecs	50/60 msecs
distributed	80/90 msecs	90/110 msecs

In the table, columns represent minimum and mean observed times while rows represent the two situations we can have during the elaboration:

- **a totally local computation**, which means that `time_test` and $\mathcal{TSM}$ run on the same machine
- **a distributed computation**, which means that `time_test` and $\mathcal{TSM}$ run on different machines; in this case the performances are heavily influenced by the slower machine. In general our experience is that $\mathcal{TSM}$ have not to run on the slower one

Observing this table we can do some consideration:

- we do not mention the max values obtained, because they are meaningless for their random varability.
- in general both min and mean value had the same values if we consider the local case, while for the network case the same values have just the same behaviour as the max values.
- we can consider 30 msecs as a physical limit for our machines, because we never reached smaller times.
- the real performances for each of the PVM/Linda primitives could be very different from the ones you could get with the utility `time_test`, but you must consider that `time_test` realizes these benchmarks *by using $\mathcal{TSM}$ in a totally dedicated way.*

5 Conclusions

The package developed so far consists of a series of directories which resemble PVM 3 organization—and actually have been designed so to find their place within the PVM file system. There, a number of testing and benchmarking codes have been inserted consisting of the master code, its translation, the code for the $\mathcal{ES}$ and a makefile. In the release 1.0 of the product there will be also a directory for the preprocessor.

The previously shown tests can be of course developed through message passing using e.g., the PVM library. A potential problem of an explicit use of message passing is that the workload distribution in a heterogeneous computing environment is up to the master process; in PVM/Linda, as remarked at page 12, eval servers asynchronously access a pool of work-units (already known as integrating subranges) in $\mathcal{TS}$ so they feed themselves according to

their actual speed. In such a way the slower ones do not affect the overall performances as it would be in a a-priori homogeneous distribution.

References

[1] N. Carriero and D. Gelernter. How to write parallel programs:a guide to the perplexed. *ACM Computing Surveys*, 21:323–357, September 1989.

[2] N. Carriero and D. Gelernter. Linda in context. *Communications of the ACM*, 32, Number 4:444–558, April 1989.

[3] G. A. Geist and V. S. Sunderam. Network based concurrent computing on the pvm system. Technical report, Oak Ridge National Laboratory.

[4] V. S. Sunderam. Pvm: A framework for parallel distributed computing. Technical report, Emory University.

[5] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. Pvm 3 user's guide and reference manual. Technical Report ORNL/TM-12187, Oak Ridge National Laboratory, May 1993.

[6] Lusk E. Butler R. User's guide to the `p4` parallel programming system. Technical report, August 1992.

[7] Skapura D. M. Freeman J. A. Freeman. *Neural Networks — Algorithms, Applications and Programming Techniques*. Addison Wesley, 1991.

[8] Schoinas G. Issues on the implementation of programming system for distributed applications. Technical report, University of Crete (dept. of Computer Science).

[9] M. E. Lesk. Lex – a lexical analyzer generator. *Comp. Sci. Tech. Rep.*, 39, 1975.

[10] S. C. Johnson. Yacc: Yet another compiler compiler. *Software — Practice and Experience*, 5:395–496, 1975.

A LIBRARY TO CREATE A DISTRIBUTED ENVIRONMENT THROUGH A PVM BASED COLLECTION OF REMOTE FUNCTION SERVERS

F. P. Murgolo, V. De Florio, P. Li,
G. Musaio Somma, U. Tolve

SASIAM/Tecnopolis CSATA Novus Ortus
- 70010 Valenzano (BA) Italy -
tel.: +39 80 8770265 fax: +39 80 6951868
e-mail: root@fourier.rob.csata.it

Abstract

A need whose importance is increasing more and more in research laboratories is the capability of distributing computation among a set of heterogeneous hosts, each of whom is specialised in a particular kind of operations so that the user can gain the best mapping between its problem and the computer architecture. From this derives the search for a good match between the needs arising from user's problem and hardware (processing power, available I/O channels, ...) and software (availability of special libraries) functionalities. RHL tends to ease this search by offering a common concise interface, which can be used easily, between a user program (whose host is called client) and the libraries available in different hosts of the net (called servers). The restricted number of functions and their simplicity makes this small library an easy means to develop distributed applications and it could be of particular interest for laboratories using specialised computer systems and willing to share their software functions.
It is to be remarked that using this product is possible to spawn a set of parallel independent actions among the different heterogeneous hosts in the PVM cluster, exploiting therefore the wider set of resources available in the Network Computer. Moreover, the user must not be aware of technicalities concerning parallelism and communication.

Introduction

RHL, whose name stands for "Remote Host Library", is a tool for distributing processes across a TCP/IP network of UNIX heterogeneous workstations. All communications are based on PVM, the Parallel Virtual Machine mechanism developed at Oak Ridge National Laboratory.
RHL gives to the user the basic services offered by PVM but with a higher level interface which masks all technicalities concerning the set-up of communications, management of data transfers, and cleaning up procedures.
Let's imagine we are writing a rather complex application that for its intrinsic nature would certainly benefit of the aggregate power of many different computers. This is often a real need because large and complex scientific applications often need to use different hosts at the same time, in order to exploit either the overall computing performances (resulting from the concurrent use of them), or peculiar capabilities of each in computing particular classes of data. Besides the capability of performing

parallel computations on this network connected environment, it would be highly desirable to have the possibility to carry out the computations of particular classes of data on the machine best suited for that purpose.
RHL allows the real integration, at application level, of libraries written and optimised purposely for each architecture, and the execution of particular computations on the architecture which appears to be the most versatile in performing that task, or the only one on which is available the library.
RHL performs all such actions using the features of PVM permitting a network of heterogeneous Unix-based computers to be used as a single large parallel machine; the most complex problems regarding communication among different architectures have been successfully solved, allowing programmers to quickly and efficiently exchange data among concurrent processes.
Many packages, including message passing systems (Meiko's CS-TOOLS, Argonne's P4, GMD's PARMACS) and non-message-passing products (SCA's LINDA system), could be used as lower layer for RHL; given the high degree of portability and spreading of PVM we decided to adopt it as the basis for our work.
Until now not enough efforts have been done to facilitate the still heavy work of application developers which have to concentrate themselves not only on their algorithm but also on the many problems of communication, synchronization, scheduling that underlie the creation of a distributed application. A usual situation in this case is with the programmer forced to set up a real communication protocol among the different entities in which his/her application has been decomposed. This is the rationale of our product, RHL: we feel that it is of the highest importance to create a tool which enables the user to easily develop a distributed application.
RHL allows:

1. application developers to design their codes in an easy way, without any care of communication features, concentrating their efforts on which data have to be exchanged and which host have to be used rather than on how to exchange data among hosts;
2. the overall use in one distributed application of specific libraries of code installed on various hosts (Servers), by one (or more) processes running on Client machines;
3. users working on a client host to execute functions residing in a server host library, collecting later results on the client machine.

All these operations are executed transparently of the user, which can be completely unaware of the communication among processes and of its management.

The RHL tool

RHL, as previously stated, is a PVM based software which offers the user a simple interface to set up a distributed application (see fig.1).
Once the tool has been properly configured, a user can write a code in which referred functions are not restricted on the some host in which he/she is working, but on the contrary they can be on any PVM-configured host, both local and remote. RHL manages all needed communications for transferring data back and forth from a client program to a set of server programs.

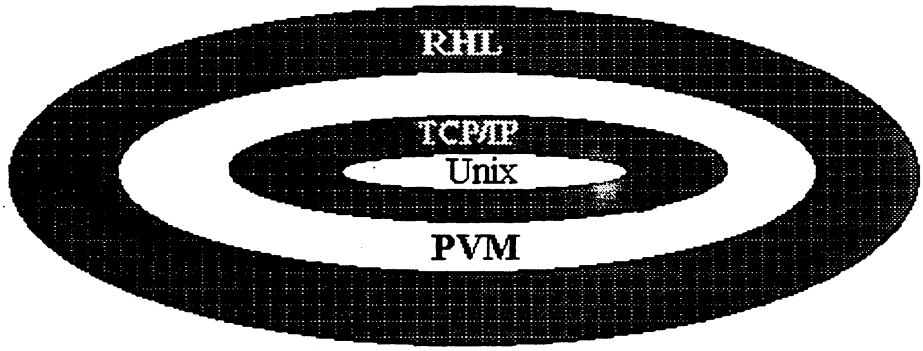


fig. 1

Architectural Choices

Starting from the consideration that a function call statement is specified simply referring the function name and its arguments, RHL has been developed trying, as much as possible, to approximate this interface, reaching transparency and ease of use.
Let's try to draw the operations needed to reach this state of things. If we need to call in such a way a function from a set of libraries residing on a number of remote hosts we have to:

1. dispose of a special code, which can be run on the remote hosts (so we need to create that code, and we need a way to spawn it and destroy it at will);
2. dispose of a communication channel between our code and the one on the remote hosts (some mixture of hardware and software links, see fig.2);
3. uniquely identify the particular function to be executed;
4. specify data to be transferred and direction of communication (if we need to transfer data from our code to the remote one or vice versa).

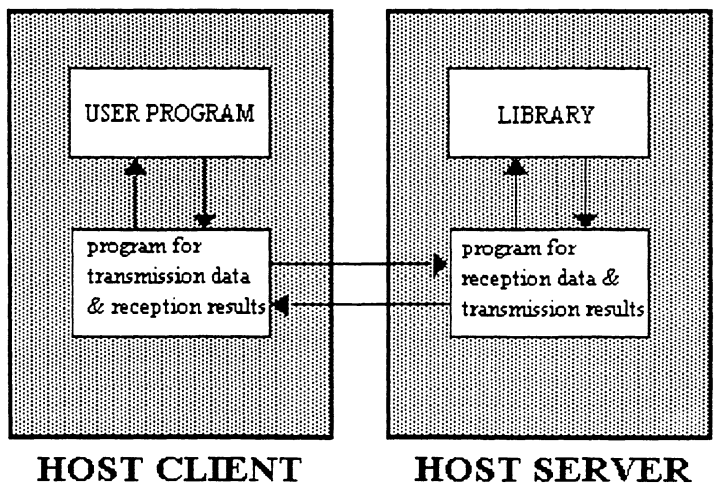


fig. 2

These four needs reach fulfilment in RHL: indeed,

1. *RHL* implements the special code, ***rhl_se*** , which is the server side whose task is to execute the remote functions which must be specified in a configuration phase;
2. we consider as *pre-existent* the hardware/software layers needed for this;
3. *the installer,* in the configuration phase, uniquely identifies the functions on both client and server;
4. *the user* which finally wants to recall the remote function, specifies through a format string data, data types, and direction of data flow (from the client to the server, vice-versa, or both).

Note that on a single environment they can exist different configurations of RHL, each one tailored for a certain class of users and managed by at least one person (the installer). Such person is the one who executes previously mentioned action 3., so the minimum know-how level for the user to start operating is considerably shortened (action 4.)

The RHL Syntax

In order to distribute computation task among a set of hosts, at least four actions are needed: activation of the application code on the configured servers, data dispatching and results collection, deactivation of the distributed application. To make this possible we developed the four functions which constitute RHL. Before introducing these functions we want to remark that, during syntax design, we tried to adopt a syntax quite similar to that of other standard functions; moreover we heavily focused on the masking of communication technicalities (protocols, synchronization etc.) and intrinsic limits of lower layers (PVM limit on the maximum packet length etc.).

The RHL primitives are:
rhl_init, whose name stands for **R**emote **H**ost **L**ibrary **Init**iation, initializes the system. Its syntax is:

int rhl_init (int *num_hosts, char *host_list[], int hid[]);

where:
num_hosts is the total number of hosts among which we want to distribute our computation.
host_list is a vector with the TCP/IP symbolic names which identify the hosts in our application.
hid is an output vector whose entries are filled with negative values which are PVM error return codes if something went wrong; a positive value for hid[i] means that the i-th host was successfully initialized. Note that a failure of a host does *not* mean a failure of the application: simply, that host does not contribute to the computation (its tasks are not computed).

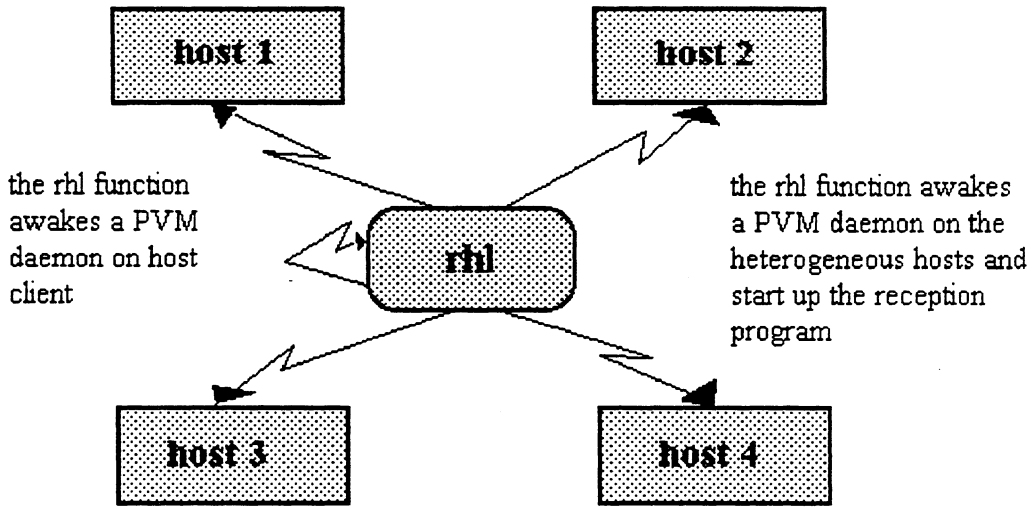


fig. 3

In detail, rhl_init actions are:

- activation of the PVM daemon on the server host. Hosts are added to the actual PVM configuration only for the duration of the application; the state of the cluster is then restored.
- activation of the server sides of the application (rhl_se) on all hosts specified in *host_list[]*.
- return of the error code.

rhf, whose name stands for **R**emote **H**ost **F**unction, transmits to a specified host the request of computing a certain function. All data are specified for what concerns their type and direction (input/output/both). All transmission of data needed for the server to compute the specified function are managed by RHL. It does *not* block waiting results to be available. The maximum quantity of data that can be transferred equals to 10MB, therefore overcoming the communication limit of PVM which forces packet length to 4KB.
Computed results are to be recalled later through the *rhd* function which will be soon discussed. This simple convention results in the capability of starting more than one thread-it is the way RHL uses to reach parallel processing.
rhf is a function with a variable number of parameters; its syntax is:

```
int *rhf (char *host_name, char *func_name,
                const char *format_string, arguments...)
```

where:
host_name is the TCP/IP logical name of a particular host on which we want to execute a particular function.
func_name is the name of the function that will be executed.
format_string is a string which, in a manner resembling quite similar C functions like *printf*, univocally specifies type, direction and dimension of arguments that follow.
The syntax of this string is:

"{ ***%direction[number]type*** }$_1^n$ "

direction is the character 'I' if the corresponding parameter is of input to the computed function, 'O' if it is an output parameter, 'B' if both i.e., it must be input to the function and changed by its execution. Finally, the character 'F' specifies the return value of the remote function;
number is an optional parameter which specifies that the corresponding parameter is a pointer to an area of the specified *number* of elements of the subsequently specified *type*;
type is a string of one or two characters describing the type of corresponding data to be passed as arguments to the remote function. Type identifiers are:

c char
d int
ld long int
sd short int
f float
lf double

The function returns an integer pointer which identifies the remote call.

Aim of this function is to collect data needed by the function in the server host, pack and deliver them. The first packet to be sent is a control packet which allows proper controls unpacking of data by the server and specifies the particular function to be run.

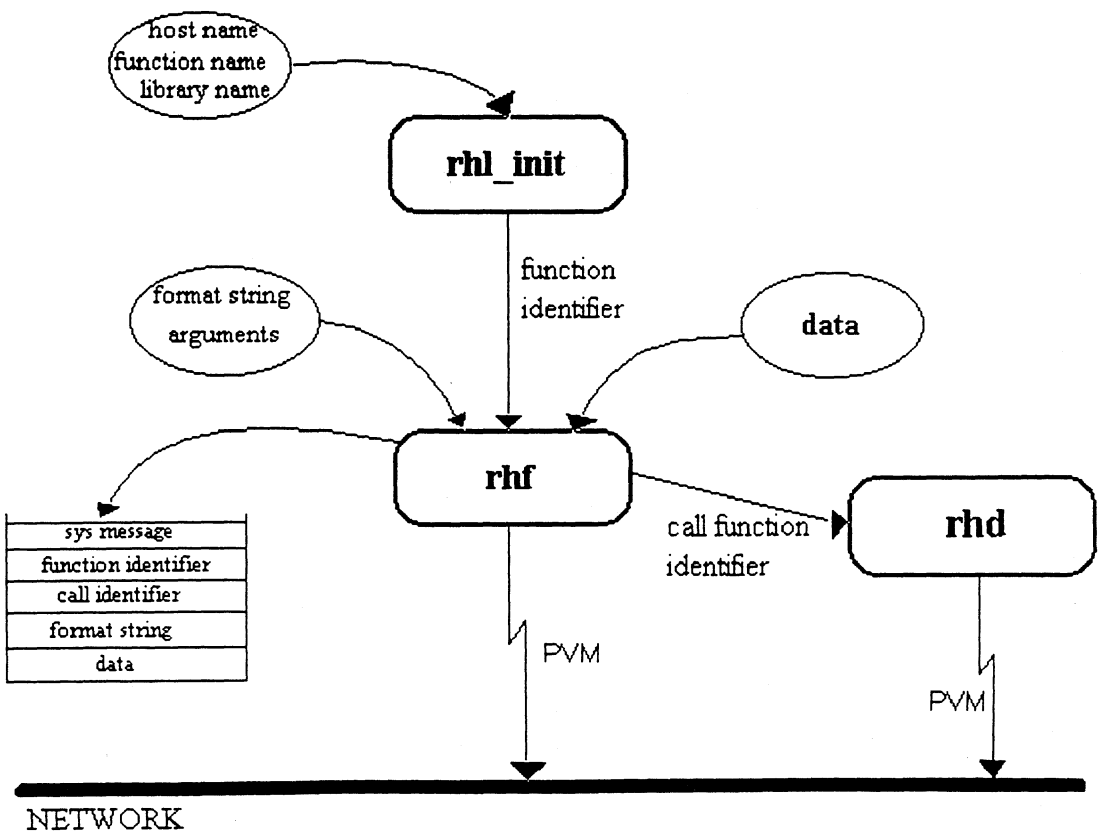


fig. 4

rhd (**r**emote **h**ost **d**ata) receives output data resulting from the call of the function specified in a previous rhf. His syntax is the following one:

int rhd (int *func_id);

where:
func_id is an integer code returned by a previously executed rhl. Such code uniquely identifies a function from any other.
rhd returns an integer error code.

Till we execute rhd we do not stop waiting results of an rhf. This simple statement means that we can start the execution of a new remote function **before** the previous one completed. Moreover rhd is a way to synchronize **subsequent** execution requests.

The use of this couple of functions, **rhf** and **rhd**, easily enables the introduction of parallelization and synchronization mechanisms into the user code.

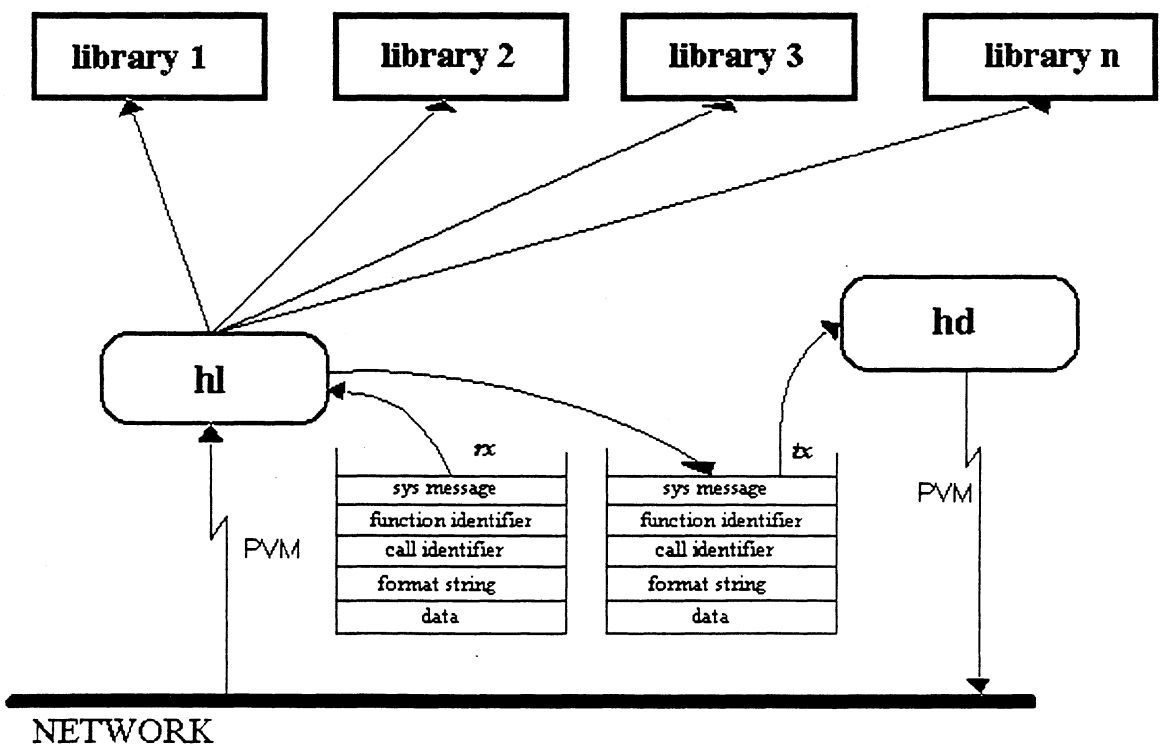


fig. 5

rhl_close (**r**emote **h**ost **l**ibrary **close**): kills all tasks activated on the servers and detaches hosts in the PVM cluster that were added by rhl_init, so the state of the PVM to the original state.

Performances

An example of distributed application between heterogeneous hosts that have been developed with great ease through the use of RHL is that in Listing #1. This program is a modification of a code developed for displaying images through the X window library.

This **RHL** application performs the following actions

1. grabs an image from a camera installed on the host *lagrange* (an hp9000/750)
2. filters that image on the host *convex* (a CONVEX C220)
3. displays the filtered image on the host that started the application: *shannon* (an IBM RS6000)

In our case we succeeded in exploiting the camera resource, peculiar of the host lagrange, and the computational power and the highly optimized libraries of a vector/parallel computer as the Convex C220.

In this case no parallelism is exploited between machines because the Convex machine waits for the image to be grabbed before filtering it. A simple extension would be the development of a pipe, using **rhf** and **rhd** functions, between lagrange and convex so to create a real time filtering of an image.

On the contrary, Listing #2 is an example of distributed application in which all hosts execute the same kind of operation in parallel. The example program numerically computes the integral of the function cos(x)*exp(sin(x)); the function which computes such integral with the trapeziums rule has been liked with the server side of RHL. The integration domain is decomposed in intervals which are divided among the hosts. The function which computes the integral is executed in parallel on the various server hosts and in the end of the computation data are requested by the client.

From the graphic in fig. 6 it is quite clear the advantage in executing the application in parallel between more hosts through RHL. Abscissa values represent the number of utilized hosts, ordinate values depicts execution time of the integral. Each curve represents a different subdivision of the domain of the function. Computers used in this experiment are (in the order in which they are added to the configuration):

shannon.rob.csata.it	IBM RS6000/320
asmit.rob.csata.it	IBM RS6000/320
convex.rob.csata.it	CONVEX C220
fourier.rob.csata.it	IBM RS6000/530
lagrange.rob.csata.it	HP 9000/750

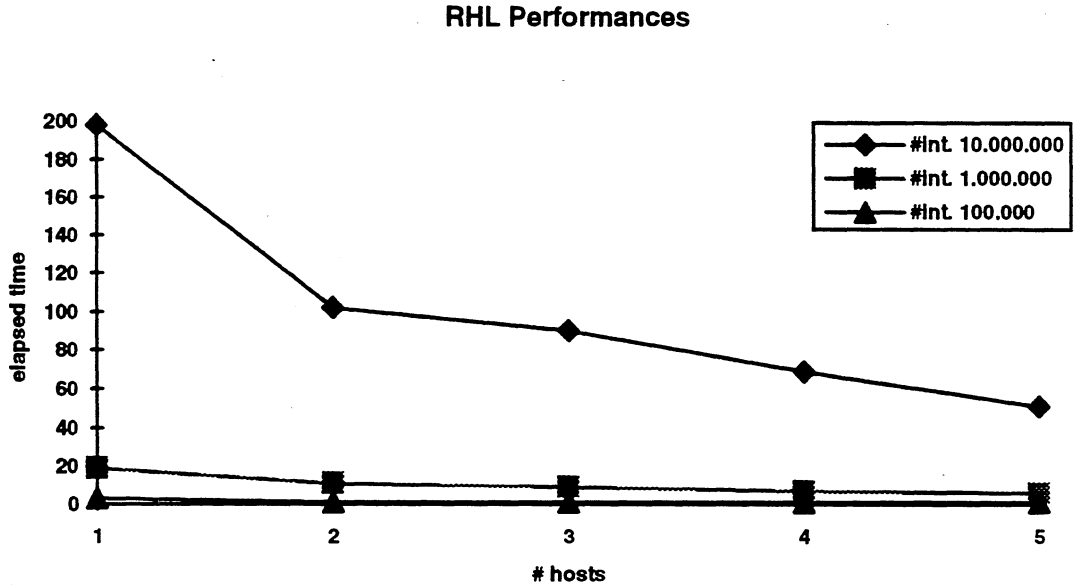


fig. 6

Conclusions

In conclusion we think that the RHL library can be of certain aid for the developer of distributed applications. Thanks to its ease of use it allows the development of quite complex applications able to exploit to the utmost the computing resources in a laboratory; in fact, by solving all communication problems between the hosts, it leaves as the only task of the programmer the establishment of the topology of his/her network for parallel and distributed computing.

The advantage in using of the RHL package is threefold:

- the use of architecture that best matches the task;
- the use of software functionalities available only on certain hosts in the PVM network:
- the use of special purpose devices whose drivers are available only on certain hosts in the PVM network.

The software have been tested for using on IBM RS6000, SUN SparcStation, HP9000, and CONVEX C220 machines.

```
main(argc, argv)
int argc;
char **argv;
{
    Window        w;
    char         *image;
    char         *mynew;
    XEvent        myevent;

    int num_host=2;
static char *lista_hosts[] = {"lagrange.rob.csata.it","convex.csata.it"};
    int hid[3],dim=512;
    int exitid,fil=0,tag=64,err=5;
    int chiamata[90];

    char *fn[] = {"lgrab","filtra"};

    image = (char *) malloc(512 * 512);

    Open_Display("");

    Allocate_Colors_Bw();
    w = Open_Window(508, 0, 512, 512); rhl_init(&num_host,lista_hosts, hid);


        if (hid[0] >= 0)
                rhd(rhf(lista_hosts[0],fn[0],"%B262144c %ld",image,dim)); if (hid[1] >= 0)
                rhd(rhf(lista_hosts[1],fn[1],"%F1d %B262144c %ld %ld",&err,image,fil,tag));

    mynew = Adjust_array(image, 512, 512);
    while (1) {
        View(w, mynew, 0, 0, 512, 512);
    }
    rhl_close(&exitid);
    free(mynew);
    fclose(fr);
}
```

listing 1

```
main()
{

int num_host=NUM_HOST;
char *lista_hosts[] = HOSTS;
char *fn[] = {"integrale"};
int hid[NUM_HOST];
int exitid;
int chiamata[NUM_HOST],i;
int int_parz,resto_int;
double amp_int_parz;

double ris[NUM_HOST],min[NUM_HOST],max[NUM_HOST],risultato=0;

int interv[NUM_HOST];

rhl_init(&num_host,lista_hosts, hid);

int_parz=INTERVALLI/NUM_HOST;
amp_int_parz = (MAX-MIN)/NUM_HOST; resto_int=INTERVALLI%NUM_HOST;

for (i=0; i<NUM_HOST;i++) {
    min[i]=amp_int_parz*i+MIN;
    max[i]=amp_int_parz*(i+1)+MIN;

if ((resto_int != 0) && (i == (NUM_HOST-1))) int_parz += resto_int;

    if(hid[i] >= 0) {
            chiamata[i]=rhf(lista_hosts[i],fn[0],"%F1lf %llf %llf %ld", &ris[i],min[i],max[i],int_parz);
            }
    }

for (i=0; i<NUM_HOST;i++)
    if (hid[i] >= 0)
            rhd(chiamata[i]);

rhl_close(&exitid);

for (i=0; i<NUM_HOST;i++)
            risultato += ris[i];

printf("risultato=%22.20lf\n",risultato);
fflush(stdout);
}
```

listing 2

Distributed memory, parallel computing with dynamic load-balancing on heterogeneous network of workstations and a CONVEX C3210 computer

K. Boryczko[1], J. Kitowski[1,2], J. Mościński[1,2] and M. Pogoda[1]
[1]Institute of Computer Science, AGH, al. Mickiewicza 30, 30-059 Cracow, Poland
[2]Academic Computer Centre CYFRONET, ul. Nawojki 11, 30-950 Cracow, Poland

1 Introduction

Network-based multicomputers constitute the emerging parallel architecture at present. Clusters of several or more powerful workstations and computers built around general networks such as Ethernet or FDDI easily enter the field of the most advanced supercomputing. Use of diverse computing resources, or metacomputing, empowers users by providing transparent access to a variety of computer services.

To get adequate parallel speedup and efficiently fetch the idle cycles of the networked computers during normal load, the algorithm with load-balancing procedure and the programming environment should be selected and optimized properly. Another important element is a method for evaluation of such a network in its standard load to define its capability for large-scale distributed computing.

In this paper we present a simple load-balancing procedure for the short-range Lennard-Jones molecular dynamics (MD) algorithm for distributed computing in PVM environment. Timings are presented for a heterogeneous (Ethernet) network of workstations including also CONVEX C3210 as one of the nodes.

Another problem discussed in the paper is network evaluation for large-scale computing. For this purpose a parallel version of the standard SLALOM benchmark and EP kernel from the NAS parallel benchmarks were adopted.

2 Load-balancing procedure for distributed MD simulation

Resume of the MD algorithm

In MD approach, the time evolution of the system of particles is simulated. The particles dynamics is determined by relations reflecting their mutual interactions rather than by the averaged fields. The equations representing the Newtonian laws of motion are solved every timestep for each particle. The size of simulated system is limited by high computational demand.

In this study an algorithm with short range Lennard-Jones interactions is adopted, with the computational box being a long cylinder. Traditionally, the total energy, E, for a fixed number of particles, N, in a fixed volume, V, is conserved (simulation of the microcanonical ensemble). The algorithm is useful for studies of fluid and mixtures in microcapillaries or growth of optical fibres [1]-[3]; it is called the *Pipe Method* [4]-[6]. To resume the algorithm, a brief overview is given below.

Periodic boundary conditions (*pbc*) are introduced along the cylinder axis only (in z-direction). Since in fluid of uniform density the number of neighbours interacting with a given molecule is nearly constant – on a base of the cutoff radius, r_C, the integer cutoff number n_C is introduced, where n_C is the number of neighbours interacting potentially with a given particle, and N is the number of particles in the computational box. Particles in the cylinder are sorted due to their z coordinates and the index vector

is set up to return to original particle indices. x and y particle coordinates are gathered due to the index vector. During the calculations of forces, the computational cylinder is shifted step-by-step in respect to its copy to subsequent neighbouring particles from 1 to n_C. The algorithm is suitable for vector and parallel processing.

Distributed approach

For distributed computing on a network of computers domain decomposition is proposed. The cylindrical computational box is divided along z direction into loosely coupled domains allocated on different computers [7, 8]. In PVM environment, one of the computers is the host and the rest are nodes (see Fig.1).

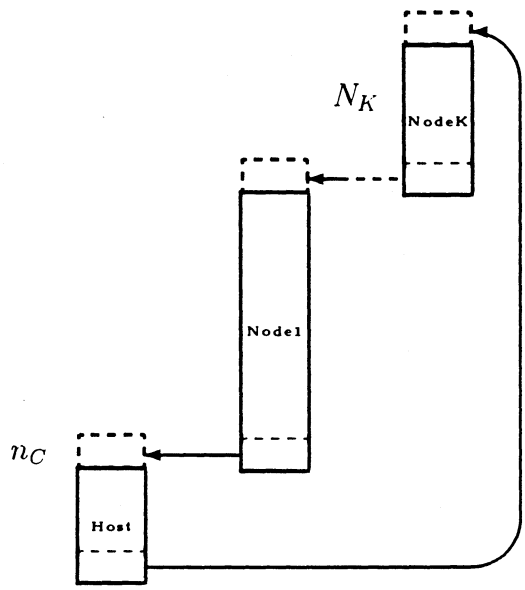


Figure 1: Domain decomposition of the pipe method.

The following elements are decomposed and calculated independently in the domain: interparticle interactions, interactions between the particles and the wall of the computational box, solution of the Newtonian equations of motion, physical quantities and sorting. For K PVM nodes, the number of particles in $(K+1)$ domains, N_k $(k = 0, \ldots, K)$, varies, resulting from the load-balancing procedure. At the beginning of simulation, number of particles in every node, including host, is the same, N_k^0, thus the overall number of particles in simulation, N, is equal to $N = (K+1)N_k^0$. The communication is needed only for particles moving from one domain to another (if determined by the motion equations or by the load-balance procedure), for interparticle forces calculations at the boundaries of the domains and for sending partial global results from each domain to evaluate global quantities of the whole box.

For forces calculations at the boundaries the first n_C particles positions from the "upper" domain are sent to the "lower" one, increasing its number of particles by n_C (cf. Fig. 1), to direct forces calculations, which are in turn sent back as reaction forces to the "upper" domain.

The host program differs from the node ones in some additional work for control, input and output, computing of physical quatities and the load-balancing. Since the most time consuming calculations are well decomposed, performance of the host and node programs is similar.

For the present work a load-balancing procedure is included for computing on heterogeneous networks of computers heavily used by other users.

Criterion for the load-balancing (LB) is to get the same execution *wall-clock* time, on each node, measured with function `gettimeofday()`. The criterion takes into account the individual performance of each computer in the network and its current load. To fulfil the criterion, particles should be dynamically shuffled between the nodes according to the balance function.

The efficiency of the algorithm (without LB) executed on a homogeneous network of computers, separated from the external load, or on a multiprocessor (when each domain contains the same number of particles) is close to 1 [8], thus the algorithm scales well with the number of processing nodes. So the simple balance function, δ_k, with low computational complexity is proposed:

$$\delta_k = N_k\,(1 - \frac{T_k}{< T >}), \qquad k = 0, \ldots, K, \tag{1}$$

where T_k is the wall-clock time of the current simulation step for k computer ($k = 0, \ldots, K$) and

$$< T >= \frac{\sum T_k}{K+1}. \tag{2}$$

δ_k defines the number of particles to be shuffled from or to the k computer according to δ_k sign, proportionally to the difference in execution wall-clock time for computers in every timestep. Some additional verifications are required taking into account the smallest possible number of particles on a computer resulting from n_C, conservation of the total number of particles in the system and particles dynamics causing their moving between computers.

3 Benchmarking the distributed architecture

The performance evaluation of a network of computers for large-scale computing is a much more complicated task in comparison with benchmarking sequential or vector machines. This task critically depends on the problem, the algorithm adopted and the way the distribution of workload is made. The last problem concerns parallel languages or programming systems (for example HP Fortran [9], PCN [10] or Network Linda [11]) as well as message-passing programming environments (typical examples are PVM [12], ParaSoft Express [13] or p4 [14]). A more uniform message-passing environment is under development [15].

Many benchmarks are known for sequential, vector and multiprocessor architectures. Some of them, specially prepared for multiprocessors and network of computers are still in progress, as Parkbench suite for example [16].

We have adopted two of the well known, classical benchmarks – SLALOM and EP kernel from NAS parallel benchmarks for distributed computing and try to compare on that base the network of heterogeneous computers with other machines.

Distributed SLALOM benchmark

SLALOM (Scalable Language-independent Ames Laboratory One-minute Measurement) [17] is the first benchmark based on fixed time rather then fixed problem comparison. The fixed time seems to be more representative of the way people use computers. It also increases the scope and longevity of the benchmark. The benchmark solves a complete, real problem – optical radiosity on the interior of a box and takes into account input, problem setup, solution and output.

As originally implemented, SLALOM used a direct solver (by LDL factorization) for solving a set of linear equations. For a problem of size n patches the solver grows as n^3 and dominates the time for large values of n. Application of the iterative conjugate gradient method results in lowering the computational complexity to $o(n^2)$ and constitutes the new method used in the solver. For comparison of the methods MFLOPS cannot be compared directly, so the number of pathes should be used.

In our study we adopted the nCUBE version of the algorithm using the old method (i.e. the direct solver by LDL factorization). The code has been rewritten from Fortran 77 into C language

In the distributed algorithm the plane (x, y) is divided into squares of patches allocated on different computers and necessary communication is organized at domain boundaries. k_x and k_y are numbers of computers in tiles in x and y directions. Numbers of patches in x and y directions are the same, $n_x = n_y$, and number of patches for $k = (k_x, k_y)$ computer in x and y directions are $n_{k_x x}$ and $n_{k_y y}$. For the algorithm without LB $n_{k_x x} = n_{k_y y}$, so the equal squares of patches are allocated on the computers.

The cost of computing on k computer depends on $n_{k_x x} \times n_{k_y y}$; changing $n_{k_x x}$ and $n_{k_y y}$ one gets possibility to adapt the demand to the current external load of the computers.

In the proposed algorithm static LB is adopted, i.e. during the first phase of the benchmark a short test of SLALOM is run to define the proper batch for the computers by $n_{k_x x}$ and $n_{k_y y}$ choice. The speed measure for $k = (k_x, k_y)$ computer, s_k, is defined:

$$s_k = \frac{a}{T_k} \tag{3}$$

where T_k is wall-clock time for k computer of the short test and a is a parameter, $a = 1500$. Using the ratio:

$$r_{k_x k_y} = \frac{s_k}{\sum s_k} \tag{4}$$

one gets:

$$n_{k_x x} = \sqrt{n_x n_y r_{k_x 1}} \quad \text{for} \quad \forall k_x \tag{5}$$

and

$$n_{k_y y} = r_{1 k_y} \frac{n_x n_y}{n_{1x}} \quad \text{for} \quad \forall k_y. \tag{6}$$

Distributed EP kernel

The Embarrassingly Parallel (EP) kernel from the NAS parallel benchmarks [19, 20] is adopted as a second benchmark for the network of computers. It generates pairs of Gausian random deviates according to a specific scheme and tabulates the number of pairs in successive square annuli.

The original EP kernel has been rewritten from Fortran 77 into C language. On every node the complete program is executed with the original loop 150 distributed between the nodes, see Fig.2.

```
      DO 150 K = 1, NN
      ..................
      ..................
-----------------------------------
      for (k = lnt; k < unt; k++)
      {
      ..................
      ..................
```

Figure 2: Distribution of the 150 loop of EP kernel to sequential chunks, with boundaries defined by different `lnt` and `unt` for each computer separately.

In Fig.2, `NN`= 2^{m-m_b}. Problem size, $n = 2^m$, defines the number of complex pairs of uniform (0,1) random numbers used for generation of Gausian random deviates and $n_b = 2^{m_b}$ is the size of each batch of uniform random numbers. m_b can be set for convenience on a given system, since it does not affect the results. `lnt` and `unt` arise from the load-balancing applied for the network of heterogeneous

computers.

Two methods are applied for the static load-balancing defined at the begining of the basic benchmark. They are similar to those proposed for the SLALOM benchmark (Eq.3,4). In every case the first phase of the run is to obtain the effective speed measure of computers of the network during their normal load. For this purpose a short test is run on the computers just before the basic benchmark to get **lnt** and **unt**.

In the first method the test is based on floating point multiplication, division and logarithm; each operation is invoked $l_{1,2,3} = 1000$ times. First two have the weight $w_{1,2} = 1$, while for the last $w_3 = 2$. For k computer, the speed measure, s_k, is defined as

$$s_k = a_1 \frac{\sum w_i l_i}{T_k} \tag{7}$$

where T_k is wall-clock time in seconds and a_1 is a constant ($a_1 = 1500$). **lnt** and **unt** on k computer is obtained as a part of the continuous range **1,...,NN** (Fig.2) using the ratio r_k

$$r_k = \frac{s_k}{\sum s_k}. \tag{8}$$

In the second method the test is based on EP kernel run for small problem size ($n = 2^{16}$, $m_b = 8$). In this case

$$s_k = \frac{a_2}{T_k} \tag{9}$$

and r_k is obtained similar to Eq.8, where T_k is wall-clock time and $a_2 = 1000$.

4 Results

In every case of distributed computing with CONVEX C3210 as a computing node, PVM environment was used with SUN SPARCserver 470 as a host. The experiments were performed during the standard load of the computers; especially C3210 was heavily used. On the Ethernet network, besides C3210, the workstations shown in Table 1 were applied.

Computer (name)	Clock, MHz	RAM, MB	OS and opt. level
SUN SPARCstation SLC (SLC)	20	8	SunOS v4.1.1; -O3
SUN SPARCserver 470 (S470)	40	32	SunOS v4.1.1; -O3
SUN SPARCstation 2 (SS2)	40	32	SunOS v4.1.1; -O3
IBM RS/6000-320 (RS6)	20	16	AIX v3.1; -O
HP Apollo 9000/720 (HP9)	50	16	HP-UX 8.05; -O
CONVEX C3210 (C2)	40 (ns)	128	ConvexOS 10.0; -O2

Table 1: Comparison of computers.

Distributed MD computing

At the beginning of simulation number of Argon particles on every computer was the same, $N_k^0 = 32004$ ($r_C = 2.5\sigma$, where σ is the Lennard-Jones parameter). Host and node programs were coded in C with vector version of the algorithm for C3210 adopted from Fortran [4, 5] and vectorized (with option -O2 and vector directives). On Convex `renice=2` parameter was introduced.

In Fig.3 the execution wall-clock time of the distributed program for every timestep, T, is presented for two short runs. Computing in the domains, k, needs synchronization, then $T = \max_k T_k$. With LB method proposed (Eq.1) the execution wall-clock time is significantly reduced.

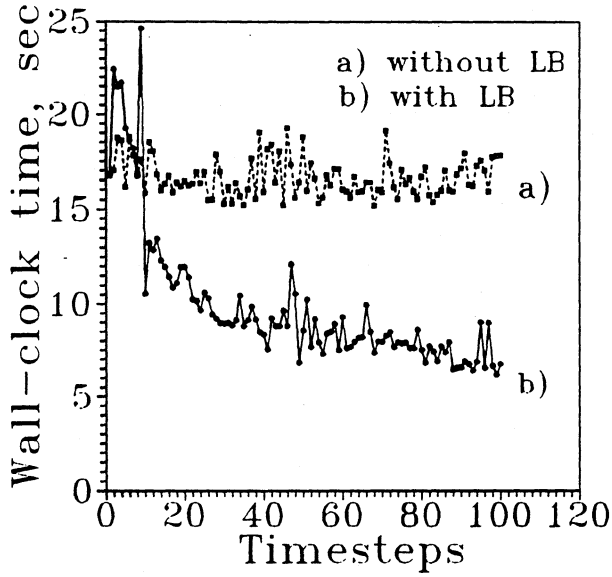


Figure 3: Wall-clock time for every timestep, T, for the network of 7 machines: S470, C2, 2*SS2, HP9, 2*RS6. Standard load. PVM v2.4.

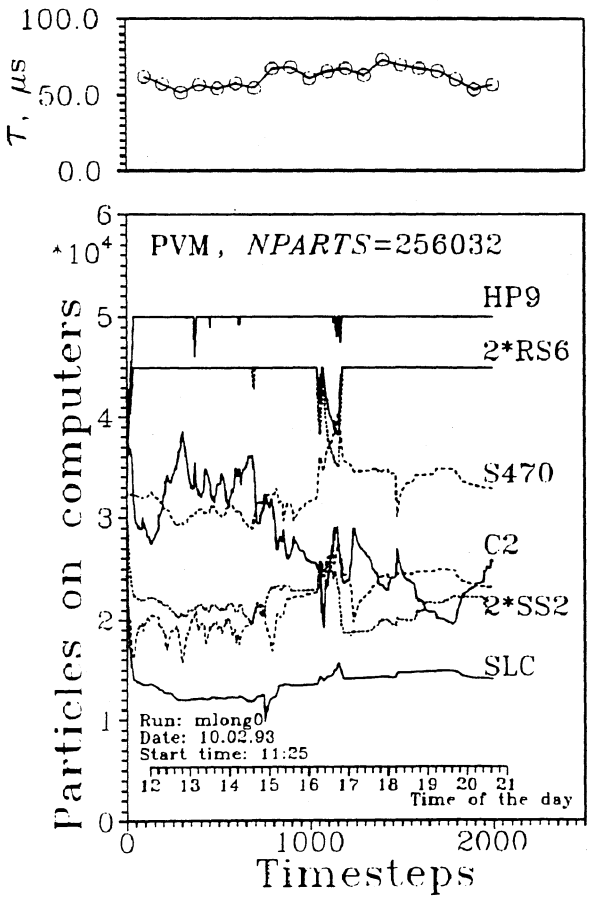


Figure 4: Execution wall-clock time, τ, and particle distribution between 8 processing nodes. Standard load. PVM v2.4.

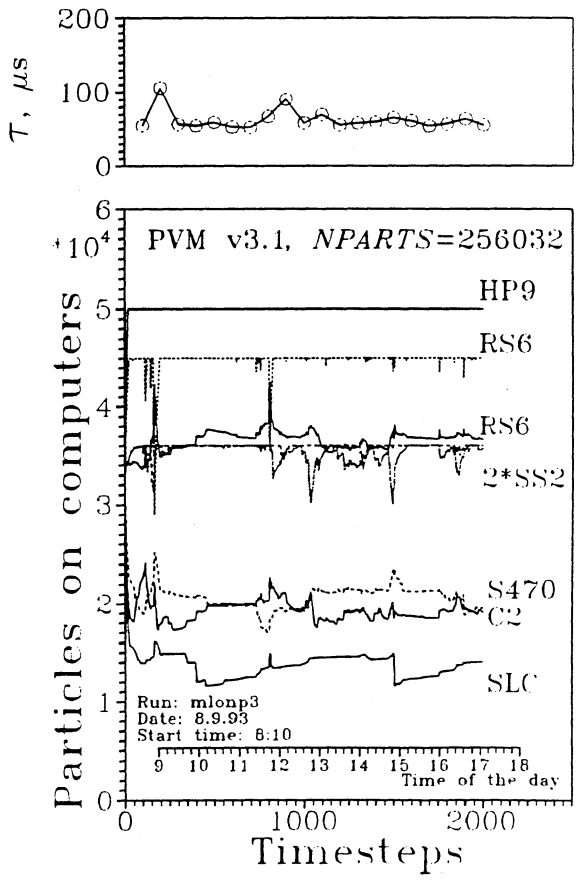


Figure 5: Execution wall-clock time, τ, and particle distribution between 8 processing nodes. Standard load. PVM v3.1.

In Figs.4 and 5 the execution wall-clock time per timestep and particle, τ, and particle distribution between computers are presented for two long runs on the network of 8 heterogeneous machines. In these cases, τ is obtained for every 100 timesteps of simulation. For some workstations the constant, maximum allowed distribution was attained – this feature originates from the small system memory (for HP9) or from the simultaneous use of the workstations for other jobs (for SS2). Using the limits, swapping is reduced.

No significant difference in performance between PVM v2.4 and v3.1 is observed. For the whole runs, $\tau_{24}^{avg} = 61\mu s$ and $\tau_{31}^{avg} = 63\mu s$ for PVM v2.4 and v3.1 respectively. For Convex C3210 cpu time is equal to $64\mu s$ [8]. In practice, job turnover time for C3210 could be much higher. Another benefit from the distributed computing is the large (distributed) memory available.

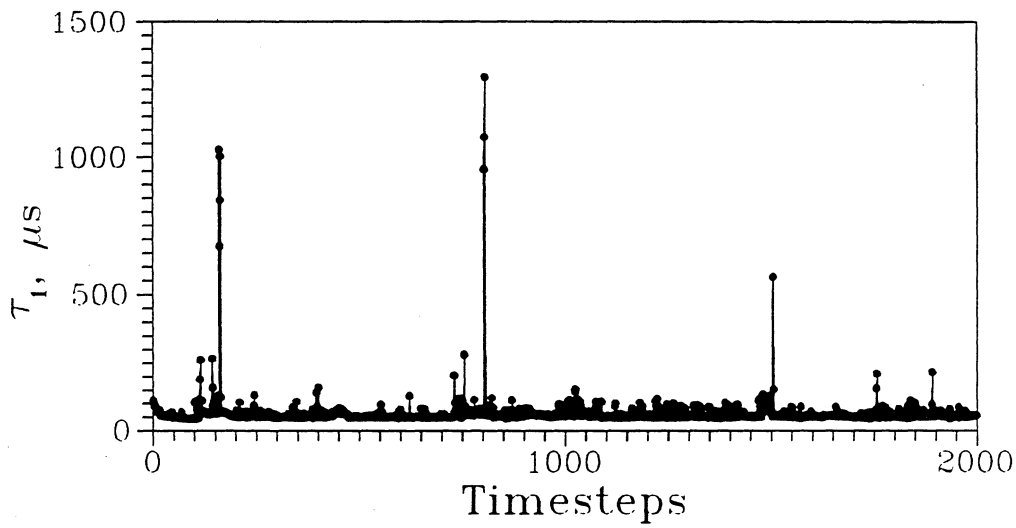


Figure 6: Execution wall-clock time per particle for every timestep for run presented in Fig.5.

In Fig.6 the execution wall-clock time per particle for every timestep, τ_1, is presented for the run shown in Fig.5. The peaks represent dramatic changes in the external load of the computers or resending the data between them, which influence also the particle distribution. However, these elements do not degrade much the performance of the distributed computing.

Distributed SLALOM benchmark

In our experiments we used the standard C compilers from the Unix operating system; the results are averaged over 3-5 trials. For test purposes we also ran the original C language sequential SLALOM code (with the new version of the algorithm – the conjugate gradient method) on the computers of our network.

Computers	MFLOPS [17, 18]	MFLOPS
HP9	—	4.1
RS6	5.6 (xlf)	3.9
SS2	—	1.4
S470	—	1.4
SLC	—	0.7
SUN 4/370, 25 MHz	1.75 (ucc)	—

Table 2: Performance of sequential SLALOM using conjugate gradient method.

In Table 2 these results are compared with those known from the literature – we believe that the

difference in performance arises from the different compilers: **xlf** Fortran 77 compiler for RS6 and ucc for SUN 4/370 were used respectively [17, 18].

No.	Computers in the network	without LB		with LB	
		Patches	MFLOPS	Patches	MFLOPS
1.	S470, C2, SS2, SLC	42	0.61	—	—
2.	S470, C2, RS6, SLC	42	0.63	151	2.2
3.	S470, C2, RS6, RS6	129	1.72	387	4.1
4.	S470, C2, RS6, HP9	131	1.70	394	4.3
5.	S470, C2, 4*RS6, 2*SS2	107	1.47	475	6.3

Table 3: Performance of distributed SLALOM without and with load-balancing using LDL method. Standard load. PVM v2.4.

Some preliminary results of the distributed SLALOM algorithm (the old version, **with LDL** method) are presented in Table 3. No optimization of the algorithm for vector processing was performed, only -O2 was applied. MFLOPS are based on the wall-clock time (using **gettimeofday()**). In the first phase of the runs with the static LB - the test cases of 15 s were performed.

We studied the performance of the network of heterogeneous computers starting with configuration with low-end machines and further replace them by the computers with higher performance. For the algorithm without LB the performance is limited by the slower computer in the network (compare items 1 with 2 and 3 with 4). Using the algorithm with LB one gets the significant increase in performance in spite of the standard load of the computers.

For nCUBE 2 with 4 processors (20 MHz, Fortran+coded BLAS) the performance in (Patches/MFLOPS) is 560/3.8 [18]. It is roughly similar to 475/6.3 obtained for 8 networked computers (item 5 in Table 3). For MasPar MP-1 (12.5 MHz, C) with 2048 processors one gets 1055/22 [18], which means that in the reported experiments the tested network is rated higher than 0.45 Patches performance and 0.28 MFLOPS performance of MP-1 with 2048 processors. Due to the standard load of the computers of the network the distributed performance is close to its lower limit.

Parallel efficiency for nCUBE is rather low, especially for Patches measure. For nCUBE with one processor the measure is 354/1.13 [18].

Since the distributed algorithm is based on the parallel algorithm developed for multiprocessors, for which the communication overhead is lower than for the network (due to their more sophisticated architecture and speed of the links between the nodes) — we conclude that the performance measure of distributed computations obtained on the network shows its ability for practical applications.

Distributed EP kernel

Results of the EP kernel runs on the heterogeneous network of computers (with PVM v2.4) are presented in Fig.7. for the problem size $n = 2^{24}$, $m_b = 16$ and 64 bit arithmetics. At most, 5 computers were used in the experiments: S470, C2, SLC and two RS6 (Table 1). For Convex compilation was performed with -O2 option only, neither modification of the algorithm nor vector directives were applied. The timings are averaged over 3-5 runs in every case. Since the experiments were performed during the standard load of computers, high dispersion of the results was observed.

Two methods of the static load-balancing (LB) as proposed in Eqs.7-9 are applied. They are compared with results without any load-balancing procedure, in which intervals (**unt** - **lnt**) (see Fig.2) are the same for every computer.

In order to verify efficiency of the load-balancing methods the speed measure for the computers, s_k (as

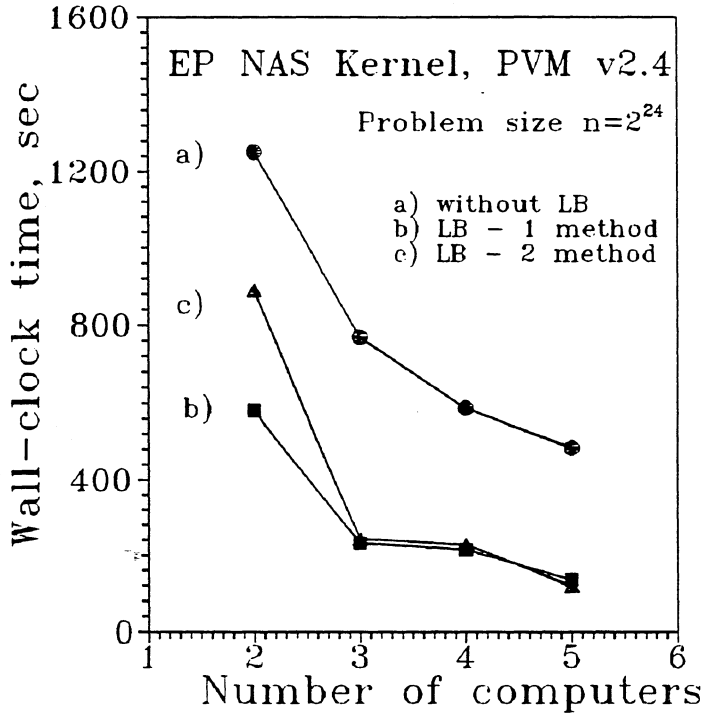


Figure 7: Wall-clock time for EP kernel run on the network of computers. The sequence of computers included: S470, C2, RS6, SLC, RS6.

defined in Eqs.7 and 9), obtained during the first phase of the benchmark is shown in Fig.8. The values result from averaging for the same runs with 5 computers presented in Fig.7.

The differences in speed measure reflect the high load of the Convex computer used in the experiments in comparison with other computers which were only lightly loaded. Except Convex, the numbers show the approximation of the relative performance of these machines.

Both methods of the static load-balancing decrease the wall-clock time in similar way, thus the form of the static load-balancing procedure does not influence much the efficiency of the distributed EP kernel.

	cpu time, s	
Computer	$n = 2^{24}$, $m_b = 16$	$n = 2^{28}$, $m_b = 19$
RS6	315.4	$\sim$ 5000.
S470	719.1	$\sim$ 14000.
SLC	1319.	—

Table 4: Performance of the sequential EP kernel in C language, for two problem sizes.

For the problem size $n = 2^{24}$ and $m_b = 16$, the results can be informally compared with those obtained from sequential processing (Table 4) and others presented in [19] for one processor of the Cray Y-MP, for which EP benchmark (in Fortran 77) takes 11.6 s cpu time. Although it is unusual to compare cpu with wall-clock time, informally we can estimate, that 5 networked computers (with the standard load) for which the wall-clock time is equal roughly to 125 s, are rated higher than 2.5 performance of IBM RS/6000-320 and higher than 0.1 performance of the Cray Y-MP.

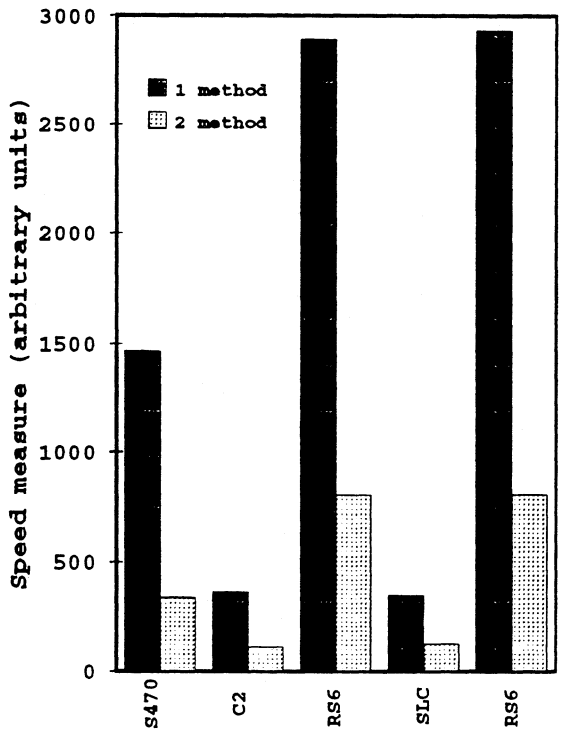


Figure 8: Speed measure,s_k, for the computers of the network. The sequence of the listed computers: S470, C2, RS6, SLC, RS6.

Computer	Number of processors	Execution time, s [19] (Fortran 77)	Wall-clock time, s (C language)
BBN TC2000	64	284	—
CM-2	32K	50.9	—
iPSC/860	128	61.3	—
Y-MP	1	150.1	—
Y-MP	8	20.1	—
Computer network	5	—	2960

Table 5: Performance of the parallel EP kernel for $n = 2^{28}$ problem size.

In Table 5 the wall-clock timing is shown for the problem $n = 2^{28}$ and $m_b = 19$ run on the same network of 5 heterogeneous computers during the standard load. It is compared with the execution time for different multiprocessors [19]. Estimating, this network (with 5 computers) for this greater problem size is rated higher than 0.05 of Cray Y-MP with one processor. In practice, job turnover time could be similar for CRAY Y-MP and the network of computers.

In spite of many difficulties in formulation of parallel algorithms and evaluation of the network performance the use of distributed computing seems to be at the beginning for successful run.

Acknowledgments

The authors thank Mr. W. Alda for the help during the computer experiments. K.B. is grateful to Polish Scientific Committee (KBN) for partial financial support under Contract PB 8 S503 021 05.

References

[1] J.M.D. MacElroy and S.-H. Suh, "Computer simulation of moderately dense hard-sphere fluids and mixtures in microcapillaries" *Mol. Phys.*, **60**, 2 (1987) 475.

[2] U. Heinbuch and J. Fischer, "Liquid argon in cylindrical carbon pore: molecular dynamics and Born-Green-Yvon results", *Chem Phys. Lett.*, **135**, 6 (1987) 587.

[3] A.M. Ferrenberg and M.J.P. Nijmeijer, "A parallel molecular dynamics algorithm for simulation of the growth of optical fibres", paper presented at 1993 PVM Users' Group Meeting, Knoxville, May 10-11, 1993.

[4] J. Mościński, J. Kitowski, Z.A. Rycerz and P.W.M. Jacobs, "A vectorized algorithm on the ETA 10-P for molecular dynamics simulation of large number of particles confined in a long cylinder", *Comput. Phys. Commun.*, **54** (1989) 47.

[5] J. Mościński, M. Bargiel, J. Kitowski, Z. Skotniczny, Z.A. Rycerz and P.W.M. Jacobs, "Vectorized molecular dynamics algorithms for very large number of particles", *Proc. of 1989 International Conference on Supercomputing*, June 5-9, 1989, Crete, Greece (ACM Publications, Baltimore, USA).

[6] J. Kitowski and J. Mościński, "Microcomputers against Supercomputers ? – On the geometric partition of the computational box for vectorized MD algorithms", *Mol. Simul.*, **8** (1992) 305-319.

[7] K. Boryczko, M. Bubak, J. Kitowski, J. Mościński and M. Pogoda, "Molecular dynamics and lattice gas parallel algorithms for transputers and networked workstations", EUROMECH Coll.287, Sept. 21-25, 1992, Cagliari, Italy; also *Transport Theory and Statistical Physics*, in print.

[8] J. Mościński, K. Boryczko, J. Kitowski, M. Pogoda, W. Alda, M. Bubak, W. Dzwinel, R. Slota and M. Noga, "Distributed molecular dynamics on a cluster of workstations and Convex supercomputer", *Proc. of the Convex Worldwide User Group Conference*, March 21-25, 1993, Richardson.

[9] "High Performance Fortran Language Specification",
"High Performance Fortran Journal of Development", High Performance Fortran Forum, May 3, 1993, Rice University.

[10] Foster, I., and Tuecke, S., "Parallel programming with PCN", *Argonne National Laboratory*, ANL-91/32 rev.2, January 1993.

[11] Carriero, N.J., and Gelernter, D.H., "Linda in context", *Comm. ACM* **32** (1989) 444-458.
Bjornson, R., Kolb, C., and Sherman, A., "Ray Tracing with Network Linda", *SIAM News*, **24** 1 (1991).
Carriero, N., Gelernter, D., Kaminsky, D., Westbrook, J., "Adaptive paralelism with Piranha", Yale University, 1993 (draft).

[12] Sunderam, V.S., "PVM: a framework for parallel distributed computing", *Concurrency: Practice and Experience*, **2** (1990) 315-339.
Geist, A. Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., and Sunderam, V., "PVM 3.0 A users' guide and reference manual", *ORNL/TM-12187*, USA (February, 1993).

[13] "Express v3.0 Introductory Guide".
"Express v3.0 Operational Manual", *ParaSoft* Corp., Pasadena (1990).

[14] Butler, R., and Lusk, E., "User's guide to the p4 programming system", *Argonne National Laboratory*, ANL-92/17, October 1992.

[15] Dongarra, J., Hempel, R., Hey, A.J.G., Walker, D.W., "A proposal for a user-level, message passing interface in a distributed memory environment", June 1993, ORNL/TM-12231.

[16] J. Dongarra, private information (1993).

[17] J. Gustafson, D. Rover, S. Elbert and M. Carter, "SLALOM – Surviving Adolescence", *Supercomputing Review*, **Dec. 91** (1991) 54.

[18] "The SLALOM Benchmark Report", unpublished results (1990).

[19] D.H. Bailey, E. Barszcz, J.T. Barton, D.S. Browning, R.L. Carter, L. Dagum, R.A. Fatoohi, P.O. Frederickson, T.A. Lasinski, R.S. Schreiber, H.D. Simon, V. Venkatakrishnan and S.K. Weerantuga, "The NAS parallel benchmarks – summary and preliminary results", private information.

[20] D.H. Bailey, J. Barton, T.A. Lasinski and H.D. Simon, "The NAS parallel benchmarks", Report RNR-91-002 Rev.2 (1991), NASA Ames Research Center, Moffett Field.
"NAS parallel benchmark results", *Proc. Supercomputing'92* (1992) 386-393.
"NAS parallel benchmark results", *IEEE Parallel and Distributed Technology*, **Feb. 1993** 43-51.

Parallel Algebraic Computations on Meta Series

Thomas Friedrich, Wolfgang Lassner
University of Leipzig
Augustusplatz 10-11, 04109 Leipzig, Germany

Abstract

New computer facilities of the CONVEX Meta Series at the University Leipzig have stimulated their use for distributed and parallel algebraic computations with REDUCE.
Our metacomputing environment consists in a cluster of eight HP-RISC nodes connected by FDDI with a CONVEX C 3820. It was in that configuration the first Meta Series installation in Germany [1].
The system forms together with a C 220 (128 MB memory and 10 GB disks) and more than 28 HP - Workstations, among them eight diskless HP 9000/715-33 workstations and two HP 9000/735 CRX graphic servers, the hardware basis at our Computer Center for its new service.
The eight HP 9000/735-99 of the Meta Series cluster have different memory sizes: 4×32 MB, 2×80 MB, and 2×144 MB. They use locally 4×1 GB disks and 4×2 GB disks. However, as a first observation to new users, his home directory for all nodes is usually located on one of the C 3820 disks. Our two processor C 3820 with 512 MB memory and 30 GB disks offers its server capabilities by Ethernet to the main users (quantum chemistry, molecular physics, meteorology, environmental research). Main use of the heterogeneous cluster is realized through batch queues handled by ConvexNQS+.
The computer algebra group at the Department of Mathematics and Computer Sciences has developed several packages for Gröbner bases calculations in commutative and noncommutative algebra.
Now our first distributed and parallel applications are running on the Meta system.
The problem we solve has its origin in Sophus Lie's method to solve differential equations using symmetry analysis. There are today worldwide about ten computer algebra packages which determine the symmetry algebra of differential equations.
If the symmetry algebra is determined as a finite dimensional Lie algebra it remains the identification among Lie algebras from known classification tables. The problem whether two Lie algebras are isomorphic can be decided by the existence of solutions of a system of polynomial equations due to the following theorem.

Theorem Two N-dimensional algebras L and $\tilde{L}$ over $\mathbf{F}$ given by the structure constants $\{c_{ij}^k\}$ and $\{\tilde{c}_{ij}^k\}$ are isomorphic if and only if the system of equations

$$a_i^k a_j^l c_{kl}^n - \tilde{c}_{ij}^m a_m^n = 0, \quad d - det(a_i^j) = 0$$

in unknowns $\{a_i^j, d\}$, $(i, j = 1, \ldots, N)$ has a solution in $\mathbf{F}$ with $d \neq 0$.

A computational method to identify Lie algebras by Gröbner bases calculations has been developed and implemented in a REDUCE program ISOLIE/ASYS [2].
Now a parallelized version makes use of the following facts and developments.

The Meta Series cluster is recommended for coarse-grained parallel computing under ConvexPVM.
REDUCE is a computer algebra system written in Portable Standard Lisp (PSL), running on several platforms among them on CONVEX C Series and on HP 9000/735. REDUCE is very fast in its implementation on HP 9000/735 [3].
RR [4] a REDUCE interface for PVM supports the use of the computer algebra system REDUCE and the symbolic language Portable Standard LISP in parallel on several processing nodes using the service of PVM.
The Gröbner bases package CALI [5] was parallelized for our applications. The method makes use of factorized Gröbner basis calculations with constrains. A master slave concept is realized in REDUCE. It runs on the Meta Series cluster under PVM using RR – the parallel algorithm support for PSL based REDUCE.
The parallel Gröbner basis calculations have been applied to the identification of the five-dimensional symmetry Lie algebra of the known Burgers equation. Computing times have been measured and compared with a sequential treatment of the problem [6] and within different parallelizations. Some technical problems with diagnostic tools and PVM stability will be discussed.

References

[1] Friedrich Th. Man kann wieder rechnen mit/an der Universität Leipzig. Universitätszeitung 5/93, p. 12-13, 1993.

[2] Gerdt V.P., Lassner W. Isomorphism Verification for Complex and Real Lie Algebras by Gröbner Basis Technique. To be published in the Proceedings of International Workshop "Modern Group Analysis: Advanced Analytical and Computational Methods in Mathematical Physics" (Acireale/Catania, Italy, October 27-31, 1992), M.Torrisi (Ed.), Kluwer Academic Publishers, 1993.

[3] Hearn A.C. REDUCE User's Manual. Version 3.4. The Rand Corporation, Santa Monica, 1991.

[4] Melenk H., Neun W. RR: Parallel Symbolic Algorithm Support for PSL Based REDUCE (DRAFT). Konrad-Zuse-Zentrum f"ur Informationstechnik, Berlin, 1993.

[5] Graebe H.-G. CALI - A REDUCE Package for Commutative Algebra, available at redlib@sc.zib-berlin.de , Leipzig, 1992.

[6] V.P. Gerdt, N.V. Khutornoy, W. Lassner. Computer Detecting of Lie Algebra Isomorphisms, to appear in the proceedings of the International Conference on Programming and Mathematical Methods for Solving Physical Problems, Dubna, June 14-19, 1993.

(Extended version as preprint. 17/93, Naturwissenschaftlich-Theoretisches Zentrum, Universität Leipzig, 1993.)

SCALABLE PARALLEL PROCESSING

Steve Wallach
Convex Computer Corporation, Tx, USA

A detailed description of the Convex SPP-1 will be presented. Both the software and hardware characteristics will be discussed.

APPENDIX

Vectors, Clusters, MPP's - Competition or Coexistence

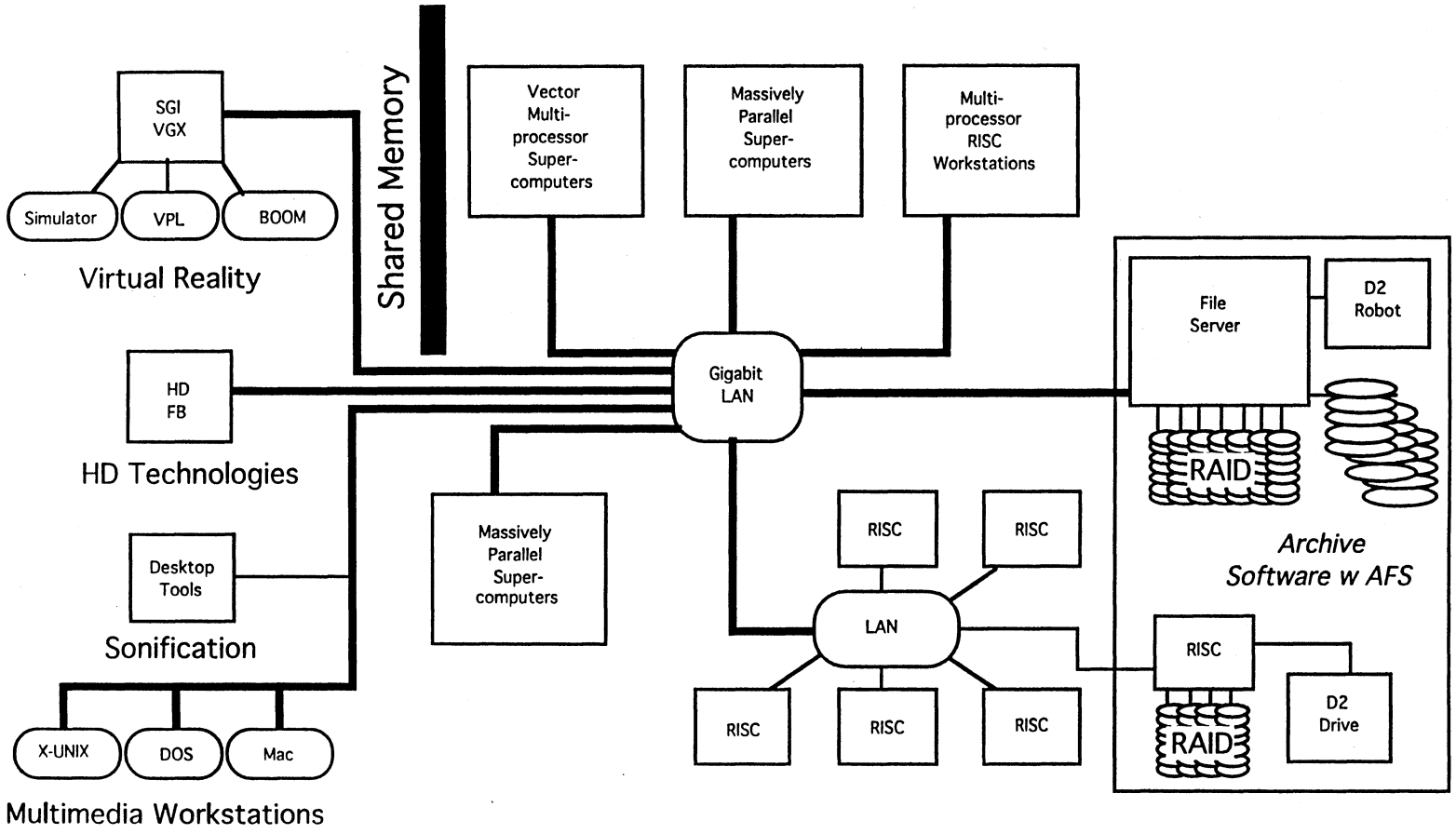
Frank Baetke

The computer industry is experiencing a major paradigm shift which is clearly visible in the trend to RISC-based technology but is much less evident in the software area. Some may conclude that all mainframe computers as well as vector computers in the high performance arena will be history within this decade, squeezed between workstations with steadily growing performance on one and the new promise of massively parallel systems on the other end. Others go even further claiming that more or less everything can be done in future with so called clustered nets of workstations.

Different evaluation criteria and performance characteristics are the base for above arguments. To get a more unbiased picture the underlying architectural and design principles as well as strongpoints and weaknesses of the different system architectures have to be discussed.

A different point of view is to consider the components of a whole network as an entity where different functional units have much more specialized tasks as editing and visualization, high performance computing on shared and distributed memory processors and finally archieving and other file related services.

Larry Smarr's* Vision : Metacomputer

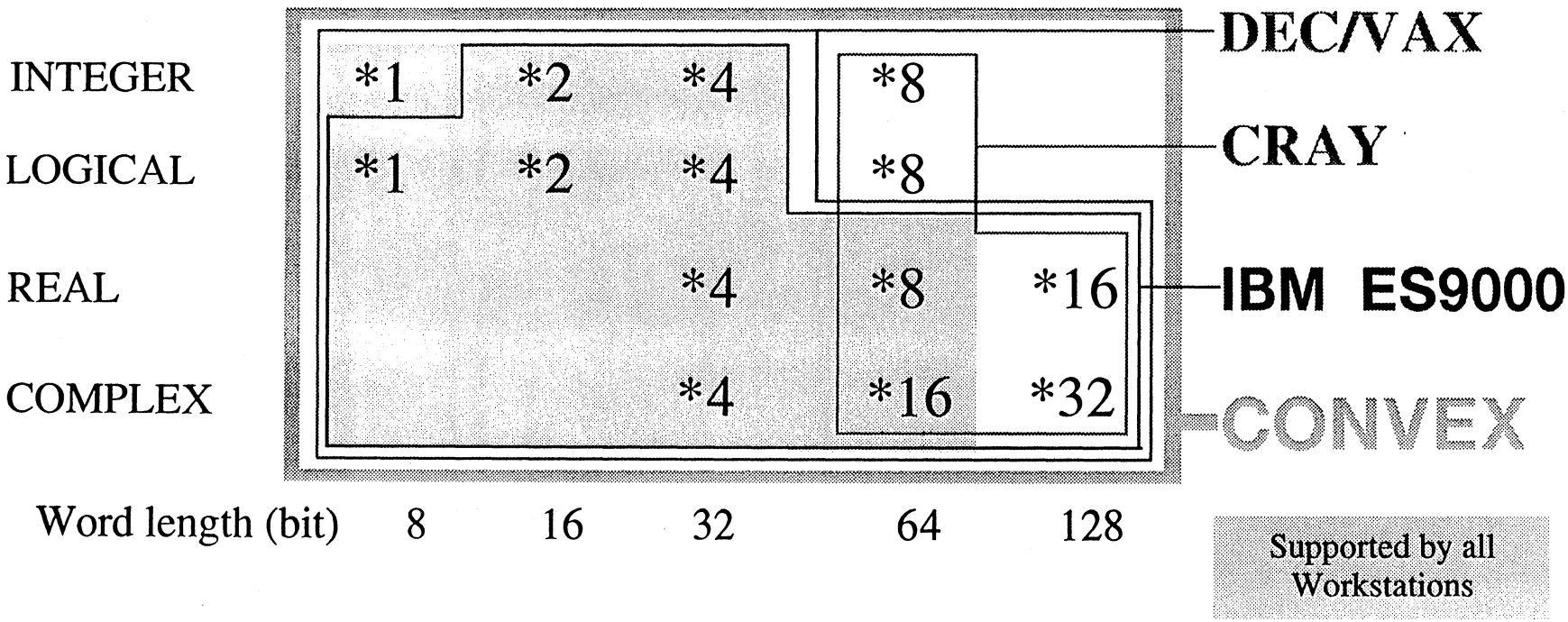


*** Director of NCSA**

MP/FB

The Advantage Of A Modern Design

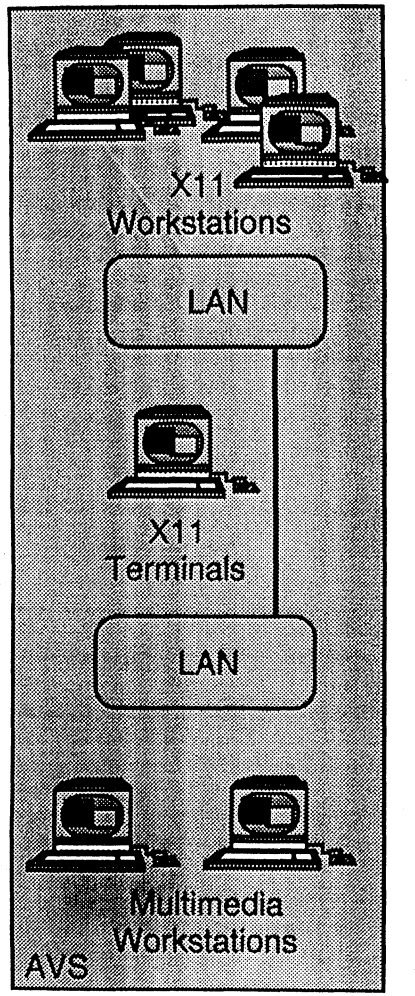
Data Types in C and FORTRAN



Only CONVEX has vector instructions for: All LOGICAL data types
All INTEGER data types
REAL data types 32 bit and 64 bit (full IEEE)

CONVEX and HP

Visualization and Editing

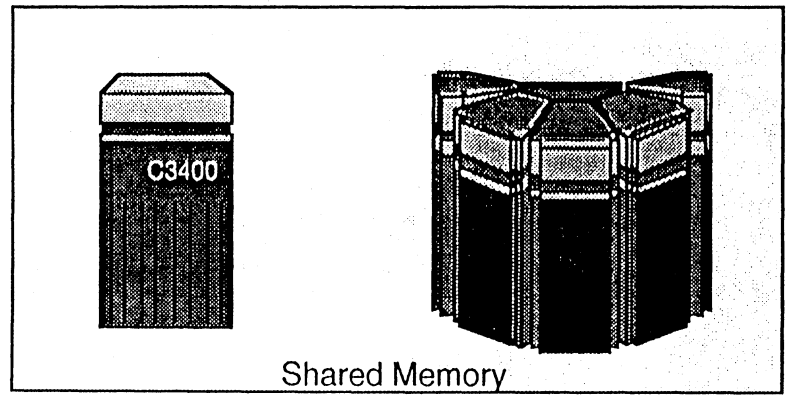


Visualiziation
Peripheral Resources

- Workstations
- X - Terminals
- Peripherals
- Networks

CONVEX

Vector/Parallel Computers

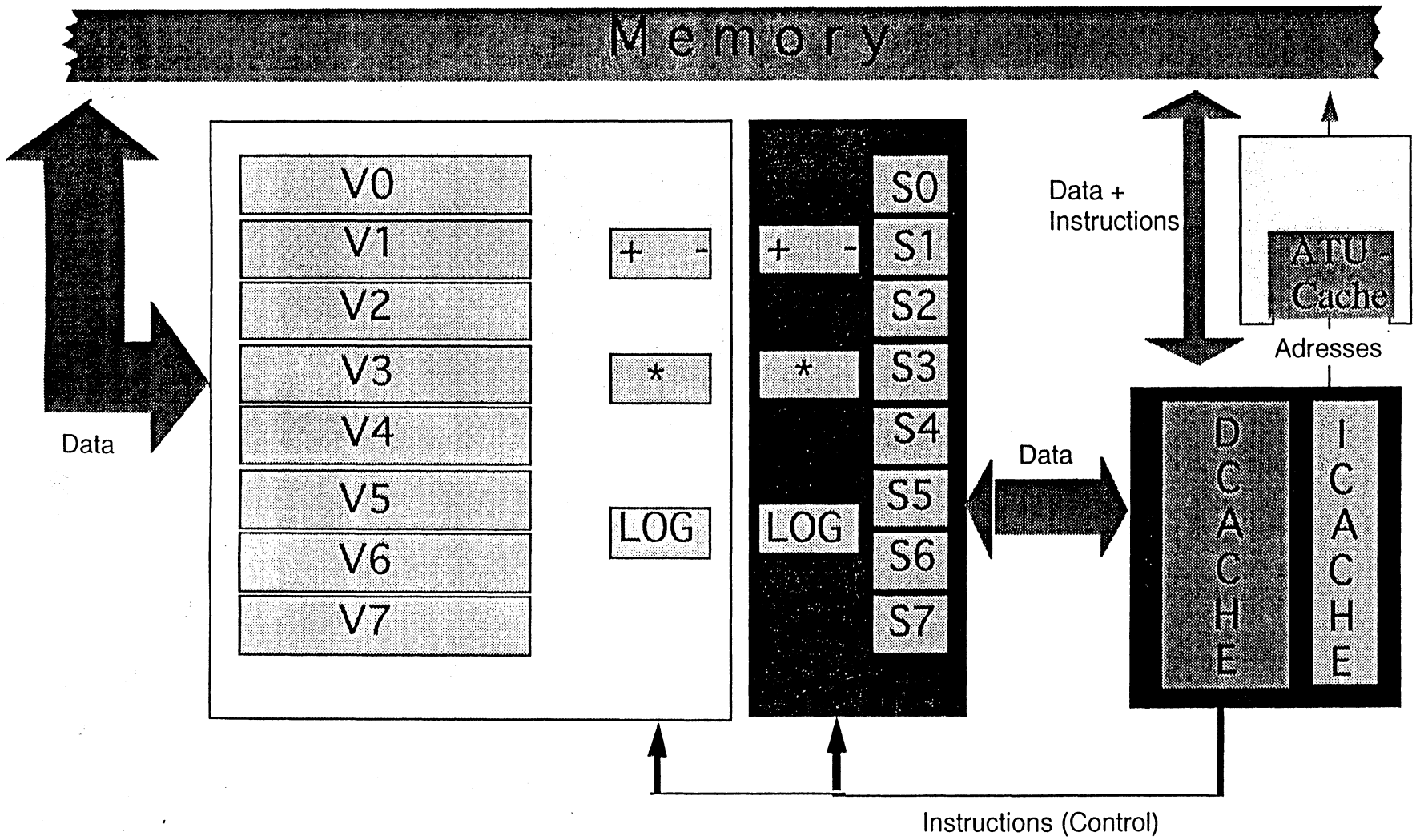


- SMP Vector/Parallel systems - fully aircooled
- 1 - 8 CPU's based on GaAS/BiCMOS/ECL
- Up to 2 GFlops performance
- Up to 4 GBytes main memory
- Full IEEE compatibility provided in hardware
- Byte oriented addressing
- Virtual memory management and demand paging
- More than 1100 installations (Record !)

MFLOPS of the GRB poisson solver

		orginal version			vectorized version		
Nx	Ny	C3810	HP		C3810	HP	
			730	735		730	735
17	17	12,9	10,9	14,2	42,0	21,6	53,3
33	33	24,6	11,6	13,3	88,5	27,8	72,9
41	41	28,7	11,3	17,7	113,2	29,1	77,4
65	65	41,2	10,4	16,9	135,8	30,5	79,6
129	129	62,7	*11,2*	*16,2*	157,0	*26,1*	*58,0*
257	257	77,1	*8,6*	*11,3*	162,7	*14,5*	*22,7*
513	513	81,4			164,5		
1025	1025	83,8			165,1		
2049	2049	85,4			165,0		

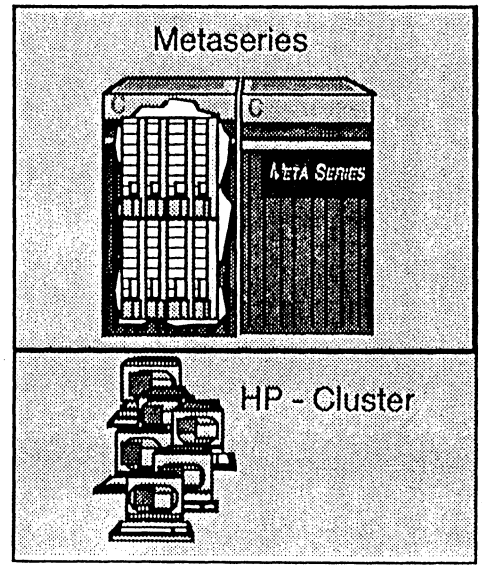
Vectors added to RISC-Architecture



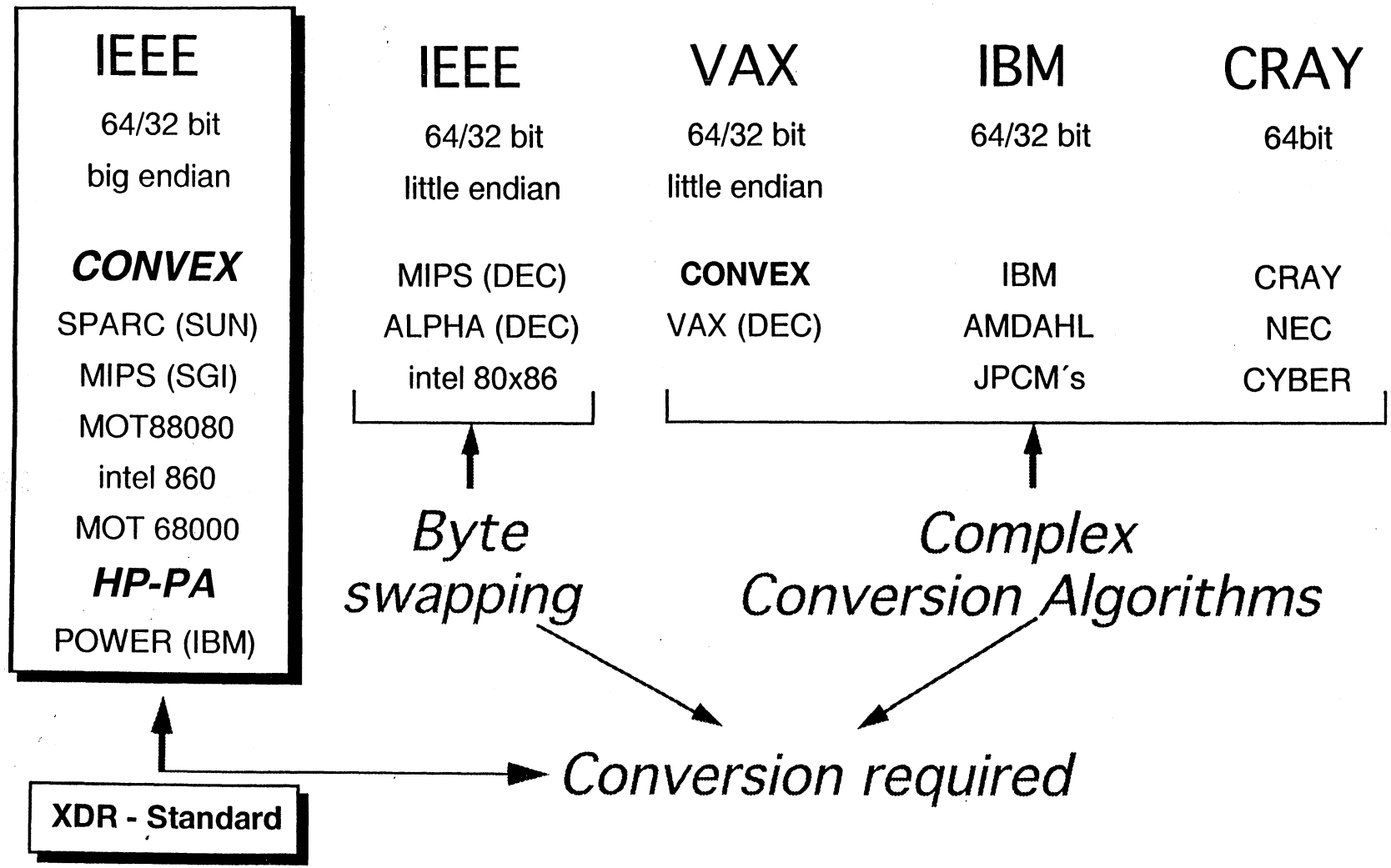
CONVEX and HP

Distributed Memory Systems (Clusters)

- Loosely Coupled Parallel System (CONVEX + HP 700)
- Meta Series includes all C-Series benefits
- Tuned Software: NQS+, PVM, MLIB, ConvexOS
- EMP Programming Model (Explicit Message Passing)



Floating-Point Formats

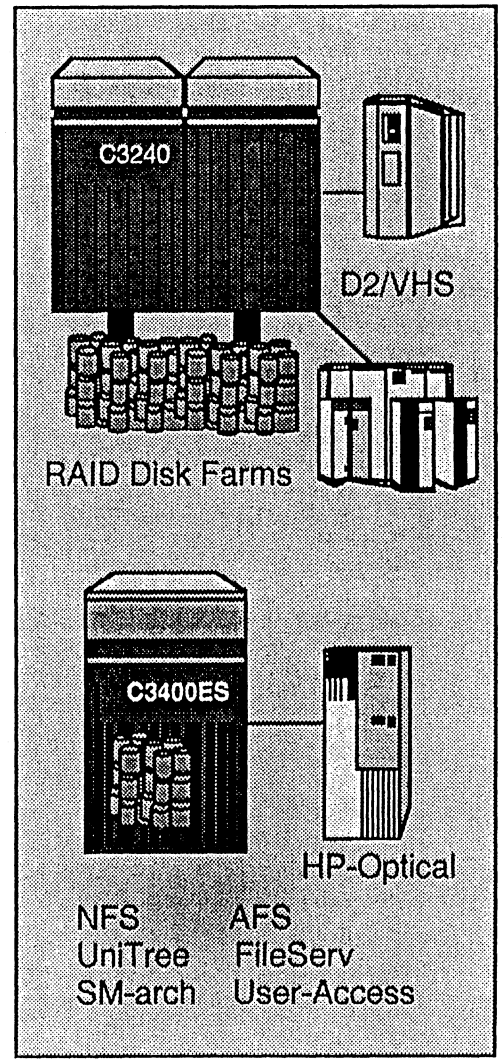


CONVEX and HP

The Complete Picture

"Convex is unique amongst providers of mass storage solutions in that it has embraced all three mass storage application types (Archieving, File Serving, File Migration) with a unique solution for each type."

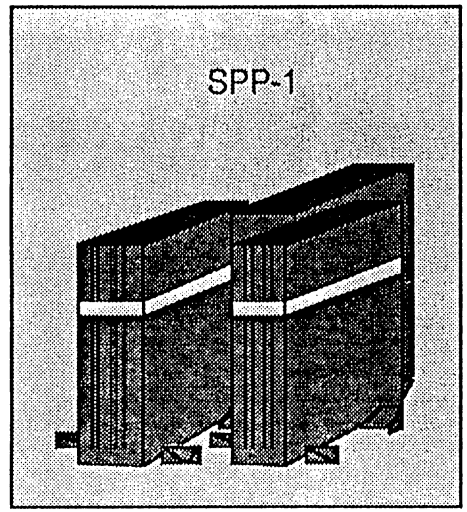
Dr. Sanjay Ranade
Focus Report on Mass Storage

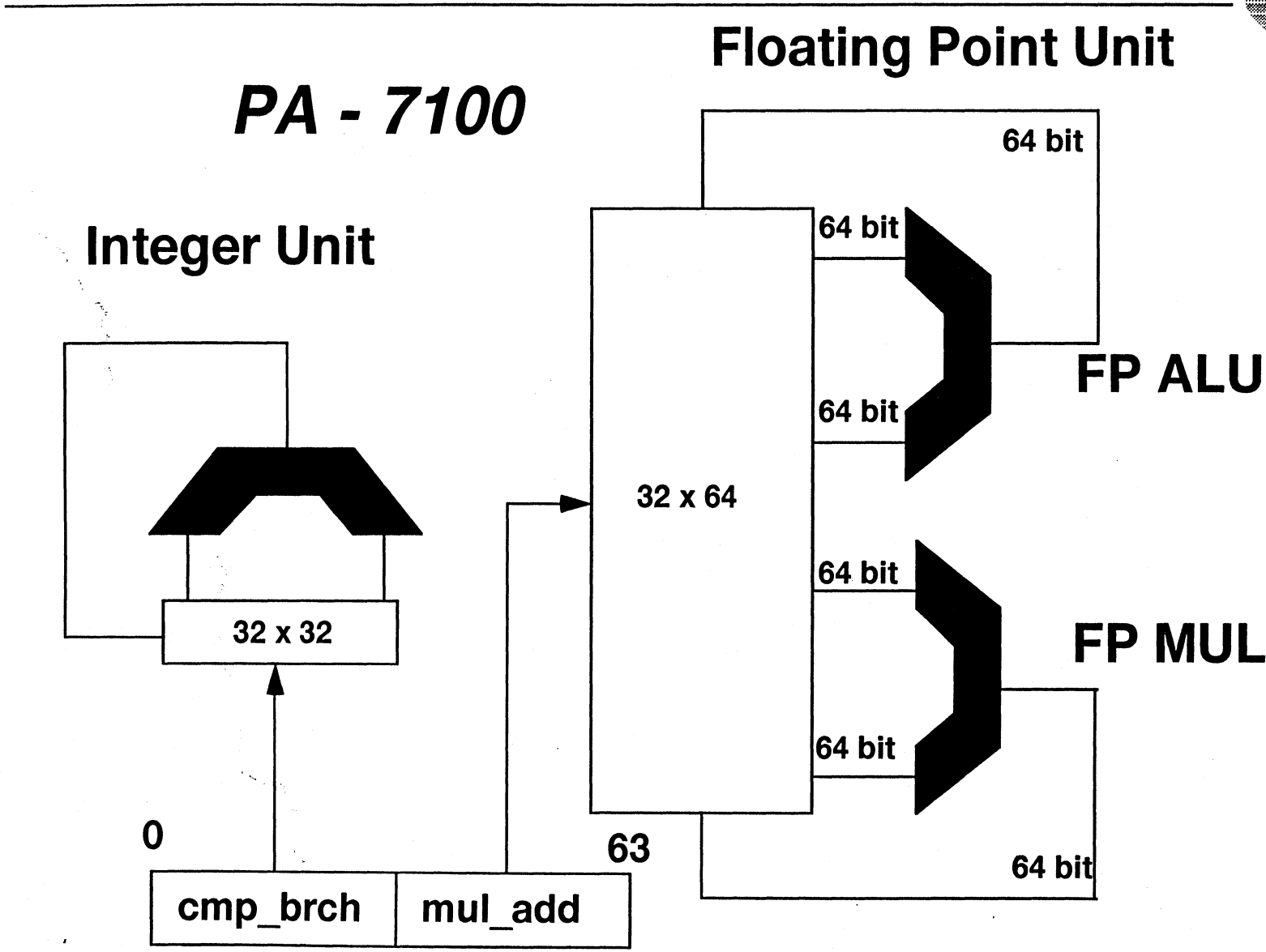


CONVEX

The CONVEX MPP: SPP-1

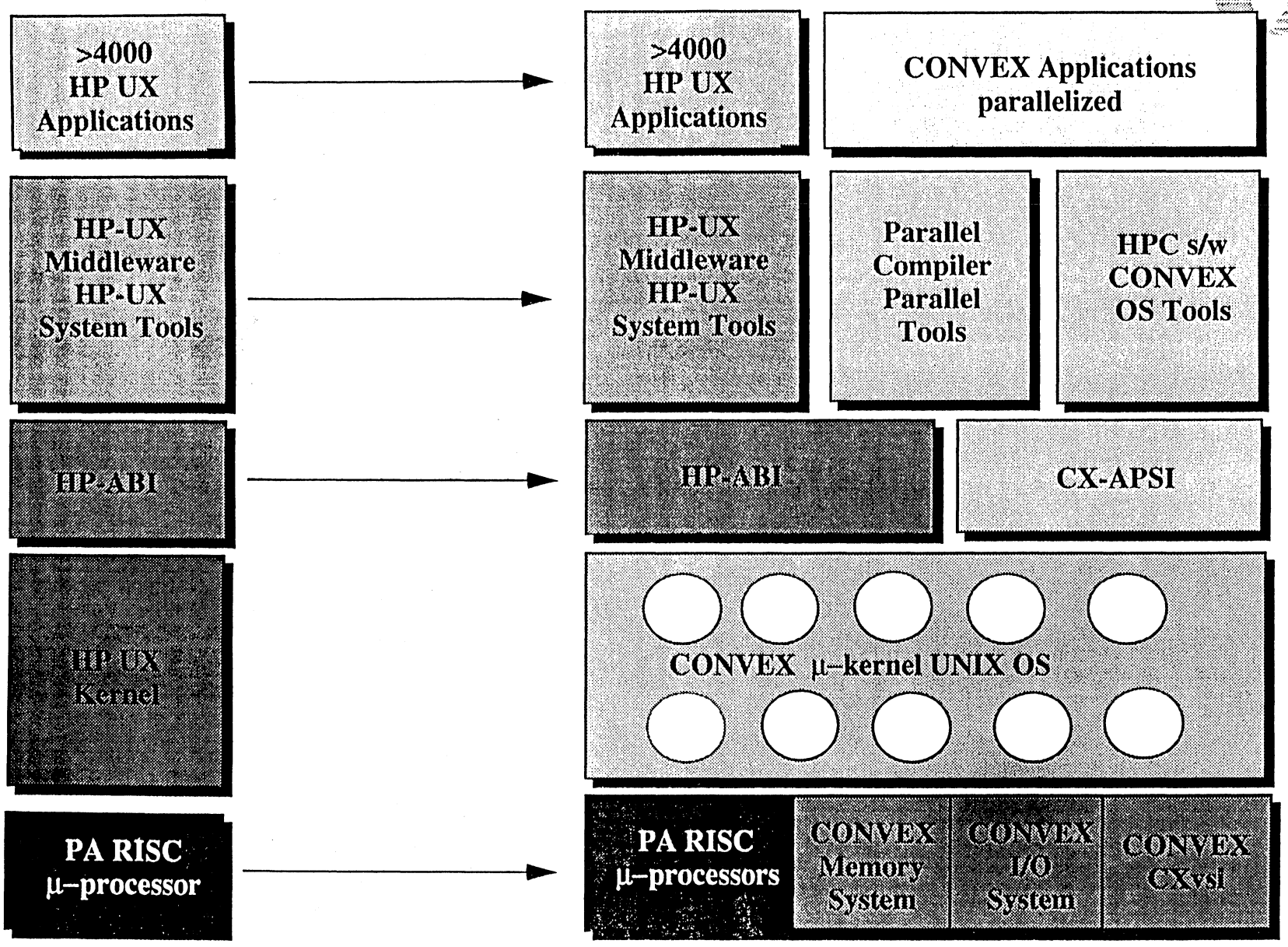
- Based on HP-PA RISC Technology (7100 CPU)
- Architecturally Scalable up to 3 Teraflops
- Standalone Multiuser Operation
- ConvexOS Features and HP-UX Compatibility
- Automatic Parallelization with CONVEX APC Technology
- VSM Programming Model (Virtual Shared Memory)
- EMP Programming Model (Explicit Message Passing)




PA - 7100
Floating Point Unit
Integer Unit
64 bit
64 bit
64 bit
FP ALU
32 x 64
32 x 32
64 bit
64 bit
FP MUL
0
63
64 bit
cmp_brch
mul_add

HP - Components

CONVEX SPP-Components



MPP Programming Model



One model with two styles

Data parallel

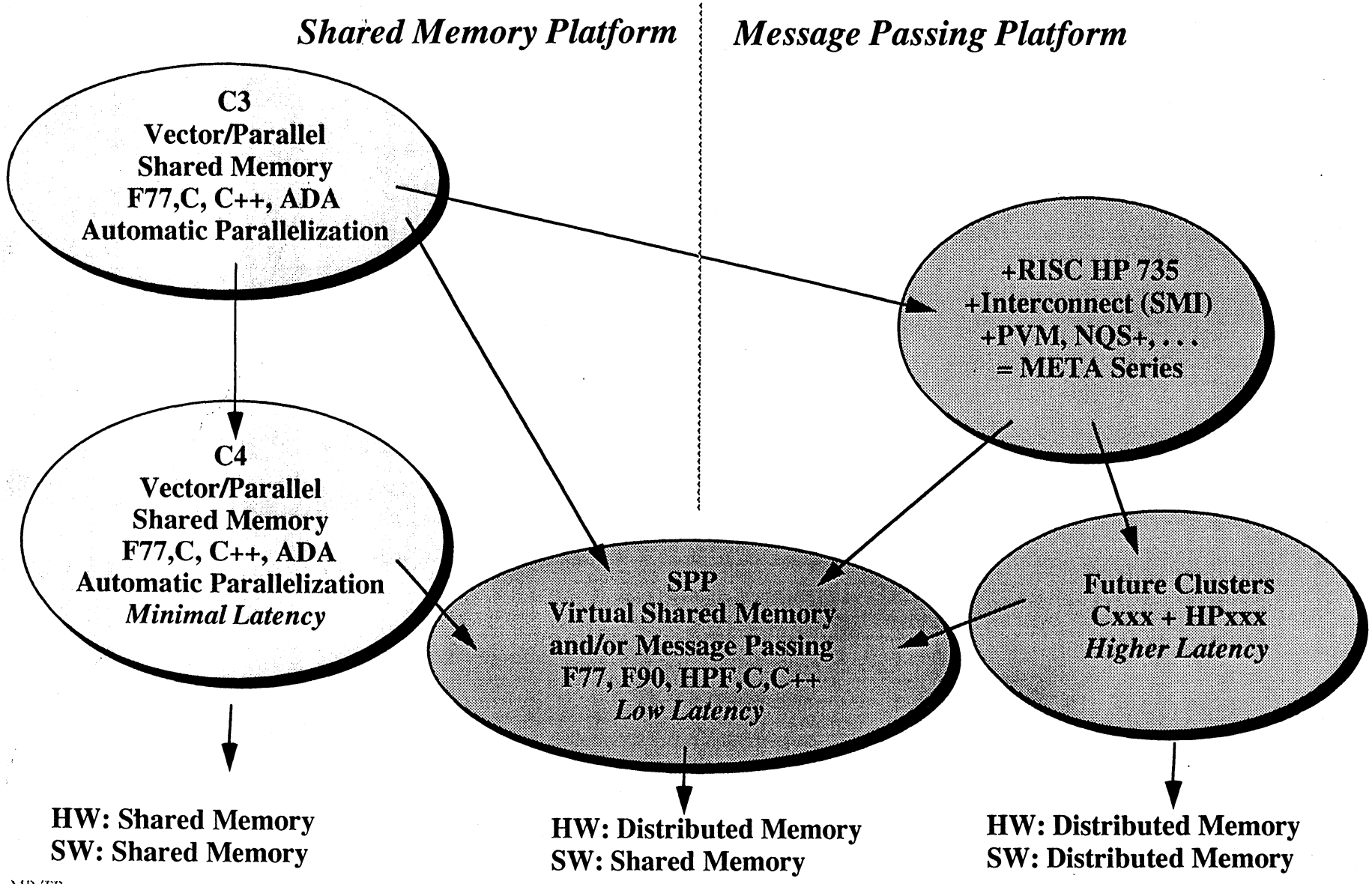
- single multithreaded process (shared memory model)
- each thread has 4 Gbyte virtual address space
- virtual space can be all shared, partially shared or all private

Message passing

- multiple processes
- each process has unique 4 Gbyte address space

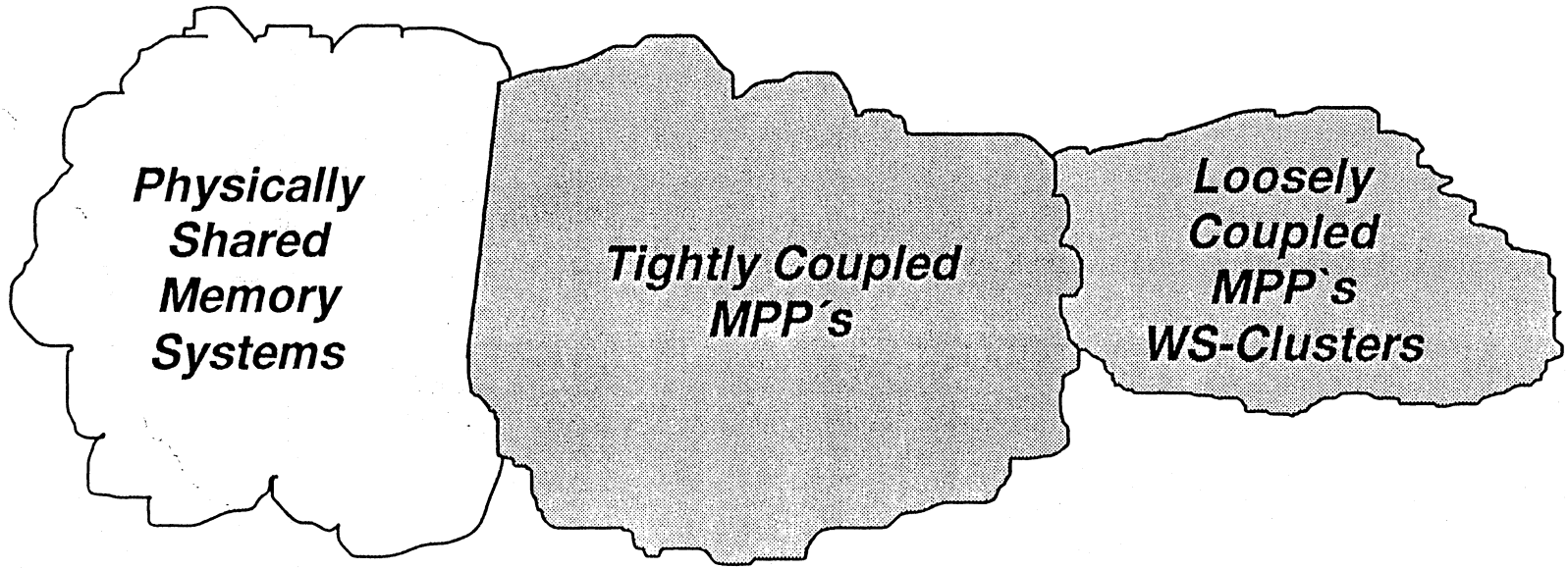
A hybrid of data parallel and message passing is possible

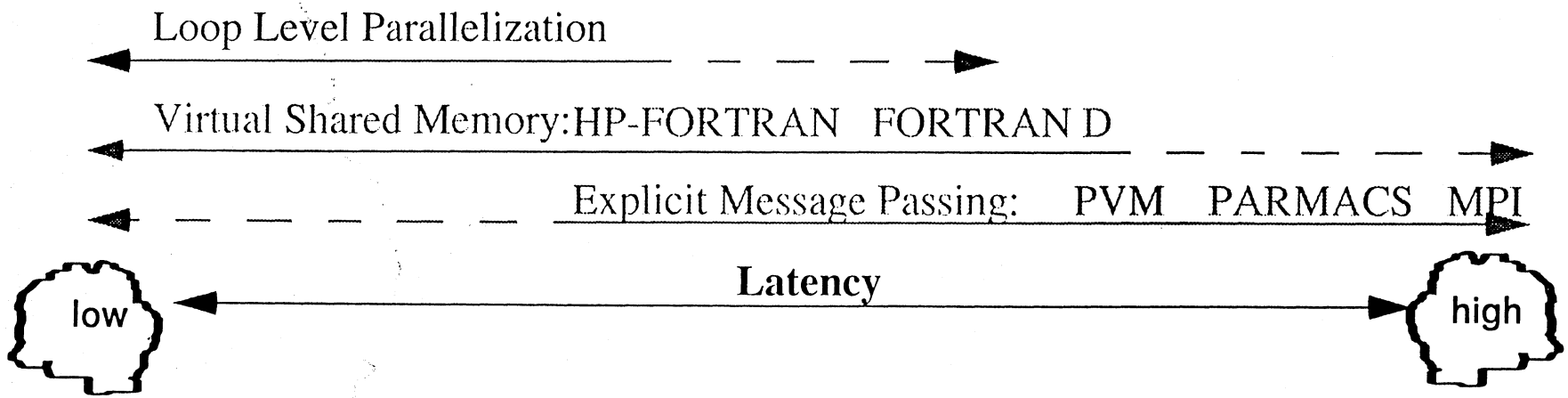
Platforms for Programmers/Applications



MP/FB

Parallel Potentials

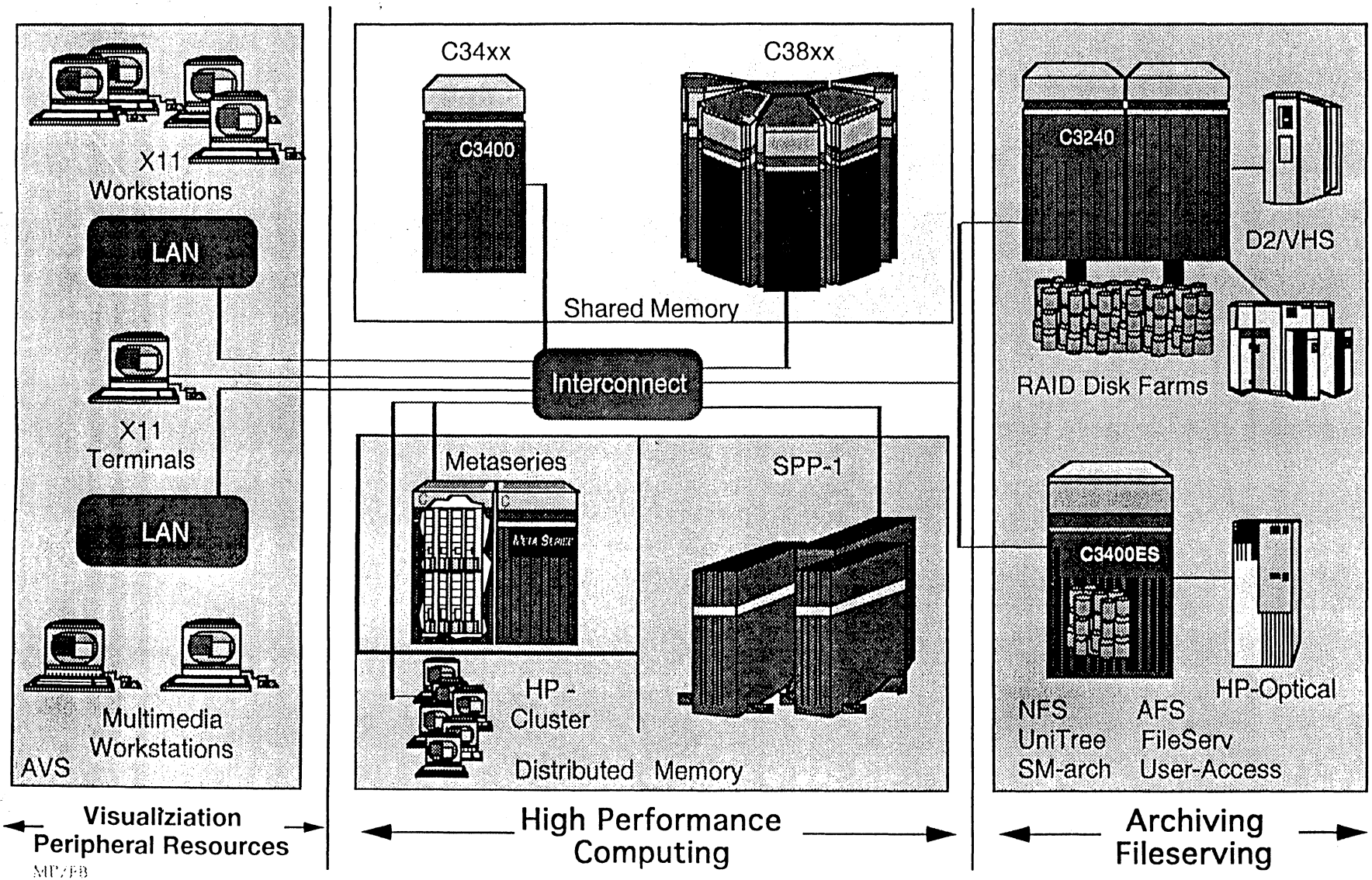




MP/FB

CONVEX and HP

The Complete Picture





LABEIN
TECHNOLOGICAL RESEARCH CENTRE

Bizkaiko Foru Aldundia
Ekonomi Sustapen Saila

Diputación Foral de Bizkaia
Departamento de Promoción Económica