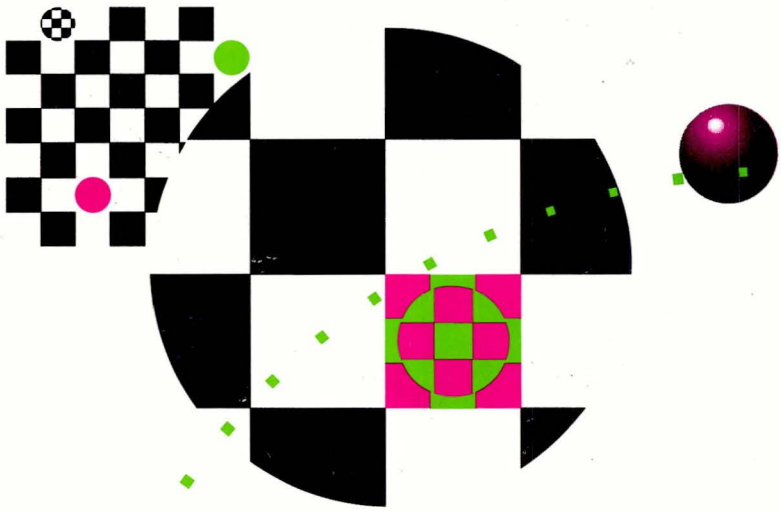


CONVEX

CXpa Reference

Second Edition

CONVEX Computer Corporation
3000 Waterview Parkway
P.O. Box 833851
Richardson, TX 75083-3851
United States of America
(214)497-4000

CXpa Reference

Order No. DSW-253

Second Edition

CONVEX Press
Richardson, Texas
United States of America

CXpa Reference

Order No. DSW-253

Copyright ©1994 CONVEX Computer Corporation
All rights reserved.

This document is copyrighted. This document may not, in whole or part, be copied, duplicated, reproduced, translated, electronically stored, or reduced to machine readable form without prior written consent from CONVEX Computer Corporation.

Although the material contained herein has been carefully reviewed, CONVEX Computer Corporation does not warrant it to be free of errors or omissions. CONVEX reserves the right to make corrections, updates, revisions or changes to the information contained herein. CONVEX does not warrant the material described herein to be free of patent infringement.

UNLESS PROVIDED OTHERWISE IN WRITING WITH CONVEX COMPUTER CORPORATION (CONVEX), THE PROGRAM DESCRIBED HEREIN IS PROVIDED AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. SOME STATES DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES. THE ABOVE EXCLUSION MAY NOT BE APPLICABLE TO ALL PURCHASERS BECAUSE WARRANTY RIGHTS CAN VARY FROM STATE TO STATE. IN NO EVENT WILL CONVEX BE LIABLE TO ANYONE FOR SPECIAL, COLLATERAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, INCLUDING ANY LOST PROFITS OR LOST SAVINGS, ARISING OUT OF THE USE OR INABILITY TO USE THIS PROGRAM. CONVEX WILL NOT BE LIABLE EVEN IF IT HAS BEEN NOTIFIED OF THE POSSIBILITY OF SUCH DAMAGE BY THE PURCHASER OR ANY THIRD PARTY.

CONVEX, CONVEX CXpa, and the CONVEX logo ("C") are registered trademarks of CONVEX Computer Corporation.
PostScript is a trademark of Adobe Systems Inc.
UNIX is a trademark of UNIX System Laboratories, Inc., a wholly owned subsidiary of Novell, Inc.
X Window System is a trademark of the Massachusetts Institute of Technology.

Printed in the United States of America

Revision information for

CXpa Reference

Edition	Document No.	Description
Second	710-004730-005	Released December 1994 with V3.1 of CONVEX CXpa software. This document applies to C Series and Exemplar (SPP Series) systems.
First	710-004730-004	Initial release, March 1993. Released with V2.0 of CONVEX CXpa software.

Contents

Using this book xi
Purpose and audience xi
Organization xi
Notational conventions xii
Command syntax xii
General conventions xii
Notes xiii
Architecture dependencies xiii
Associated documents xiv
Ordering documentation xiv
Technical assistance xiv

Part 1 Using CXpa

1 Introduction 3
What is a profiler? 3
Why use a profiler? 4
Steps for learning CXpa 5
Overview of CXpa 7
Available metrics 7
SPP Series metrics 7
C4 Series events 8
CXpa interfaces 8
X window mode 8
Line mode 9
Batch mode 9
Graphic analysis of performance data 9
Performance reports 10
CXpa limitations 11
Learning CXpa quickly 15
Using CXpa in X window mode 15
Using CXpa in line mode 20
Editing the command line 22
Setting the PDF 25
Changing the PDF name in X window mode . . . 25
Changing the PDF name in line or batch mode . . 26

Analyzing PDFs only 29
Analyzing PDFs in X window mode 29
Opening other PDFs 30
Choosing an active PDF 30
Analyzing PDFs in line mode 30
Opening other PDFs 30
Using batch mode 31
Using batch mode from the command line 31
Command file input using the -x option 31
Argument input using the -e option 32
Using batch mode from a script 33
Using online help 35
Using the buttons and scroll bars 36
Using the menus 37
Selecting a topic 38
Searching for a topic 38
Exiting from help 39

2 Compiling for profiling 41
Compiling for CXpa 43
Compiling and linking in one step 45
Compiling and linking separately 45
Advanced compiling 47
Advanced compiling 49
Profiling routines that call uninstrumented routines . 49
Using instrumented libraries with `-cxpalib` . . . 50
Using the `-cxpamon` compiler option 50
Problems with compiling routines with both `-cxpa` and `-cxpab` 50
Problems with not compiling parallel regions for use with CXpa 50
Problems using the Fortran `OPTIONS` statement with CXpa 50

3 Choosing performance data to collect . . 51
Introducing source code regions 53
Source code annotations for regions 54
Selecting source code regions 55
Selecting regions in X window mode 57
Selecting types of regions to profile in all routines . 58
Selecting types of regions to profile in specific routines 59
Selecting all regions in your program for profiling . 61
Selecting and deselecting regions in line mode 63
Selecting or deselecting one type of region in all routines 64

Selecting or deselecting one type of region in specific routines 65
Selecting or deselecting one type of region at specific lines 67
Selecting or deselecting all regions in specific routines 68
Selecting or deselecting all regions at specific lines . 70
Selecting or deselecting all regions in all routines . . 71
Introducing metrics 73
Metrics 73
All architectures 73
C4 and SPP Series 74
SPP Series events 75
Definitions 75
Events 76
C4 Series events 76
Selecting metrics in X window mode 79
Selecting events (C4 and SPP Series only) 81
Selecting metrics in line mode 85
Choosing metrics to collect at the command line . . 85
Event types 86
SPP Series events 86
C4 Series events 87

Part 2 Reference pages

4 Reports 91
Filtering report data 92
X window mode 92
Line mode 92
Viewing reports 93
X window mode 93
Line mode 93
Report header 93
Basic Block report 95
Loop reports 99
Computation 100
Time to Solution 101
Events (C4 and SPP Series only) 103
Vector Register Utilization (C Series only) . . . 105
Interpreting Mflops data 111
Vector Register Utilization report 111
Vector chaining 113
Vector chaining and functional units 113
Functional unit reservation 117
C2 and C3 Series vector functional units 118

C4 Series vector functional units 119
Redundant operations and vector functional units 120
Register bank conflicts 120
C2 and C3 Series register banks 120
C4 Series register banks 120
Chimes 121
Calculating chimes 121
Vector Mflops 123
Calculating estimated Mflops 123
Calculating peak Mflops 125
Parallel Region reports 127
Time to Solution 127
Events (C4 and SPP Series only) 129
Routine reports 135
Computation 136
Time to solution 136
Events (C4 and SPP Series only) 137

5 Windows 143
2D Profile window 145
Source code correlation 146
3D Profile window 151
Source code correlation 152
Graph rotation 152
About CXpa 157
Analysis Control window 159
Invoking CXpa with a PDF only 159
Choosing a current PDF 160
Opening PDFs 160
Analysis Report window 163
Customize Report 167
Executable Manager window 169
Profiling a program 170
Filter Profile 175
Filter Report 177
Help window 179
Info Executable 183
Info PDF 185
Info Session 187
New PDF 189
Filtering files 190
Selecting a file 190
Open PDF 193
Filtering files 194
Selecting a file 194
Profile Customization 197

Profile Selection 199
Selecting regions and metrics for profiling in all routines 200
Selecting regions and metrics in individual routines . 201
Save Profile 205
Filtering files 206
Selecting a file 206
Save Report 209
Filtering files 209
Selecting a file 210
Scale 213
X-Axis scaling 213
Y-Axis scaling 214
Scheduling Selection 217
Sort 219
Source Code Selection 221
Changing source files 221
Source Code window 223
Annotations 224
Changing source files 224
Source Search Path 227
Adding a directory to CXpa's search path 227
Removing a directory from CXpa's search path . . 228
Subcomplex Selection dialog 229
Visibility Selection 231

6 Commands 233

`add path` 235
`analyze` 237
`collect` 241
`continue` 245
`cxpa` 247
Using the X windows version of CXpa 249
Using the line mode version of CXpa 250
Using the CXPA environment variable to specify options 250
`deselect` 255
`help` 259
`info` 263
Executable information 263
PDF information 263
File and directory information 264
`list` 265
`list selectable` 269
`path` 273
`quit` 275
`rerun` 277

run 281
select 285
set events 289
set pdf 293
set subcomplex 295
set visibility 297
source 299
stop 301
version 303

7 Messages 305

Index 323

Using this book

Purpose and audience

The *CXpa Reference* is both a user's guide and a reference. This book introduces new users to CXpa and describes the windows, dialog boxes, and commands that appear in the CONVEX CXpa software.

This manual is also available through the CXpa online help system.

Organization

This manual is organized as follows:

- **Part I, "Using CXpa"**—Contains three chapters that introduce new users to CXpa.
  - **Chapter 1, "Introduction"**—Introduces CXpa and profiling to new users.
  - **Chapter 2, "Compiling for profiling"**—Contains the information needed to compile programs for use with CXpa.
  - **Chapter 3, "Choosing performance data to collect"**—Introduces and describes the types of metrics that can be collected on each architecture and how to select source code regions and metrics for profiling.
- **Part II, "Reference pages"**—Contains three chapters.
  - **Chapter 4, "Reports"**—Contains detailed descriptions of the textual performance reports that CXpa creates.
  - **Chapter 5, "Windows"**—Contains descriptions and explanations for each window and dialog used in CXpa's X window interface.
  - **Chapter 6, "Commands"**—Contains descriptions, syntax rules, and examples for all CXpa commands.
  - **Chapter 7, "Messages"**—Lists and explains CXpa messages.
- **Index**

Notational conventions

This document uses the following notational conventions.

Command syntax

Consider this example:

```
(CXpa) command <param1> [, ...] {a | b} [<param2>]
  ①      ②       ③       ④     ⑤      ⑥
```

1. `(CXpa)` is the CXpa command prompt.
2. **`command`** must be typed as it appears.
3. *<param1>* indicates a parameter that must be supplied.
4. The horizontal ellipsis in brackets [, ...] indicates that additional parameters may be specified.
5. Either **`a`** or **`b`** must be specified.
6. [*<param2>*] indicates an optional parameter.

General conventions

- **`Bold constant-width font`** identifies user input in examples.
- *Italics*:
  - Designate user-supplied variables in a command-line example (when enclosed in <>)
  - Indicate document titles
- `Constant-width font` designates input and output, including:
  - Command names and options
  - System calls
  - Program statements, command output, and error messages returned
- Horizontal ellipsis (...) shows repetition of the preceding item(s).
- Vertical ellipsis shows that lines have been left out of an example.

- Words and abbreviations that indicate keyboard keys you press are identified in a distinctive bold type. For example, **RETURN** refers to the carriage return key. Words separated by a hyphen indicate two keys that you press simultaneously. For example, **CTRL-X** indicates that you must press and hold down the **CTRL** key and then press the **X** key.
- The shell prompt is shown as a percent sign (%).
- Unless otherwise indicated, source code examples are in Fortran.

Notes

NOTE: A NOTE highlights information that may be of particular interest regarding the software or your files.

Architecture dependencies

The architecture of the computer system can affect some aspects of CXpa and the program you are profiling. These architecture dependencies are indicated in several ways throughout this book.

If the entire reference topic applies to a particular architecture, it is marked by a symbol similar to the following:

The above symbol means that the reference topic applies to SPP Series machines only.

In other cases, architecture dependencies are indicated by the title of a section or by a notation in parentheses.

Associated documents

For more information, refer to:

- *Exemplar Programming Guide* (DSW-067) — Describes efficient methods for programming in Convex C and Fortran on Exemplar (also known as SPP Series) computers. Topics covered include the SPP Series programming model, automatic optimizations, basic and advanced manual optimizations, and the Convex Parallel Support Library (CPSlib).
- *Exemplar Architecture* (DHW-014) — Describes the Convex implementation of scalable parallel processing on Exemplar (SPP Series) machines.
- *C Optimization Guide* (DSW-089) — Describes the types of optimizations available in Convex C and shows you how to use optimization directives and options.
- *Fortran Optimization Guide* (DSW-034) — Describes the types of optimizations available in Convex Fortran and shows you how to use optimization directives and options.

Ordering documentation

To order the current edition of this or any other CONVEX document, send requests to:

CONVEX Computer Corporation
Customer Service
P.O. Box 833851
Richardson, TX 75083-3851 USA

Include the order number or the exact title.

In some cases, you might not want the latest edition. To order a specific edition of a document, contact your local CONVEX office or call theTechnical Assistance Center (TAC).

Technical assistance

If you have questions that are not answered by the documentation, contact the CONVEX Technical Assistance Center (TAC). To contact the TAC, use one of the following phone numbers:

- Within the continental U.S., call 1(800)952-0379.
- From Canada, call 1(800)345-2384.
- Outside the continental U.S., contact the local CONVEX office.

The `contact` utility

The TAC recommends using the `contact` utility to report a hardware, software, or documentation problem. The `contact` utility is an interactive program that helps the TAC track reports and route them to the CONVEX personnel most qualified to fix a problem.

After you invoke `contact`, it prompts you for information about the problem. When you finish your report, `contact` mails it to the TAC electronically. The TAC notifies you within 48 hours that your report has been received.

Using `contact` requires:

- UNIX-to-UNIX Communication Protocol (UUCP) connection to the TAC
- Full path name of the program or utility in question
- Version number of the program or utility in question

Refer to the contact(1) man page for complete details.

Part 1 Using CXpa

Part 1 of this book introduces new users to CXpa and contains the following chapters and subjects:

1. Introduction
   - What is a profiler?
   - Why use a profiler?
   - Steps for learning CXpa
   - Overview of CXpa
   - CXpa limitations
   - Learning CXpa quickly
   - Setting the PDF (performance data file)
   - Analyzing PDFs only
   - Using batch execution
   - Using online help
2. Compiling for profiling with CXpa
   - Compiling for CXpa
   - Advanced compiling
3. Choosing performance data to collect
   - Introducing source code regions
   - Selecting regions in X window mode
   - Selecting and deselecting regions in line mode
   - Introducing metrics
   - Selecting metrics in X window mode
   - Selecting metrics in line mode

You can also access this information through the CXpa online help system.

Introduction 1

This chapter introduces general profiling concepts and the CONVEX performance analyzer, CXpa. If you are new to profiling, read this chapter before using CXpa.

The topics discussed in this chapter include:

- What is a profiler?
- Why use a profiler?
- Steps for learning CXpa
- Overview of CXpa
- CXpa limitations
- Learning CXpa quickly
- Setting the PDF (Performance Data File)
- Analyzing PDFs only
- Using batch execution
- Using online help

After reading this chapter, try using CXpa on one of your own programs while reading "Learning CXpa quickly."

What is a profiler?

A profiler is a software tool that measures the runtime performance of programs. Profilers typically measure the total CPU time it takes each portion of your program to execute. A profiler may also measure other things that can help you find performance bottlenecks, such as:

- Loop iteration or routine execution counts
- Wall clock time
- Vector utilization (Vector Mflops)
- Cache miss counts and latency

By running your program under a profiler's control, you can collect performance data that the profiler can display in a report or graph. These reports and graphs enable you to determine the location of performance bottlenecks.

Different profilers collect performance data in different ways. Some profilers sample a program's performance at measured intervals to get an average of a routine's execution time. This is known as statistical sampling. The `prof` utility uses statistical sampling.

CXpa measures a program's entire execution time and reports the total time spent in individual routines, loops, and parallel regions (parallel loops). This produces profiling results that are more accurate than statistical sampling.

Most profilers, including CXpa, require you to compile your program with a special profiling option. This option tells the compiler to create a special executable file that contains information that the profiler uses to collect performance data. When you run this executable file under the profiler's control, the profiler collects performance data and reports the results.

Why use a profiler?

Using the information CXpa provides, you can discover which routines and loops slow down your program the most. In some cases, simple modifications to the source code (for example, inserting compiler directives) can result in significant performance improvements. Without profiling, you might not be able to find performance bottlenecks.

Profiling is an iterative process. By profiling, making changes to your program, and profiling again, you can improve the performance of your program and develop a better understanding of code performance.

Profiling versions of your program that have been compiled at different optimization levels can provide insight into the types of optimizations that work best for given situations.

Steps for learning CXpa

Profiling a program is an iterative process. You profile your program, make changes based on the results, and profile again. This profiling experience, along with an understanding of the Convex compiler's optimizations, is crucial to improving your program's performance.

The following steps suggest ways to gain practical experience using CXpa:

Step 1 Select a small program you have written (or are very familiar with) that takes several seconds to execute and contains some loops.

Step 2 Read "Learning CXpa quickly" later in this chapter. As you read the section, perform the profiling steps on your program. Use the default options to profile routines only until you reach Step 5 below.

The first time you compile your program for CXpa, use the `-no` optimization level.

Step 3 Recompile your program at a higher optimization level and profile it again. Continue doing this until you have found the optimization level that produces the lowest CPU or wall clock time for your program.

Step 4 Identify the routine that takes the longest to execute.

Step 5 Profile the loops in that routine and identify the loop that took the longest to execute.

At this point you have identified the location in your program that has the greatest effect on your program's performance.

Step 6 Try rewriting the loop (or routine) in another way. Refer to the CONVEX optimization guides for C and Fortran or the *Exemplar Programming Guide* (DSW-067) (for SPP Series machines) for more information about optimizing your code.

Step 7 Recompile and profile the program again. Compare the execution time of the loop to the previous time.

You can repeat steps 4 through 6 to continue to identify areas of your program that are taking the most CPU or wall clock time and try to improve their performance.

Overview of CXpa

CXpa is an interactive profiler for programs compiled by the Convex C and FORTRAN compilers. CXpa is available for Convex SPP Series and C Series computers. Using CXpa, you can profile selected parts of your program, control your program's execution, and view performance information in reports or graphs.

CXpa enables you to profile your program in great detail. You can profile routines, loops, or basic blocks. CXpa also profiles optimized loops, including parallel loops created by the compiler.

Available metrics

By default, CXpa measures CPU time and iteration/execution counts for selected regions of your program, but you can also choose from the following list of metrics:

- Wall clock time (all architectures)
- CPU time (all architectures)
- Vector Mflops (C Series machines only)
- Memory access event counts (C4 and SPP Series only)
- Latency time for memory access events; latency is the time it takes to satisfy a memory access request (SPP Series only)

Available memory access event counts and latency metrics differ according to machine type.

SPP Series metrics

On SPP Series machines, cache miss event counts and latency metrics are provided for:

- Local cache misses—The number of times that data not found in the processor cache was found in local memory (memory allocated to that processor's hypernode).
- Remote cache misses—The number of times that data not found in the processor cache was found in remote memory (memory allocated to another hypernode).
- Local and remote cache misses combined

- Event latency time, event latency/CPU time, and event latency/counts metrics for the above events

On SPP Series machines, you can also specify whether memory misses are collected for read operations only or write operations only. The default is to collect events during reads and writes.

C4 Series events

On C4 Series machines, memory access event counts include:

- Data cache misses
- Data cache accesses
- Page table cache misses
- Instruction cache misses
- Vector loads and stores

Latency metrics are not available for C4 Series machines.

Refer to the "Selecting metrics in X window mode," "Selecting metrics in line mode," and "Profile Selection" online help topics or sections in this book for more information.

CXpa interfaces

CXpa has three interfaces: X windows mode, line mode, and batch mode.

X window mode

Use CXpa's X window mode when you want to use a mouse-oriented interface or when you want graphs of performance data. CXpa's window mode has the following features:

- Continual update of elapsed wall time clock
- Mouse selection of the program regions to profile and the metrics to collect
- 2D and 3D graphs of performance data that can display corresponding source code with the click of a mouse button
- Source Code window with source code region annotations
- Analysis Report window for displaying textual performance reports
- Analysis mode that enables you to compare multiple performance data files (PDFs) simultaneously, including PDFs that were created on different architectures

Line mode

Line mode is a character-based interface to CXpa. Use line mode when you are using a CRT terminal or if you prefer a line-oriented interface in an xterm window. Line mode presents performance information in textual reports.

When you start CXpa in line mode with the name of a PDF file only, you can use the `set pdf` and `analyze` commands to compare performance data for multiple performance data files (PDFs), including PDFs that were created on different architectures.

Batch mode

CXpa's batch mode enables you to run CXpa from a shell script. You can invoke CXpa in batch mode by:

- Using the `-x` option to supply a command file to CXpa
- Redirecting input, output, and standard error to and from files

Graphic analysis of performance data

The 2D and 3D Profile windows allow you to visualize the performance data collected when you run your program. You can open multiple 2D and 3D Profile windows to compare and contrast different aspects of your program's performance simultaneously.

You can obtain different views of your program's performance by selectively defining the source code regions and metrics graphed on the X and Y axes. The 2D Profile graphs the data per process, while the 3D Profile graphs the data per thread (for parallel performance).

Other features provided with the 2D and 3D profile graphs include:

- **Source code correlation**—You can click on any bar in the graph to display the source code associated with the code region being graphed.
- **Sorting and scaling options**—The data for both graphs can be sorted alphabetically, by load order, from lowest to highest, or from highest to lowest.

  Scaling options for the 2D profile graph allow you to view or specify the range (X axis) or number/percentage (Y axis) of data values displayed at one time in the 2D Profile window. This can be useful when there is a large number of data items to graph and you want to focus on a subset of the data.
- **Saving profiles**—You can save profile graphs in PostScript, xwd, or ASCII format.

- **3D profile graph rotation**—The 3D profile graph can be rotated by clicking anywhere in the graph with the right mouse button and moving the mouse.

2D and 3D profile graphs are available in X window mode only. Refer to the "2D Profile window" and "3D Profile window" online help topics or sections in this book for more information.

Performance reports

CXpa displays textual performance reports for:

- Routines
- Loops
- Parallel regions (parallel loops)
- Basic blocks

The metrics available in performance reports vary according to machine type, source code regions selected, and the options used when compiling your program. Textual performance reports are available in line mode and X window mode.

Performance analysis reports that can be displayed are as follows:

- **Computation**—Includes iteration/execution counts and CPU time.
- **Time to Solution**—Includes wall clock time and CPU/wall clock time.
- **Events**—Includes memory access events and latency metrics. Events metrics can only be collected on C4 Series and SPP machines; latency metrics are only available on SPP Series machines.
- **Vector Register Utilization**—Includes vector spills, vector flops, chime counts, and estimated Mflops. Vector Mflops can be collected for loop regions on C Series machines only.

For profiled loops (including parallel loops), performance data is provided for both original loops and optimized loops, along with information about the types of optimizations that were performed by the compiler.

For a detailed discussion of each type of report, refer to the "Reports," "Routine reports," "Loop reports," "Parallel Region reports," and "Basic Block report" online help topics or sections in this book.

CXpa limitations

This section lists CXpa's limitations. CXpa can still be used effectively in situations where these limitations occur; however, code changes and/or compiling at a different optimization level may be required.

NOTE: To view current information on CXpa limitations, in X window mode you can look at the online *Release Notice* by selecting the Release Notice option under the Help menu of the Executable Manager or Analysis Control window. In line mode, enter the command `help release` to view an ASCII text version of the *Release Notice*.

The limitations include:

- The profiling of fork()'ed and/or exec()'ed processes is not supported. CXpa will only profile the parent process.
- 3D profiles containing a large number of data points take a long time to render.
- The following constructs cannot be profiled:
  - Loops with no exit point that the compiler can detect. For example:

```
foo() { exit(1); }
.
.
.
while (1) { if (...) foo(); }
```

– Loops optimized into a single vector instruction (-O2, -O3, C Series only). For example, the following report fragment indicates that the loop at line 674 was optimized into a single vector instruction:

```
================================================================================
                          Loop Performance Analysis
================================================================================

Optimized Loops:
                          Times      Iteration Count          CPU Time
 Line    NL Optimization  Exec     Min      Max    Avg (less inner) (plus inner) PS
------- -- ------------ -------- ------ ------ ------ ---------- ---------- --
   674    0 V              N/A    N/A    N/A    N/A        N/A        N/A    u

(u) contains 1 or more uninstrumentable points
```

– Loops whose exits are implemented by the compiler's code generator via branch tables. For example, the UL in the optimization column in the following report fragment indicates that CXpa could not profile a loop region:

```
================================================================================
                          Loop Performance Analysis
================================================================================

Optimized Loops:
                          Times      Iteration Count          CPU Time
 Line    NL Optimization  Exec     Min      Max    Avg (less inner) (plus inner) PS
------- -- ------------ -------- ------ ------ ------ ---------- ---------- --
   804    0 UL             N/A    N/A    N/A    N/A        N/A        N/A    u

(u) contains 1 or more uninstrumentable points
```

- The CPU time spent in signal handlers is profiled; however, this is not reflected in the overall application CPU time (C Series only). For example, the following source code uses a signal handler. The report that follows shows that 0.357 msecs were spent in the signal, but only 0.115 msecs were spent in the total application CPU time.

```
#include <signal.h>

sigfoo() {
   signal(SIGFPE,SIG_DFL);
   printf("caught FPE!\n");
   }

main() {
  int i = 1;
  signal(SIGFPE,sigfoo);
  printf("foo %d\n",i/0);
  }
```

```
================================================================================
Executable      : /toolmaster/devtools/work7/summers/src/cxpa/test/x14114
Profile Data    : /toolmaster/devtools/work7/summers/src/cxpa/test/x14114.pdf
Process State   : terminated
CPU Time        : 0.115 msecs
Wall Clock Time : 0.058  secs
Architecture    : CONVEX C200 Series multiprocessor (2 cpus)
================================================================================
================================================================================
                              Routine Performance Analysis
================================================================================

                   CPU Time
     (less children)       (plus children)       Times
      Total        %         Total        %       Exec     PS    Routine Name
  ------------  -----  ------------  -----  -----------   -- --------------------
        0.011m   9.6%        0.357m 310.4%           1       sigfoo
        0.000    0.0%        0.077m  67.0%           1    tm main
  (t) incomplete, terminated by signal.
  (m) missing time, unable to collect or exceeds reasonable limits.
```

Learning CXpa quickly

This section takes you step-by-step through a profiling session and briefly explains how to use CXpa's standard features in X window and line modes.

Using CXpa in X window mode

To use CXpa in X window mode, follow these steps:

1. Set your DISPLAY environment variable (if it is not already set). For example:

   ```
   setenv DISPLAY mydisplay:0.0
   ```

2. Compile and link your program with:

   `-cxpa` to instrument routines, loops, and parallel regions
   `-cxpar` to instrument routines only
   `-cxpab` to instrument basic blocks only

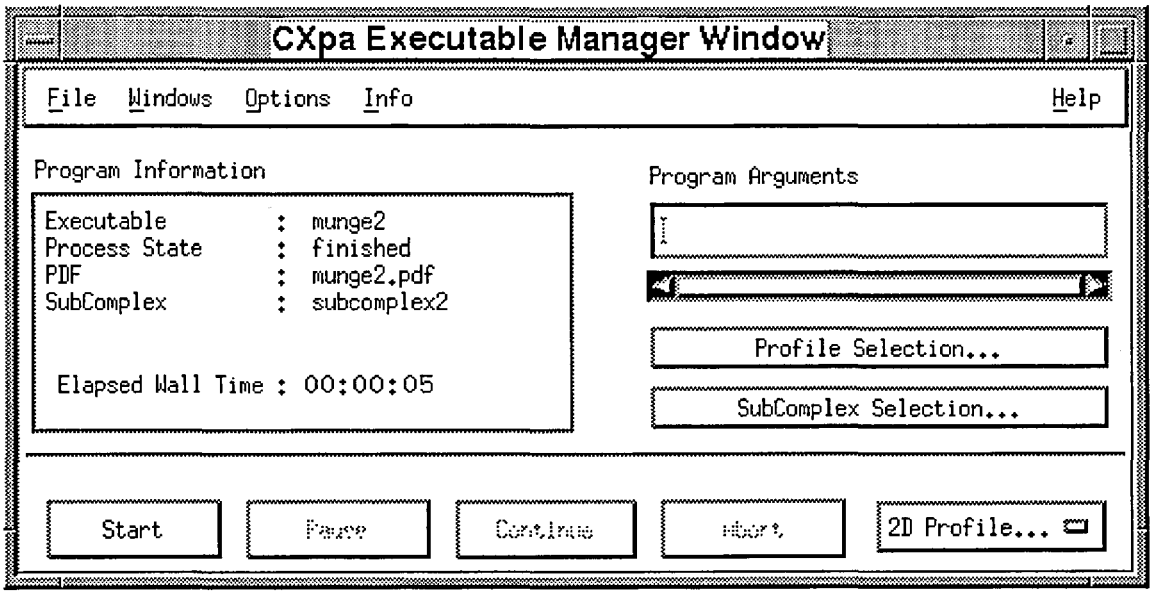
   For example:

   ```
   fc -cxpa prog.f
   ```

3. Invoke CXpa with an executable:

   ```
   cxpa a.out &
   ```

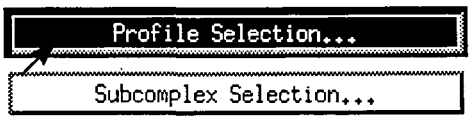
The Executable Manager window appears.



By default, CXpa collects CPU time and execution counts for all routines in your program. To use these defaults, skip to step 5.

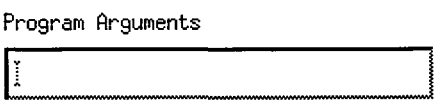
To select different source code regions (such as loops and parallel regions) for profiling and/or collect different metrics, continue with step 4.

4. Click the Profile Selection button; the Profile Selection dialog appears. Select the regions you want to profile and/or the metrics you want to collect. Refer to the "Selecting metrics in X window mode" and "Selecting regions in X window mode" online help topics or sections in this book for more information.

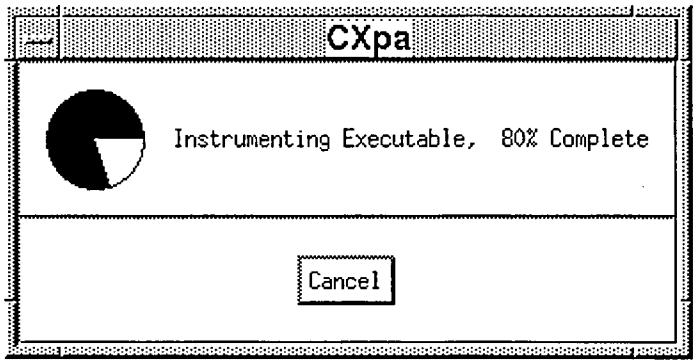


NOTE: Profiling time increases as you select more regions and metrics.

5. Enter any program arguments in the Program Arguments field.



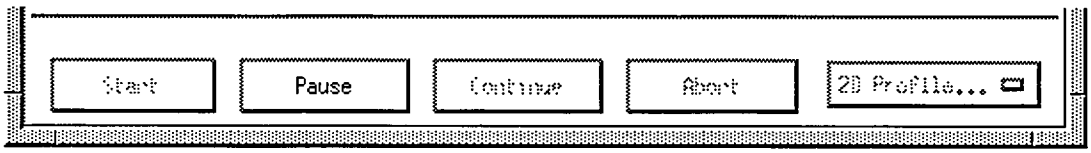
6. Press the Start button. An xterm window appears to display your program's input and output. A dialog also appears while CXpa is instrumenting your executable.



Large programs may take a while to instrument; you can press the Cancel button to stop instrumentation and cancel the run. Once your program is instrumented, your program starts executing, and the dialog disappears.

You can press the Pause button at the bottom of the Executable Manager window to suspend your program's execution during profiling.

NOTE: To obtain the most accurate profiling data, you should not pause your program during profiling. For best results, run the program to completion and then analyze the results. You will also see more accurate results if you run the program in a standalone environment.

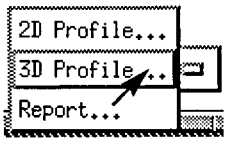


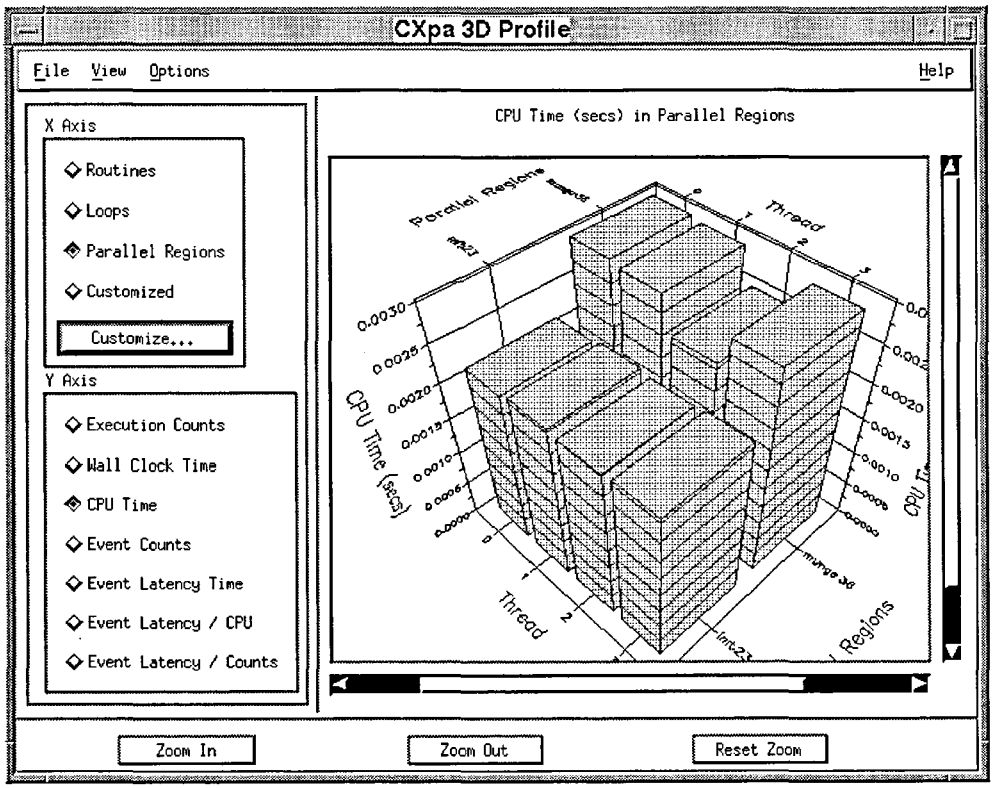
Once your program is paused, you can:

- View a 2D or 3D profile graph or text report composed of intermediate data as shown in step 7
- Abort your program
- Continue your program's execution

7. Display profile results by selecting one of the options from the button at the bottom-right corner or from the Windows menu.

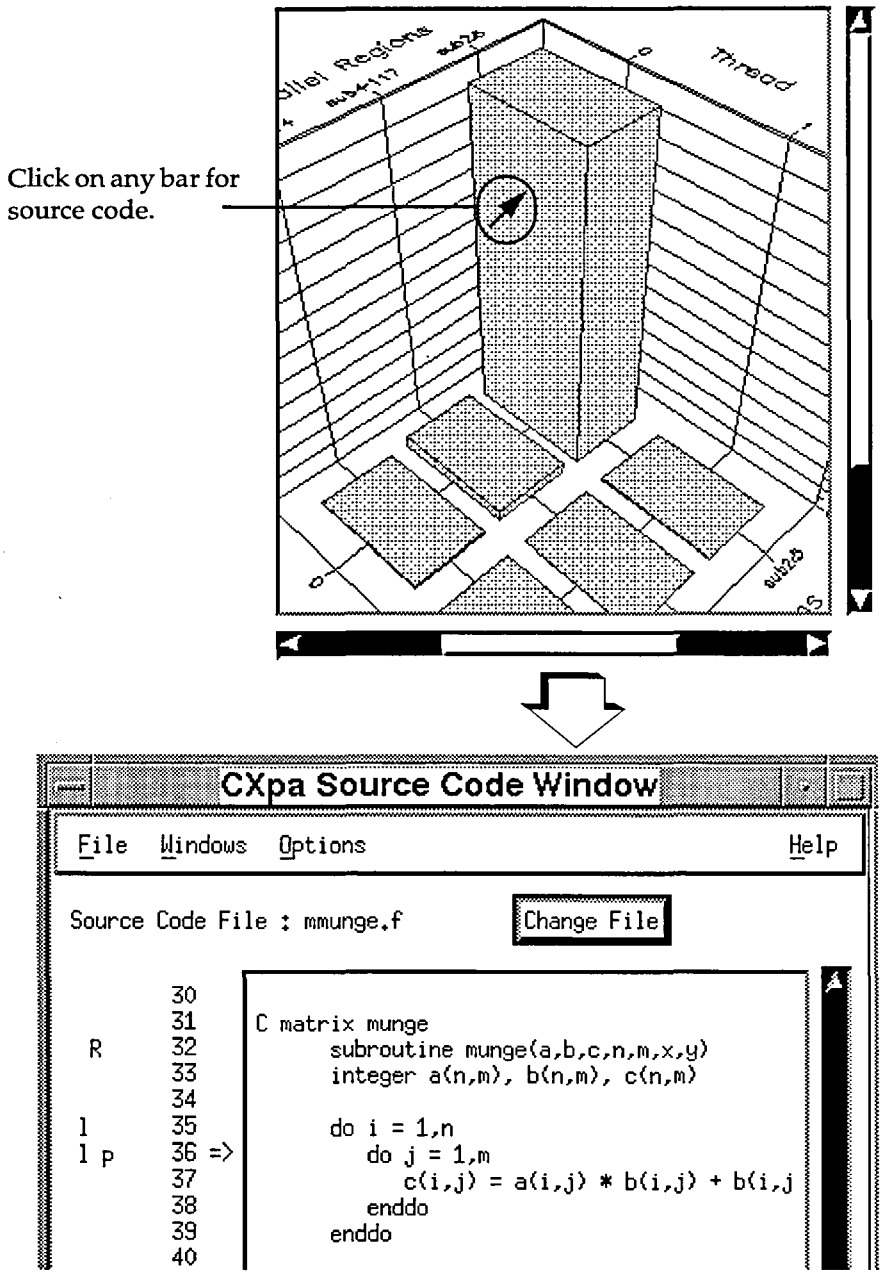
   To select the type of profile or report you want to use to display profiling results, position the mouse pointer over the button and hold down the left mouse button. Point at the option you want and release the mouse button. A window displaying performance results appears.



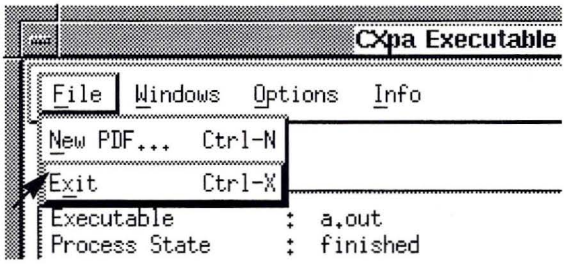


The above example (a 3D Profile window) shows the CPU time for parallel regions in a program running on an SPP Series system; performance information in the 3D Profile window is graphed on a per-thread basis.

8. From the 2D or 3D Profile window, you can display the source code represented by a bar in the graph by clicking on the bar with the left mouse button. The Source Code window appears with an arrow (=>) pointing to the start of the section of code represented by the bar you clicked.



9. Quit CXpa by selecting the Exit option from the File menu of the Executable Manager window.



Using CXpa in line mode

If you are working from a CRT or if you prefer a line-oriented interface in an xterm window, you can use line mode. To invoke CXpa in line mode, you must use the `-nw` option at the command line.

To use CXpa in line mode:

1. Compile your program with:

   `-cxpa` to instrument routines, loops, and parallel regions
   `-cxpar` to instrument routines only
   `-cxpab` to instrument basic blocks only

   For example:

   ```
   fc -cxpa -O3 prog.f
   ```

2. Invoke CXpa with an executable in line mode:

   `cxpa -nw a.out`

   Your shell prompt changes to the CXpa line-mode prompt:

   ```
                CONVEX Performance Analyzer

   Type 'help' for help.
   Reading executable a.out...
   Selecting profile a.out.pdf...

   (CXpa)
   ```

3. Select the source code regions of your program you want to profile with a form of the `select` command. For example:

NOTE: In line mode, if you do not select one or more source code regions in your program for profiling with the `select` command, CXpa will not collect any metrics.

```
(CXpa) select loop all
```

The above command selects all loops in your program for profiling.

4. Choose the type of metrics that you want to collect with the `collect` command. You can select one or more of the following four options:
   - `cpu`—Collects CPU time (available on all architectures).
   - `wall_clock`—Collects wall clock time (available on all architectures).
   - `events`—Collects memory access events and latency time. Use the `set events` command to specify the type of events collected (available on SPP Series and C4 machines only).
   - `vector_flops`—Collects data on vector register usage at the loop region level (available on C Series machines only).

   For example:

   ```
   (CXpa) collect cpu events
   ```

5. If you enabled event collection with the `collect` command, choose the type of event you want to collect with the `set events` command:

NOTE: Each use of the `set events` command overwrites its previous setting. Only one type of event can be collected per run of your program.

The type of events you can collect are architecture-specific. Refer to the reference page for the `set events` command for more information.

```
(CXpa) set events local_misses
```

The above command tells CXpa to collect the number of times your program had to access hypernode local memory to find a value due to a processor cache miss. The `local_misses` parameter is specific to SPP Series machines.

6. Run your program with the `run` command:

   ```
   (CXpa) run
   ```

   Output from your program appears as it runs.

While your program is running, you can type **CTRL-c** to pause your program.

NOTE: To obtain the most accurate profiling data, you should not pause your program during profiling. For best results, run the program to completion and then analyze the results. You will also see more accurate results if you run the program in a standalone environment.

Once paused, the CXpa prompt will reappear. While your program is paused you can:

- View intermediate performance reports by using the `analyze` command.
- Resume the execution of your program by using the `continue` command.
- Terminate execution of your program with the `stop` command.

7. When your program finishes, use a form of the `analyze` command to view final performance reports. For example:

```
(CXpa) analyze loop
```

The `analyze` command generates and displays routine, loop, block, and parallel region performance reports. To display a specific report, use the `analyze routine`, `analyze loop`, `analyze block`, or `analyze pregion` command.

CXpa displays reports in line mode using the pager specified with your PAGER environment variable. If the PAGER environment variable is not set, CXpa uses the `more` command to page the output. You can also redirect output to a file using redirection operators.

8. To exit CXpa, use the `quit` command.

Editing the command line

CXpa's line mode provides command line editing functions similar to UNIX command line editing. You can display the available editing functions by entering or **ESC-?** on the CXpa command line. The editing functions are as follows:

Function	Key sequence
Backward character	**CTRL-b**
Backward word	**ESC-b**
Beginning of line	**CTRL-a**
Capitalize forward word	**ESC-c**
Delete backward character	**CTRL-h**
Delete backward character	**DEL**
Delete backward word	**ESC-h**
Delete forward character	**CTRL-d**

Delete forward word	ESC-d
Display key bindings	ESC ?
End of line	CTRL-e
Erase line	CTRL-g
Erase screen	ESC-g
Execute current command	RETURN
Forward character	CTRL-f
Forward word	ESC-f
Kill to end of line	CTRL-k
Lower case work	ESC-l
Next command	CTRL-n
Previous command	CTRL-p
Redraw screen	CTRL-l
Redraw screen	CTRL-r
Transpose characters	CTRL-t
Transpose words	ESC-t
Upper case word	ESC-u

Related Topics

Advanced compiling
Compiling
Introducing metrics
Introducing source code regions
Reports
Selecting and deselecting regions in line mode
Selecting metrics in line mode
Selecting metrics in X window mode
Selecting regions in X window mode
Using batch mode
Using online help

Related Windows

2D Profile window
Analysis Control window
Executable Manager window
Profile Selection dialog
3D Profile window
Analysis Report window
Help window

Related Commands

`analyze`
`collect`
`help`
`select`
`set events`

Setting the PDF

When you use CXpa to profile a program, it creates a performance data file (PDF) to store profiling data. The PDF is a binary file that contains the performance data for a single run of your program. The data in a PDF is used to create 2D and 3D profile graphs and analysis reports. If you do not set the PDF name, CXpa uses the default PDF name of *<executable>*.pdf.

NOTE: PDFs are platform independent. For example, the data stored in a PDF created on an SPP Series computer can be viewed (in reports or graphs) on a C Series computer, or vice versa.

You can change (set) the name of the PDF to be written to or read from during a CXpa session. You may want to do this for two reasons:

- **To prevent CXpa from overwriting an existing PDF**—If you have invoked CXpa with the name of an executable, when you run your program CXpa overwrites all of the data in the PDF. If you do not want this to occur, you must change the name of the PDF between runs of your program, as described in the following sections.
- **To analyze a different PDF**—If you have invoked CXpa with the name of a PDF file only, you can analyze PDFs created in previous CXpa sessions, including PDFs created on different architectures or from different executables. In analysis mode, you can analyze and compare data for several PDFs during a CXpa session.

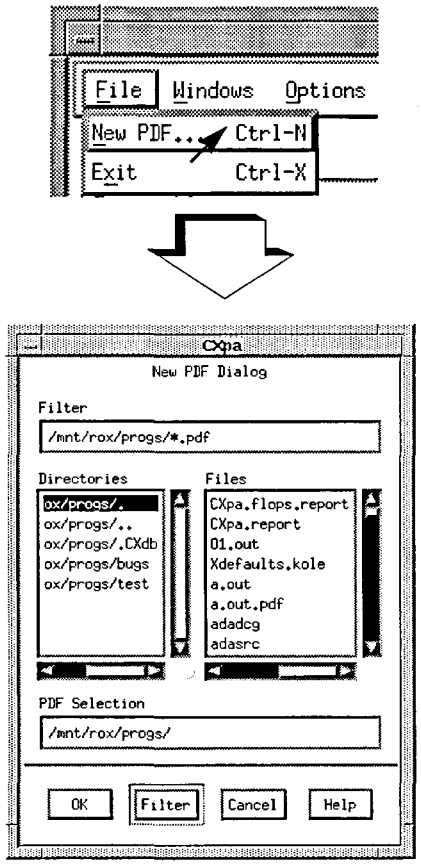
NOTE: If you invoke CXpa with an executable, you can only analyze a PDF created during the current CXpa session. To analyze a PDF created with a different executable or multiple PDF files, invoke CXpa without specifying an executable or with the name of a PDF file.

Changing the PDF name in X window mode

As shown in the figure that follows, perform the following steps from the Executable Manager or Analysis Control window:

1. Select the New PDF option from the File menu in the Executable Manager window or the Open PDF option from the File menu in the Analysis Control window. The New PDF or Open PDF dialog appears.
2. Choose a new PDF name by clicking in the Files list or by typing a new name in the PDF Selection field.
3. Press OK.

CXpa will now use the new PDF during this profiling session.



Changing the PDF name in line or batch mode

To choose a new PDF name in line mode, use the `set pdf` command:

`set pdf` *<new_pdf_name>*

CXpa will now write to and/or read from the specified PDF during this profiling session:

- If you invoked CXpa with an executable, profiling data will be written to the specified PDF when you run your program and performance analysis reports and graphs will be generated from the data in that PDF.
- If you invoked CXpa with the name of a PDF file only (without specifying an executable file), performance analysis reports and graphs will be generated from the data in the PDF that you specified.

Related Topics	Analyzing PDFs only
Related Windows	Analysis Control window Open PDF dialog Executable Manager window New PDF dialog

Related Commands	`analyze`
	`set pdf`
	`cxpa`

Analyzing PDFs only

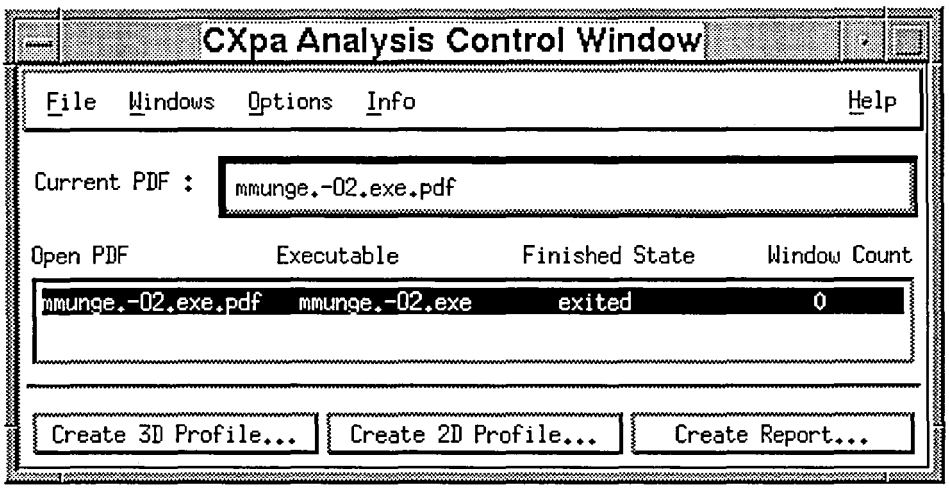
In both X window and line modes, you can view only performance data in PDFs that were created in previous CXpa sessions when you invoke CXpa with the name of a PDF file only (without specifying an executable).

Analyzing PDFs in X window mode

Start CXpa with the name of an existing PDF file only:

`cxpa` *<pdf-name>* &

The Analysis Control window appears.



NOTE: You can use a CXpa X application resource to specify automatic creation of a profile or report window when you invoke CXpa with the name of a PDF file (without specifying an executable). The resource is `Cxpa*defaultWindow`, and it can accept one of four values: `2DProfile`, `3DProfile`, `Report`, or `NoWindow`. The default is `NoWindow`.

From the Analysis Control window, you can analyze PDFs that were created in previous CXpa sessions, including PDFs that were created on different architectures or from different executables. This enables you to analyze and compare data in multiple PDFs.

Opening other PDFs

To open another PDF, select the Open PDF option from the File menu. A new entry appears in the PDF list. You can open a PDF and make it active by typing a PDF name directly in the Current PDF field.

Choosing an active PDF

You can choose the active PDF by clicking on a PDF in the Open PDF column. The buttons at the bottom of the Analysis Control window create windows that contain 2D or 3D profile graphs or textual reports from information in the active PDF.

Analyzing PDFs in line mode

To analyze PDFs only in line mode, start CXpa with the name of an existing PDF file (without specifying an executable):

`cxpa` *<pdf_name>* `-nw`

You will see a CXpa prompt. In this analysis mode, you can analyze PDFs created in previous CXpa sessions, including PDFs created on other architectures.

Use the `analyze` command to generate reports from the data in that PDF.

Opening other PDFs

If you invoked CXpa with a PDF only (without specifying an executable), you can view another PDF without quitting CXpa. To change PDFs, use the `set pdf` command:

`(CXpa)` **`set pdf`** *<pdf_name>*

Related Topics

Setting the PDF

Related Windows

Analysis Control window
Open PDF dialog
Executable Manager window
New PDF dialog

Related Commands

`analyze`
`set pdf`
`cxpa`

Using batch mode

This section describes how to use CXpa in batch mode from the command line or from a shell script.

Using batch mode from the command line

To use CXpa in batch mode from the command line, specify a command file for input when you start CXpa by using the `-x` option and redirect input and output. For example:

`cxpa -x` *<cmdfile>* `a.out <` *<input_file>* `>&` *<output_file>*

The above command tells CXpa to execute a command file at startup, read input to your program from a file, and redirect all output and messages to a file.

Command file input using the `-x` option

You can tell CXpa to execute a command file from the command line by using the `-x` option. When you use the `-x` option, CXpa executes in batch mode. The following is an example of using the `-x` option to specify a command file:

```
cxpa -x cmdfile a.out
```

CXpa executes the command file and quits when it encounters the `quit` command or the end of file. The following file listing is an example of a CXpa command file:

```
#This line is a comment.
select all
collect cpu wall_clock
run
analyze > cxpa.report
quit
```

A CXpa command file is a text file with that contains a list of CXpa commands. Each command must appear on a separate line. The # symbol denotes comments.

Argument input using the `-e` option

You can specify program arguments on the CXpa command line with the `-e` option. These arguments are used when you execute your program in CXpa with the `run` command. This option is helpful for integrating CXpa with existing program execution shell scripts and is available in CXpa's line mode, batch mode, and X window interface.

The following example shows how to use the `-e` option to specify program arguments:

```
cxpa -x cmdfile -e a.out 12 35 14
```

Following the `-e` option, CXpa expects the name of the executable (optional) followed by program arguments. No other CXpa options may follow the `-e` option.

The following section explains how to use integrate CXpa with a shell script that compiles and runs a program.

Using batch mode from a script

The following example shows a way to integrate CXpa into a script that compiles and runs a program.

```
#!/bin/csh -f
#
#Name: batch_script
#Run this script with cxpa if command line switch -cxpa is found
#

set PROFILER = ''
set PROFILER_COMP_FLAG = ''
foreach arg ($argv)
  if ($arg == '-cxpa') then
    set PROFILER = '/usr/convex/cxpa -x cxpa.script -e'
    set PROFILER_COMP_FLAG = '-cxpa'
    cat << EOF >! cmd_file
      select routine all
      run
      analyze
      quit
EOF
  endif
end

# compile a.out
#
cc $PROFILER_COMP_FLAG -O0 foo.c -o a.out

# Now run the executable
#

$PROFILER a.out arg1 arg2
```

If you include `-cxpa` when you invoke this script, it will compile the program with a CXpa option (`-cxpa`), invoke CXpa with the resulting executable file, and execute a CXpa command file that performs a batch profiling session.

To use CXpa through this script, you would use the following command line:

```
% batch_script -cxpa
```

Related Commands

`add path`
`analyze`
`collect`
`continue`
`cxpa`
`deselect`
`path`
`rerun`
`run`
`select`
`set events`
`set pdf`
`set subcomplex`
`set visibility`
`source`
`stop`
`quit`

Using online help

In X window mode, you can access the Help window by pressing any Help button or by choosing an option from any Help menu in the upper right corner of any CXpa window. Refer to the reference page for the `help` command for information about accessing online help for commands in line mode.

NOTE: To view release information on CXpa, you can look at the online Release Notice by selecting the *Release Notice* option under the Help menu of the Executable Manager or Analysis Control window.

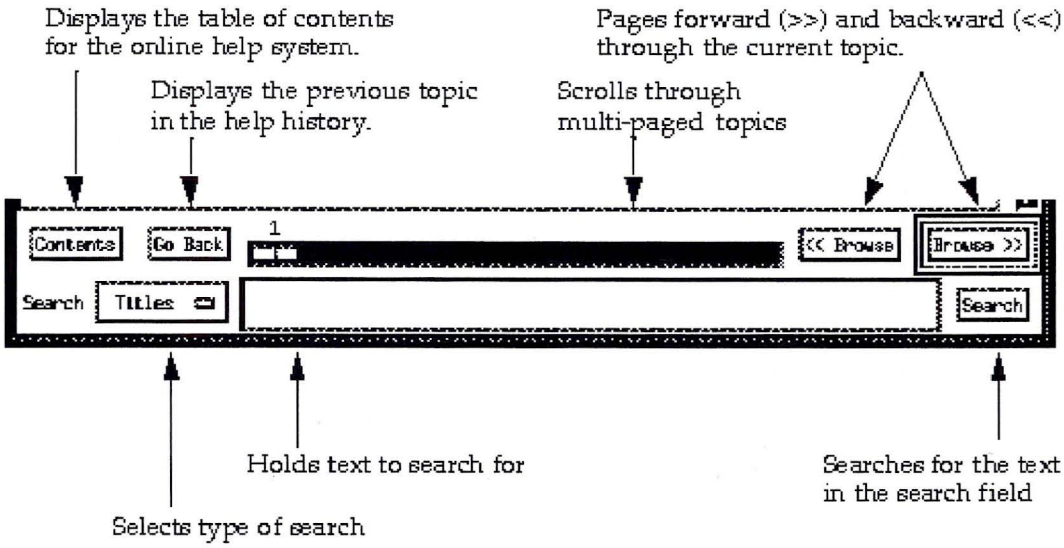
This section explains how to use the following features of the Help window:

- Using the buttons and scroll bars
- Using the menus
- Selecting a topic
- Searching for a topic
- Exiting from help

You can access this help page online by selecting the Using Help option of the Help menu in the upper-right corner of any CXpa window.

Using the buttons and scroll bars

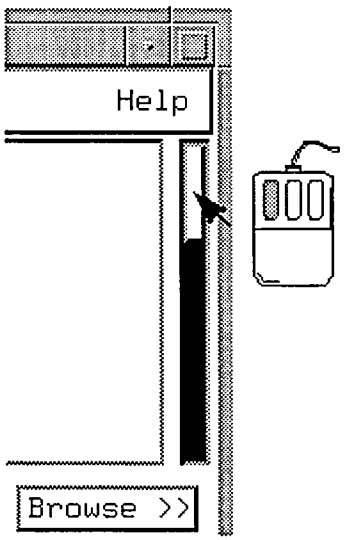
The buttons in this window are for navigating the online help system:



The following list describes each button in detail:

- **Contents**—Displays the table of contents for the online help system. The table of contents provides an overview of and access to the rest of the help system.
- **Go Back**—Displays the previous topic stored in the help history. The help history contains all the topics viewed during the current session.
- **Slider**—Scrolls through the pages of the current topic. If the current topic does not consist of multiple pages, the slider is not displayed.
- **Browse**—Pages forward (>>) or backward (<<) through the current topic, one window length at a time. These buttons are deactivated if there is no next or previous page.
- **Search Field**—Provides a place enter the text you wish to search for.
- **Titles/All Text**— Specifies a search of reference page titles or the complete text of all pages.
- **Search**—Searches for the text entered in the Search Field.

To scroll through the current page, point to the scroll bar with the mouse, and hold down the left mouse button while sliding the bar up and down to scroll the text. (The scroll bar does not appear if the entire page is displayed in the Help window.)



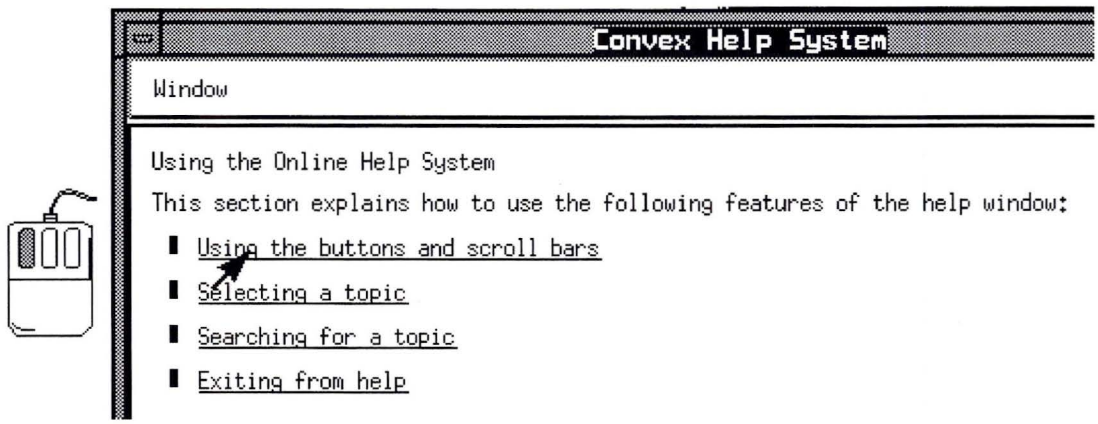
Using the menus

The two menus at the top of the window are described below:

- Window menu
  - **Print Text**—Prints an ASCII version of the current topic to your default printer using the `lpr` command. Your default printer is determined using the PRINTER environment variable.
  - **Close**—Closes the Help Window. This exits you from the Help System.
- Help menu
  - **On Help**—Displays the "Using the Online Help System."
  - **On Version**—Displays version information for the Convex Online Help System.

Selecting a topic

In this Help window, any underlined text is a help topic name. Click on the underlined text with the left mouse button to view the help page for that topic.



Searching for a topic

You can search for a topic name by performing the following steps:

1. Select the type of search you want to perform. Searching the titles of the topics is generally quicker, while searching All Text is more thorough. Typically, you will want to search the titles first, and then search All Text if a titles search proves unsuccessful.

2. Activate the Search Field by moving the cursor into the field area.
3. Type the character string that you want to search for. You can include spaces in the string, but you cannot use a regular expression.

   The topics you can search for include:
   - Report names
   - Window or dialog names
   - Topic names (for example, PDF, event, metric, line mode, and so on)
   - Message ID numbers (for example, A18)
   - CXpa commands

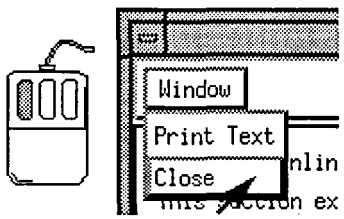
4. Click on the Search button, or press **RETURN**.

The Help window displays a list of topic names that match the search string. To view one of the topics from the search list, click on it with the mouse.

If only one topic name matches the search string, the Help window displays that topic directly.

Exiting from help

To exit from this help system, activate the Window menu with the left mouse button, and select the Close option.



Compiling for profiling 2

This chapter explains how to compile your source files for use with CXpa. You must compile your program with a CXpa compiler option before you can profile it.

You need only use CXpa compiler options on the source files of your program that you wish to profile. You can also choose what kinds of regions in your program you want to profile, whether routines, loops, parallel regions, or basic blocks.

This chapter contains two reference pages on compiling: "Compiling for CXpa" and "Advanced compiling." The following outline indicates the information covered in each page:

- Compiling for CXpa:
  - Compiler syntax and options
  - Compiling and linking in one step
  - Compiling and linking separately
- Advanced compiling:
  - Profiling instrumented routines that call uninstrumented routines
  - Using the `-cxpalib` and `-cxpamon` compiler options
  - Problems compiling routines with both `-cxpa` and `-cxpab` options
  - Problems with not compiling parallel regions for use with CXpa (C Series only)
  - Problems including CXpa compiler options (`-cxpamon`, `-cxpalib`, `-cxpa`, `cxpar`, and `-cxpab`) within FORTRAN OPTIONS statements.

Compiling for CXpa

This section describes how to compile programs for use with CXpa.

Syntax

<compiler> [*cxpa_option*] [*optimization_ option*] *<source_files>*

Parameter	Meaning
compiler	Specifies one of the CONVEX compilers: `cc`—The CONVEX C compiler `fc`—The CONVEX Fortran compiler You can also include other compiler options.
cxpa_option	Specifies CXpa options to the compiler: `-cxpa`—Compiles for routine, loop, and parallel region profiling. `-cxpar`—Compiles for routine profiling only. `-cxpab`—Compiles for basic block profiling only. `-cxpalib`—Links your program with system libraries that are instrumented for CXpa. `-cxpamon=`*dir*—Specifies an alternate directory location for CXpa data collection routines (cxpamon.o). The default for *dir* is /usr/lib.
optimization_option	Specifies an optimization level. These relate to profiling in the following ways:
Optimization level(s)	Types of regions available for profiling
`-no, -O0`	Routines (if compiled `-cxpa` or `-cxpar`); basic blocks (if compiled `-cxpab`)
`-O1, -O2`	Loops and routines (if compiled `-cxpa`); basic blocks (if compiled `-cxpab`)
`-O3`	Parallel regions and loops (if compiled `-cxpa`); routines (if compiled `-cxpa` or `-cxpar`); basic blocks (if compiled `-cxpab`)

source_files Specifies the name of one or more source files to instrument for profiling.

Description

Before you can profile your program with CXpa, it must be prepared by the compiler for profiling. The compiler adds instructions to the executable file that enable CXpa to gather performance data during execution of your program.

When you compile your program, you specify the types of regions in your program that you are interested in profiling. You can profile four different types of regions:

- **Routine**—A main routine, functions, or procedures.
- **Loops**—A loop construct (such as the FORTRAN `DO` statement or C `for` statement). To profile loops you must compile at optimization level `-O1` or higher. At lower optimization levels, the compiler does not instrument loops.
- **Parallel regions (loops)**—A loop that has been parallelized by the compiler. To profile parallel loops you must compile at optimization level `-O3`. At lower optimization levels, the compiler does not create parallel loops.
- **Basic blocks**—A basic block construct (such as the statements within a loop). A basic block is determined by the compiler, but is always a single-entry, single-exit section of code.

The compiler adds instructions around these regions in the executable depending upon the compiler option you choose.

There are three compiler options that instrument an executable for profiling. Each compiler option instruments a different type of program region:

- `-cxpa`—Routines, loops, and parallel regions (recommended for most profiling tasks)
- `-cxpar`—Routines only
- `-cxpab`—Blocks only

You can use these with any other compiler options except `-p`, `-pb`, and `-pg`; these options are designed for use with other profilers and cannot be used with CXpa options.

Only one CXpa instrumentation option (`-cxpa`, `-cxpar`, or `-cxpab`) should be used on a single source file when compiling your source code. You can, however, compile different source files for a single program with different options. If you do, use the `-cxpa` compiler option when linking them together.

With CXpa, you can select specific regions to profile. For example, you can profile some or all of the loops that were instrumented by the compiler. Instrumented regions that are not profiled are ignored by CXpa and incur virtually no overhead.

An executable that has been instrumented for use with CXpa can still be executed outside of CXpa. CXpa instrumentation is designed to have a minimal effect on the size and performance of the executable.

Compiling and linking in one step

If you are compiling your source files into an executable with a single call to the compiler, you are compiling and linking in the same step. In this case, object files are not saved, and the executable file is ready to be used by CXpa.

An example of compiling a single source file is:

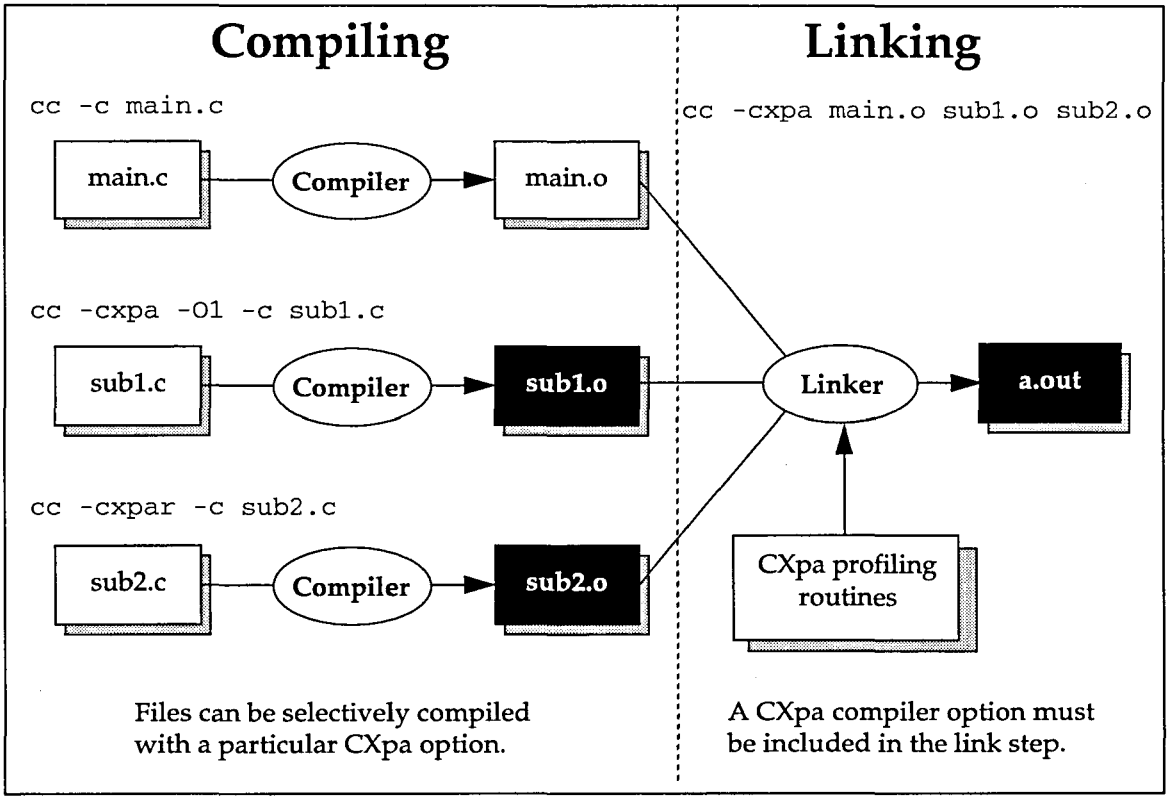
```
cc -cxpa -O1 main.c
```

The source file `main.c` is compiled at optimization level `-O1` with the compiler option `-cxpa` to produce the executable `a.out`. Because the `-cxpa` and `-O1` options were used, the executable has been instrumented for routine and loop profiling.

Compiling and linking separately

Typically, the numerous source files of large programs are compiled separately. Each source file is compiled into an object file using the `-c` compiler option and then linked together into an executable.

When compiling for CXpa, you can compile each source file with the same or different profiling option. You must, however, use the `-cxpa` option when linking as shown in the following figure.



In the previous figure, the program being compiled is made up of three source files. The source file main.c is compiled into an object file (because of the `-c` option) without adding any instrumentation for CXpa.

The source file sub1.c is compiled for routine and loop profiling with the `-cxpa` and `-O1` options, while sub2.c is compiled with the `-cxpar` option for routine-level profiling only.

```
cc -cxpa main.o sub1.o sub2.o
```

Another call to the compiler invokes the linker, which combines the object files into an executable. The linker also links CXpa's timing routines into the executable. You cannot profile using CXpa unless these routines are linked into your executable.

Advanced compiling

For information on the following topics, refer to the "Advanced compiling" section.

- Profiling instrumented routines that call uninstrumented routines
- Using the `-cxpalib` and `-cxpamon` compiler options
- Problems with compiling routines with both `-cxpa` and `-cxpab`
- Not compiling parallel regions for use with CXpa
- Problems including CXpa compiler options (`-cxpamon`, `-cxpalib`, `-cxpa`, `cxpar`, and `-cxpab`) within FORTRAN OPTIONS statements.

Related Topics Advanced compiling

Related Commands `cxpa`

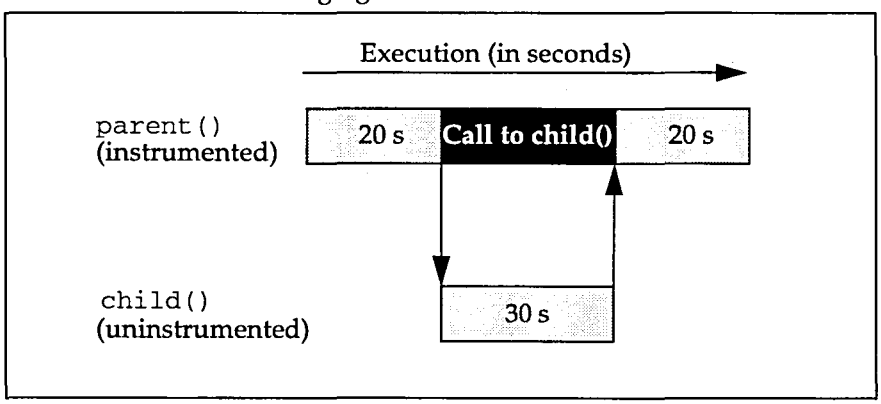
Advanced compiling

This section contains the following advanced compiling topics:

- Profiling instrumented routines that call uninstrumented routines
- Using instrumented libraries with `-cxpalib`
- Specifying an alternate directory for CXpa data collection routines with the `-cxpamon` option
- Compiling routines with `-cxpa` and `-cxpab` simultaneously
- Not compiling parallel regions for use with CXpa
- Problems using the Fortran `OPTIONS` statement with CXpa compiler options

Profiling routines that call uninstrumented routines

If an instrumented routine calls an uninstrumented routine, CXpa will not be able to separate the time spent in the uninstrumented child routine from the time spent in the instrumented parent. This condition is illustrated in the following figure.



In this figure, routine `parent()` has been instrumented for CXpa while routine `child()` has not. The time spent in `parent()` not including children is reported as 70 seconds because CXpa cannot separate time spent in `child()`. If routine `child()` had been compiled with the `-cxpa` flag, CXpa would have correctly reported `parent()` as having executed for 40 seconds not including children.

Using instrumented libraries with `-cxpalib`

Use the `-cxpalib` CXpa compiler option to link your program with installed system libraries that are instrumented for CXpa.

If you use the `-cxpalib` option, be aware that:

- The program you are profiling may execute more slowly.
- Errors in the instrumented libraries may cause your program to terminate abnormally.
- The amount of profiling data will be significantly increased.

If you want to filter out the data from the instrumented libraries during collection and/or analysis, use the `set visibility` command or disable library visibility in the Visibility Selection dialog. To bring up the Visibility Selection dialog, select the Visibility option from the Options menu on the Executable Manager window or the Analysis Control window. By default, both library and application visibility are enabled.

Using the `-cxpamon` compiler option

Use the `-cxpamon` option to specify an alternate directory location for CXpa data collection routines (cxpamon.o) when compiling your program for CXpa. This allows you to specify a different version of the CXpa data collection routines if multiple versions of CXpa are installed on your system. The syntax for this option is

`-cxpamon=`*dir*

where *dir* is the directory containing the alternate version of the CXpa data collection routines (cxpamon.o). The default alternate directory location is /usr/lib.

Problems with compiling routines with both `-cxpa` and `-cxpab`

Compiling a source file with both the `-cxpa` and `-cxpab` options can result in inaccurate timing information because the instrumentation for basic blocks interferes with the timing of routines if basic blocks are enabled.

Problems with not compiling parallel regions for use with CXpa

To get accurate timings on C Series machines, you should compile all routines that contain parallel regions (parallel loops) with the `-cxpa` compiler option when compiling those routines at optimization level `-O3`.

Problems using the Fortran `OPTIONS` statement with CXpa

Do not use CXpa compiler options (`-cxpamon`, `-cxpalib`, `-cxpa`, `cxpar`, and `-cxpab`) within FORTRAN OPTIONS statements.

Choosing performance data to collect

3

This chapter explains how to configure CXpa to collect specific types of performance data (metrics) at specified regions of your program.

The information in this chapter includes:

- Introducing source code regions
- Selecting regions in X window mode
- Selecting and deselecting regions in line mode
- Introducing metrics
- Selecting metrics in X window mode
- Selecting metrics in line mode

Introducing source code regions

Before you run your program under CXpa, you must specify the types of regions of your program for which you want to collect profiling data (metrics). CXpa collects metrics only at the regions selected for profiling.

Depending on the option(s) with which you compiled your source code, four types of source code regions can be selected for profiling:

- **Routines**—Routine regions are only available for profiling if your source code is compiled with the `-cxpa` or `-cxpar` option.
- **Loops**—Loop regions are only available for profiling if your source code contains loops and is compiled with the `-cxpa` option at optimization level `-O1` or higher.
- **Parallel regions**—Parallel regions (parallel loops) are only available for profiling if your source code is compiled with the `-cxpa` option at optimization level `-O3` (at optimization levels below `-O3`, the compiler does not parallelize loops).
- **Basic blocks**—Basic block regions are only available for profiling if your source code is compiled with the `-cxpab` option.

Regions are initially selected or deselected for profiling depending on which CXpa interface you are using:

- In X window mode, all routine source code regions in your program are initially selected.
- In line mode, all available source code regions in your program are initially deselected (you must use the `select` command to select regions for profiling).

If you run your program without selecting any source code regions to profile, no metrics are collected.

Source code annotations for regions

CXpa displays special source code annotations that indicate the location of source code regions that can be selected for profiling and whether or not they are currently selected or deselected (as shown in the following table).

Source code region type	Selected	Deselected
Routine	R	r
Loop	L	l
Parallel region (parallel loop)	P	p
Basic block	B	b

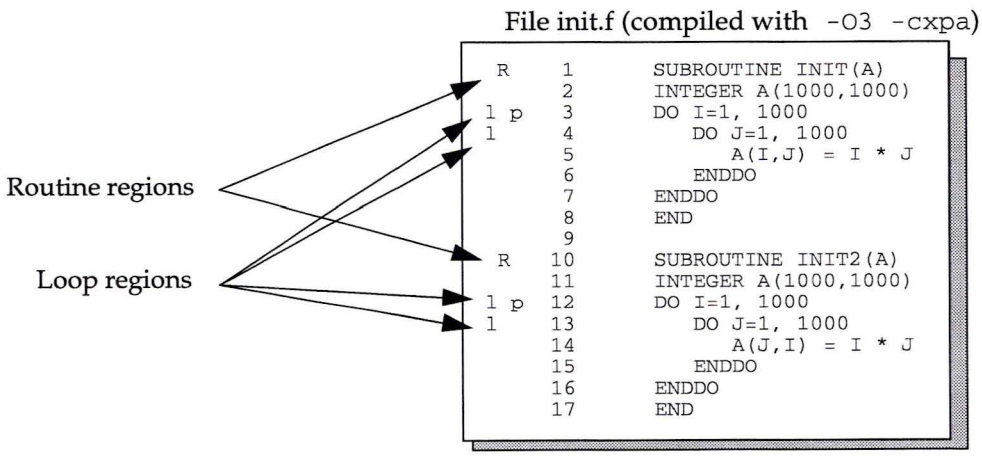
NOTE: Only routine-level profiling data is collected for selected routine regions. To profile loops, parallel regions, or basic blocks inside of a routine, you must select the corresponding loop, parallel region, or basic block region.

In X window mode, you can display annotated source code by:

- Selecting the Source Code option from the Windows menu in any CXpa window
- Clicking on a bar in the graph displayed in the 2D or 3D Profile windows

In line mode, you can display annotated source code using the `list` or `list selectable` commands.

The following figure shows annotated source code for sample routines with several types of selectable source code regions.



Each of the routine regions in the above example is currently selected for profiling. The loop regions beginning at lines 3, 4, 12, and 13 can be selected for profiling, but are currently deselected. The parallel regions beginning at lines 3 and 12 can also be selected for profiling, but are currently deselected. No basic block regions can be selected for profiling because these routines were not compiled with the `-cxpab` option.

When you run your program, CXpa collects performance data at every selected source code region that is executed. You can control which regions of your program are selected for profiling using the Region Selection dialog (in X window mode) or the `select` and `deselect` commands (in line mode).

Selecting source code regions

You can select all available source code regions in your program or a subset. You do not have to recompile your program to select or deselect regions for profiling.

Typically, you will want to select all routine regions the first time you profile your program. This provides a complete picture of your program's performance. Using this information, you can then identify the routines that take the longest to execute.

After you have identified the routines whose performance you want to improve, you may want to select only these specific routines for profiling. With fewer source code regions selected, less time is spent in the timing routines CXpa uses to collect performance data. Selecting code regions for profiling does increase wall-clock time. Profiling time increases with the number of regions selected for profiling. In general, selecting loop regions for profiling increases execution time more than selecting routine regions.

If you want to profile only parallel regions, select only parallel regions for profiling. This provides a more accurate representation of the performance of parallel loops.

CXpa provides functionality to select only specific regions for profiling. You can select or deselect:

- All routine regions
- Source code regions in specific routines
- Source code regions at specific lines (in line mode only)

You can also choose to select or deselect all types of regions (routine, loop, parallel region, and block) or only specific types (for example, only parallel regions).

For instructions on how to select regions in X window mode, refer to the reference topic "Selecting regions in X window mode."

For instructions on how to select regions in line mode, refer to the reference topic "Selecting and deselecting regions in line mode."

Related Topics

Compiling
Selecting regions in X window mode
Selecting and deselecting regions in line mode

Related Windows

Executable Manager window
Profile Selection dialog

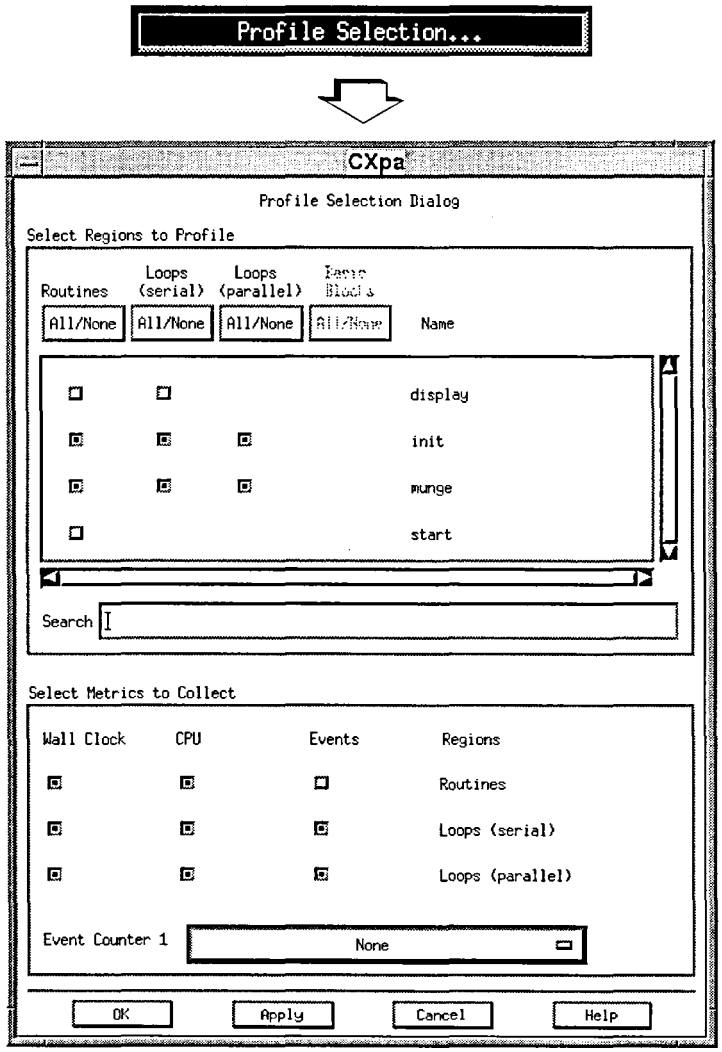
Related Commands

`deselect`
`list selectable`
`select`

Selecting regions in X window mode

By default, all routine regions in your program that were compiled with the `-cxpa` or `-cxpar` options are initially selected for profiling in X window mode.

To select a different set of regions for profiling, press the Profile Selection button in the Executable Manager window. The Profile Selection dialog appears.



Using the Select Regions to Profile portion of the Profile Selection dialog, you can select or deselect the types of regions you want to profile for all routines in your program or for specific routines, as described in the following sections.

Selecting types of regions to profile in all routines

From the Profile Selection dialog, perform the following steps to select types of regions you want to profile in all routines of your program:

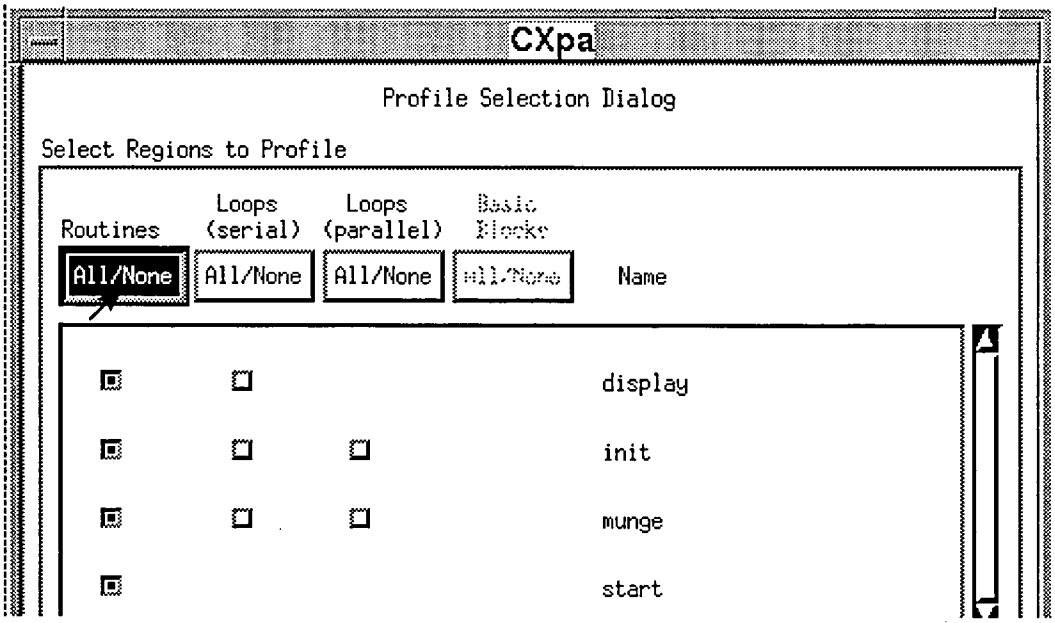
1. Make sure that all routines are selected (by default, all routines are selected when you first bring up the Profile Selection dialog). You can toggle the Routines All/None button for Routine regions to make sure that all routines are selected.
2. Click the All/None toggle buttons for the types of source code regions you wish to profile in all routines. You can:
   - Perform routine-level analysis only for all routines (the default)
   - Profile all serial loops in all routines
   - Profile all parallel loops in all routines
   - Profile all basic blocks in all routines

   If a toggle button is not displayed for a particular region type, it means that no regions of that type can be profiled for that routine. For example, loop regions are not available for profiling unless your program is compiled at optimization level `-O1` with the `-cxpa` or `-cxpar` option.
3. If you wish, select metrics to collect, then press the OK or Apply button once you have finished selecting or deselecting region types and/or metrics to apply the changes and close the Profile Selection dialog.

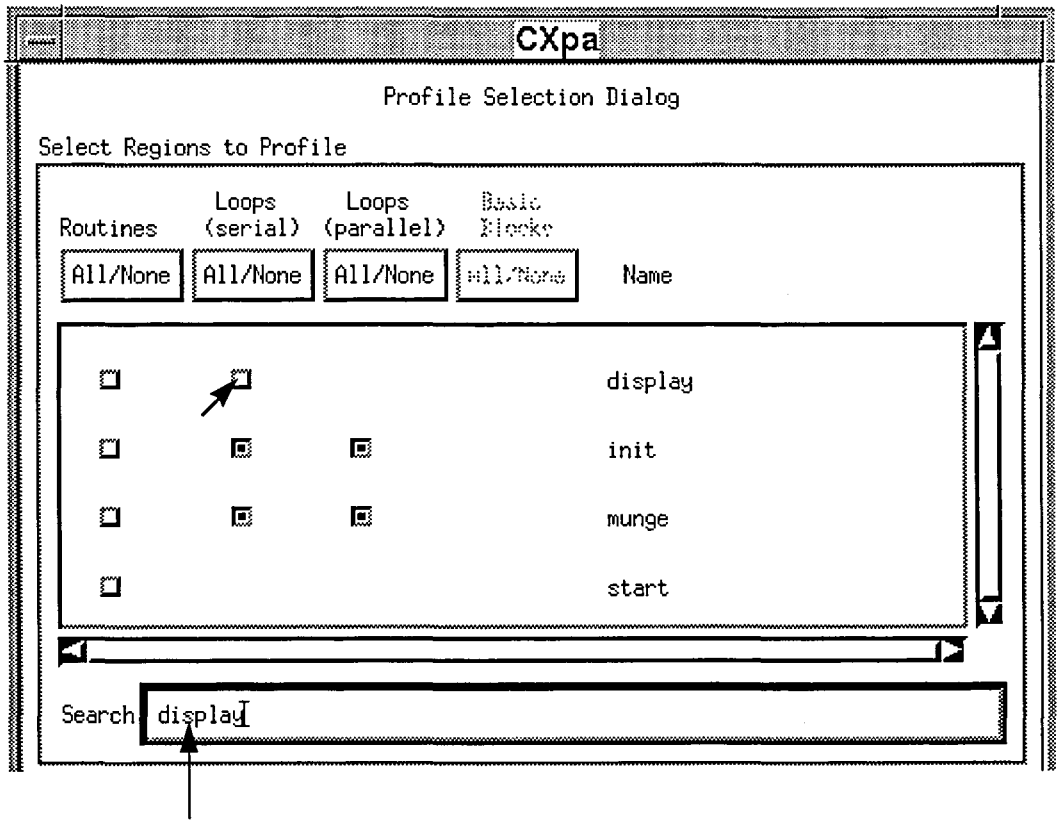
Selecting types of regions to profile in specific routines

From the Profile Selection dialog, perform the following steps to select source code region types for profiling in selected routines only:

1. Press the Routines All/None button to deselect all routine regions as shown in the following figure.



2. Select and/or deselect the types of regions you want to profile for each routine by clicking the toggle buttons to the left of the routine name.



If your program contains a large number of routines, you can type the name of a routine in the Search field, then press **RETURN** to scroll the routine list so that the desired routine is displayed.

If a toggle button is not displayed for a particular region type, it means that no regions of that type can be profiled for that routine. For example, loop regions are not available for profiling unless your program is compiled at optimization level `-O1` with the `-cxpa` or `-cxpar` option.

3. When you have finished selecting and deselecting regions, you can use the Select Metrics to Collect section of the Profile Selection dialog to select a different set of metrics to collect or press the OK button to apply the changes and close the Profile Selection dialog.

Selecting all regions in your program for profiling

From the Profile Selection dialog, perform the following steps to select all regions in your program for profiling:

1. Press the All/None buttons for all available source code region types (routines, loops (serial), loops (parallel), or basic blocks) to select all regions in all routines in your program for profiling.
2. Use the Select Metrics to Collect section of the Profile Selection dialog to select a different set of metrics to collect or press the OK button to apply the changes and close the Profile Selection dialog.

NOTE: Choosing to profile all regions in all routines of your program may significantly increase the amount of time it takes to run your program from CXpa. We advise you to select all routine regions first to find out which routines contain performance bottlenecks. Then, select specific region types in these routines for profiling through the More Detail button.

Related Topics

Compiling
Introducing source code regions
Introducing metrics
Selecting metrics in X window mode

Related Windows

Executable Manager window
Profile Selection dialog

Selecting and deselecting regions in line mode

In line mode, you can select or deselect any set of source code regions in your program with a variant of the `select` and `deselect` commands. You can:

- Select or deselect one type of source code region:
  - In all routines
  - In specific routines
  - At specific lines
- Select or deselect all source code regions:
  - In specific routines
  - At specific lines
  - In all routines

In line mode, all available source code regions in your program are initially deselected, so if you run your program without using the `select` command (that is, without selecting any source code regions to profile), no metrics are collected.

Selecting specific regions to profile provides greater control over profiling and can shorten profiling time.

Each of these methods is described in detail in the sections that follow. The same source code is used in all figures to contrast different command options.

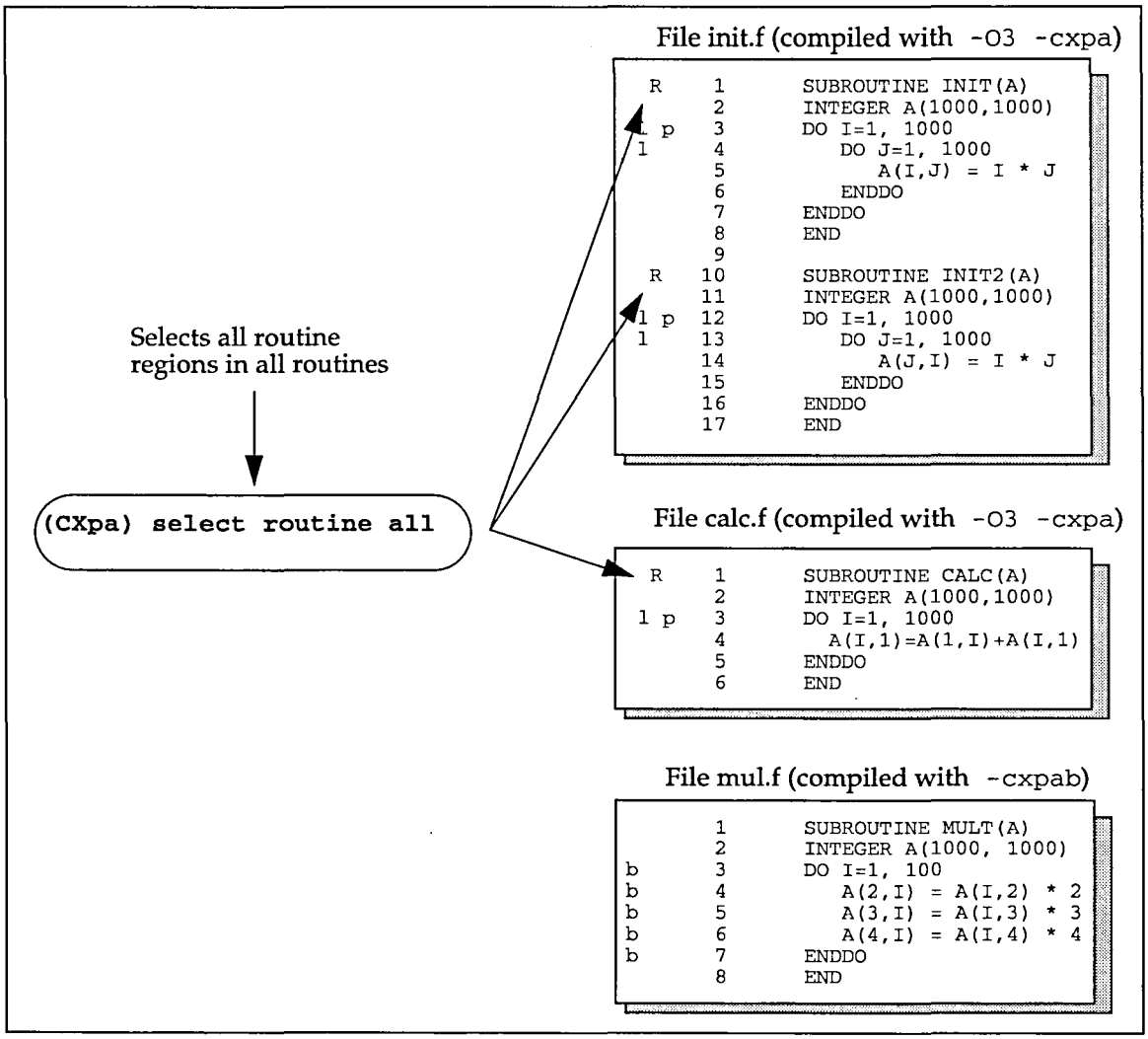
In the examples, uppercase annotations indicate selected source code regions, while lowercase annotations indicate deselected source code regions. In line mode, annotated source code is displayed using the `list` or `list selectable` commands.

Selecting or deselecting one type of region in all routines

To select or deselect one type of source code region in all routines, use one or more of the following variants of the `select` or `deselect` commands:

- `select routine all` or `deselect routine all`—Selects or deselects all routine regions in all routines compiled with `-cxpa` or `-cxpar`. Using the command `select routine all` is the best way to begin a profiling session because it identifies the routines that contain performance bottlenecks.
- `select loop all` or `deselect loop all`—Selects or deselects all loop regions in all routines compiled with `-cxpa` at optimization level `-O1` or higher.
- `select pregion all` or `deselect pregion all`—Selects or deselects all parallel regions in your program for all routines compiled with `-cxpa` at optimization level `-O3`.
- `select block all` or `deselect block all`—Selects or deselects all basic block regions in your program for all routines compiled with `-cxpab`.

For example, the following figure shows that the `select routine all` command selects all routine regions in the routines compiled with `-cxpa` or `-cxpar`. No other regions are affected.



Selecting or deselecting one type of region in specific routines

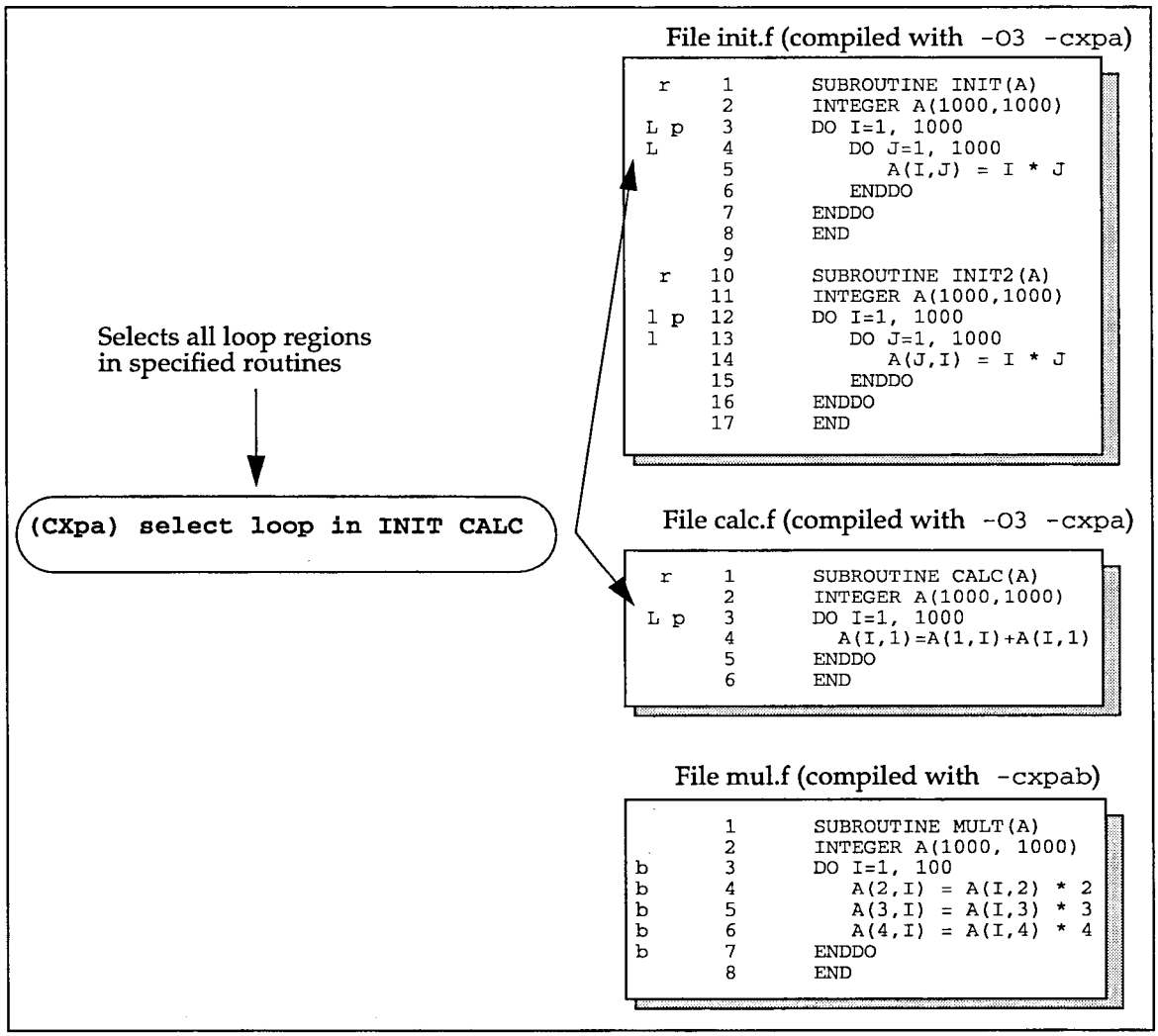
You can select or deselect one type of source code region in specific routines using one or more of the following commands:

- `select loop in` or `deselect loop in`—Selects or deselects all loop regions in the specified routine (or routines) compiled with `-cxpa` at optimization level `-O1` or higher.

- `select pregion in` or `deselect pregion in`—Selects or deselects all parallel regions in the specified routine (or routines) compiled with `-cxpa` at optimization level `-O3` or higher.
- `select block in` or `deselect block in`—Selects or deselects all basic block regions in the specified routine (or routines) compiled with `-cxpab`.

Each of these commands selects all regions of the indicated type in the specified routine (or routines). Multiple routines are separated by spaces on the command line.

In the following figure, the `select loop in` command selects all loop regions in the `INIT` and `CALC` routines. No other regions are affected.



Selecting or deselecting one type of region at specific lines

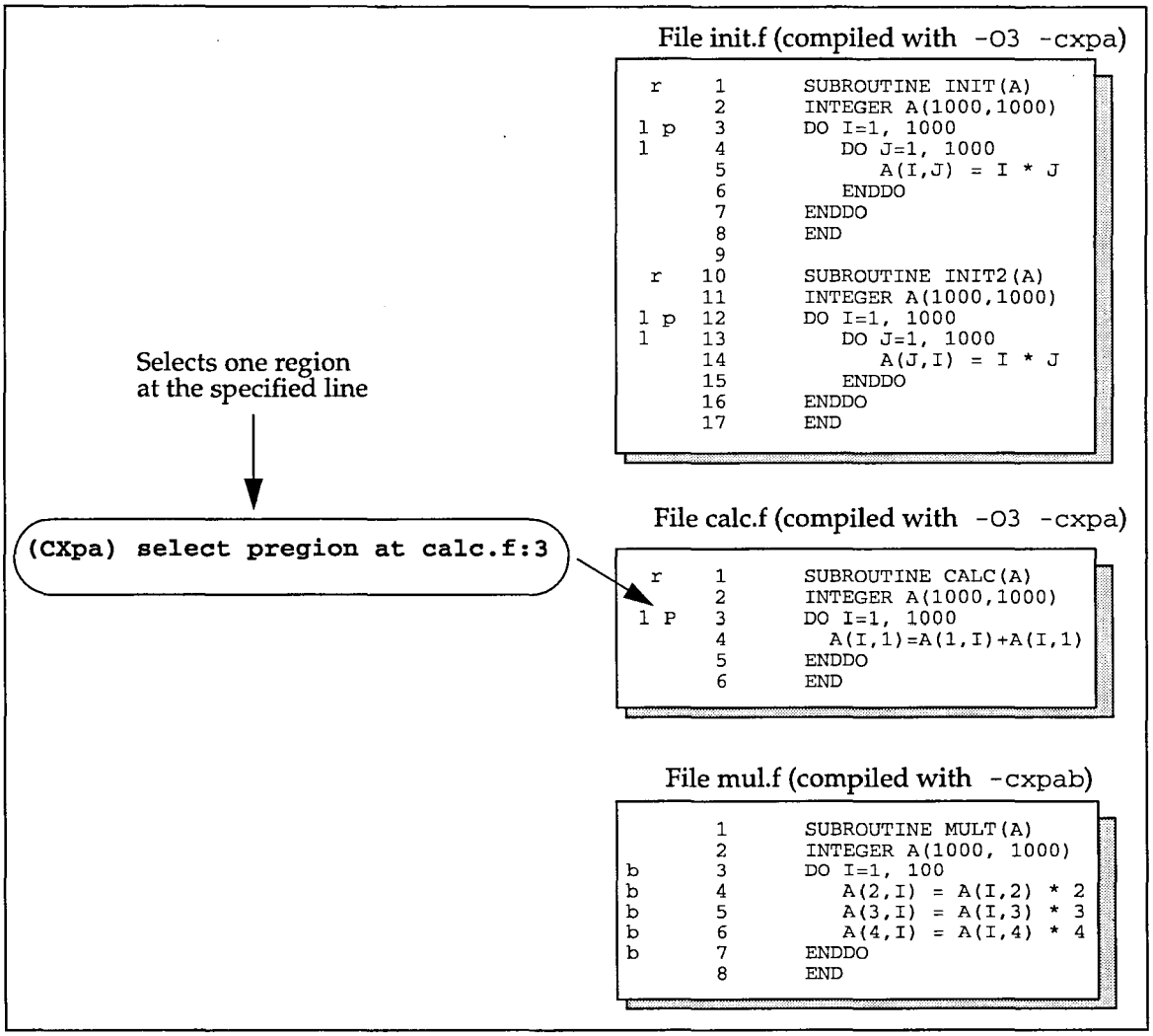
If you do not want to select or deselect all regions at a specific line, you can select or deselect a single region type at a line. This makes it possible to profile a parallel region without profiling the loop around it.

To select or deselect one type of region at a specific source line (or source lines), use one or more of the following commands:

- `select loop at` or `deselect loop at`—Selects or deselects the loop region at the specified line number (or line numbers) in routines compiled with `-cxpa` at optimization level `-O1` or higher.
- `select pregion at` or `deselect pregion at`—Selects or deselects the parallel region at the specified line number (or line numbers) in a routine compiled with `-cxpa` at optimization level `-O3` or higher.
- `select block at` or `deselect block at`—Selects or deselects the basic block region at the specified line number (or line numbers) in routines compiled with `-cxpab`.

Using one of these commands selects the region of the indicated type at the specified line in the specified file. Multiple line numbers can be specified and are separated by spaces on the command line.

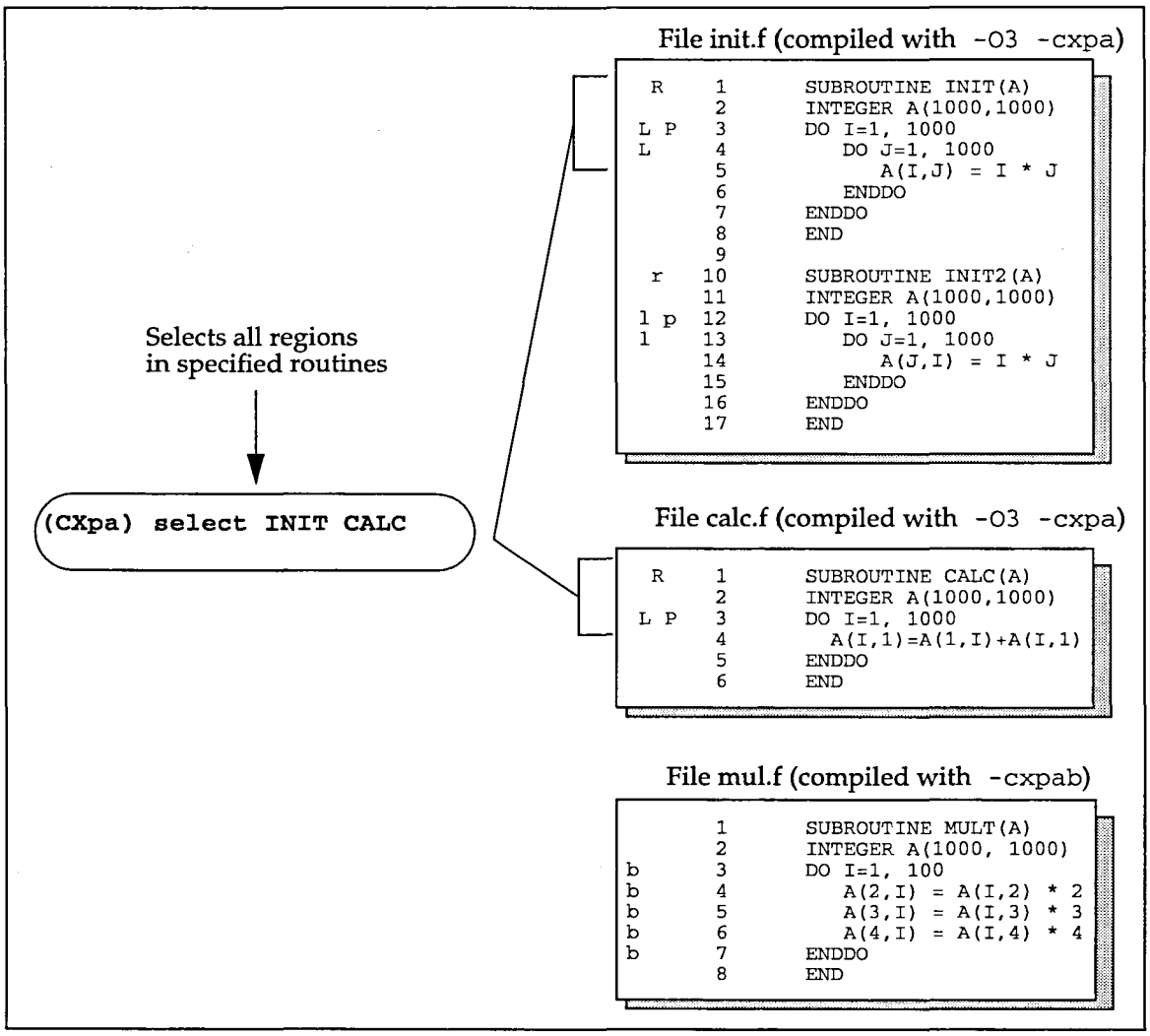
The `select pregion at` command in the following figure selects the parallel region at line 3 in the `calc.f` source file. The loop region on the same line remains deselected.



Selecting or deselecting all regions in specific routines

After profiling all routines in your program, you may want to profile only specific routines whose performance you want to improve. Use the commands `select` *<routine-name>* or `deselect` *<routine-name>* to select or deselect all regions in a routine. You can enter multiple routine names by separating them with a space.

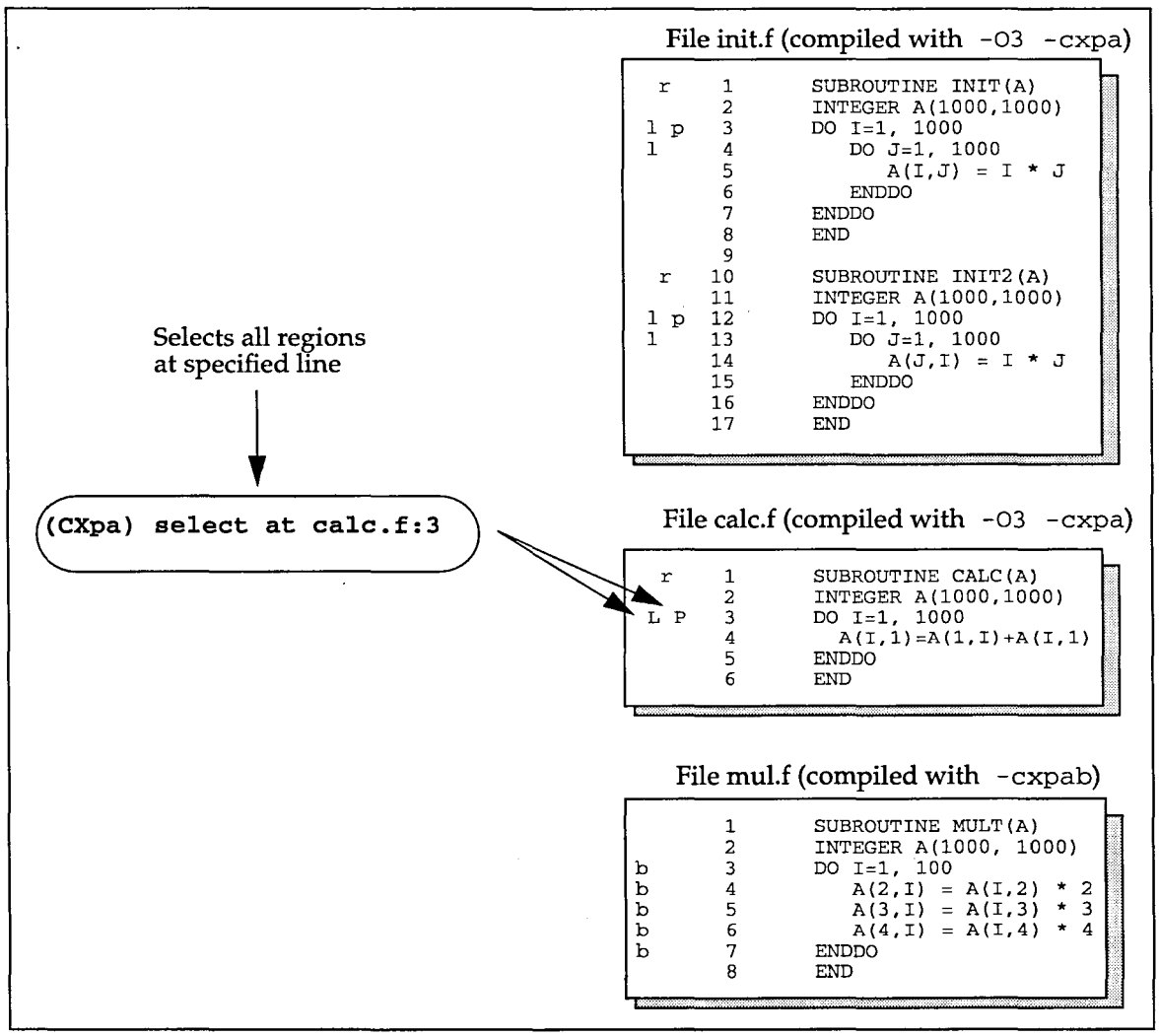
The following figure illustrates how to select all regions in specific routines.



Selecting or deselecting all regions at specific lines

You can select or deselect all regions at a specific source line. This makes it possible to profile a single loop in a routine without having to profile all of the loops within that routine.

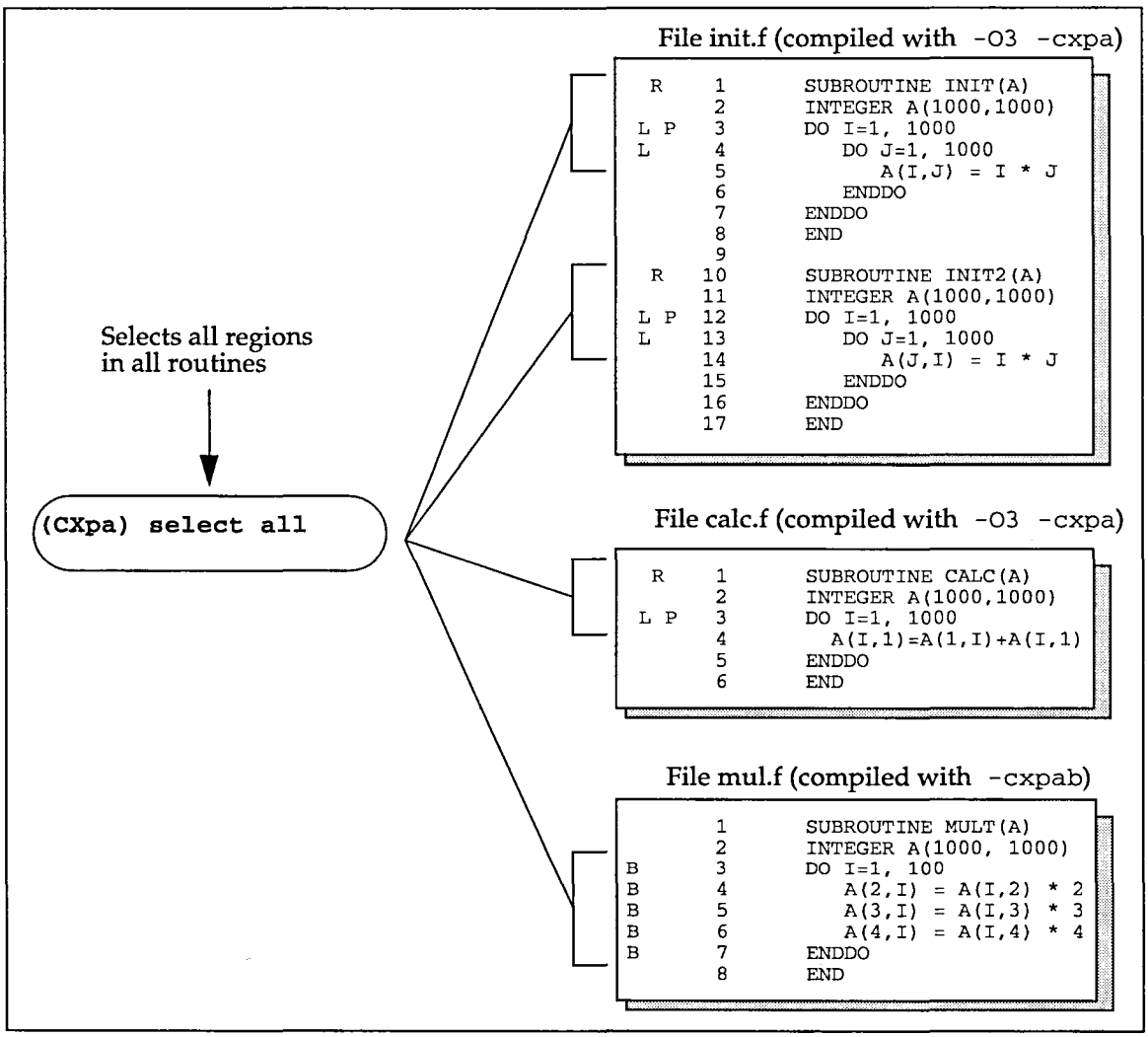
To select or deselect all regions at a line in a specific routine, use the `select at` or `deselect at` commands, followed by the file name and line number you want to enable. The file name and line number are separated by a colon. The example below shows how to select all regions at a specific line.



Selecting or deselecting all regions in all routines

Selecting all regions in all routines in your program can significantly increase the amount of time it takes to profile large programs. However, you may want to select all regions in all routines to get a broad picture of where time is being spent in your program.

Use the `select all` command to select all regions throughout your program, as shown in the following figure.



Use the `deselect all` command to deselect all regions in all routines of your program.

Related Topics	Compiling	Introducing source code regions
Related Commands	`deselect` `select`	`list selectable`

Introducing metrics

After selecting regions of your source code to profile, you can specify what kinds of data you want to collect for these regions. Collecting and comparing different metrics can help you discover performance bottlenecks:

- Loops that generate excessive cache misses
- Regions of your code that spend a significant amount of their CPU time waiting on memory
- Lack of any effective parallelism in a particular loop or routine
- Bank-contention problems which may be indicated by the average time of a cache miss for a particular region
- Uneven distribution of work across threads in parallel regions of your application

For example, if you are profiling a program on a multinode SPP Series system, you can determine how often the profiled regions of your program access remote memory. Because accessing remote memory may increase execution time, you may want to insert compiler directives or rewrite the parts of your code that require unnecessary remote memory access.

Metrics

By default, CXpa measures CPU time and iteration/execution counts for profiled regions of your program. Available metrics are listed and described in the sections that follow. Metrics designated as *computed* are not directly selectable, but are derived from other metrics.

All architectures

Execution/iteration counts

	Number of times a loop, routine, basic block, or parallel region is executed.
Wall Clock Time	Time to solution. It includes time when the process is idle.
CPU Time	Time the CPU works on the process.

CPU/Wall clock time — Concurrency factor for parallel regions. It indicates the speed-up achieved through parallelization. Computed if CPU and wall clock time metrics are selected.

C Series

Vector Mflops — (C Series only)

Millions of floating point operations per second. This includes the following metrics:

Vector spills—Number of times that a previous value in a vector register had to be written to memory.

Vector flops—Number of arithmetic vector floating point operations in the loop. This count does not include load or store operations, and assumes vector options are not masked.

Chime counts—Number of chimes occurring within a given loop. Chime is an acronym for chained instruction measurement. The greater the chime count, the more resource conflicts, and therefore the less vector chaining.

Estimated Mflops—Estimated millions of floating point operations per second. There are two subcategories:

Average—Average number of Mflops per loop invocation. This is based on the average CPU time per invocation of the loop.

Peak—Theoretical maximum number of Mflops for this loop on this architecture.

C4 and SPP Series

For the C4 and SPP architectures, metrics can also be defined for memory access events. The types of events profiled vary between C4 Series and SPP Series machines. Latency metrics are not available for the C4 Series.

Event Counts — The number of times a selected event occurred (for example, data cache misses) in the profiled regions of your program.

Event Latency Time — The time the process spent in wait latency for the type of memory event you are collecting.

Event Latency/ CPU	The percentage of CPU time the process spent in wait latency for the type of memory event you are collecting. Computed if events and CPU time metrics are collected.
Event Latency/ Counts	The average time the process spent in wait latency per event for the type of memory event you are collecting. Computed.

SPP Series events

For SPP Series computers, event counts and latency metrics can be collected for memory access events. These events can be collected during read accesses, write accesses, or both. The default is both. Remote memory miss events apply to multinode configurations only.

Memory access events occur when required data has to be retrieved from memory rather than from the processor cache. Finding data in the memory on another hypernode (remote memory) can increase wall clock time. Once you determine which parts of your program are accessing memory inefficiently, you can use compiler directives to help the compiler increase data locality.

To find bottlenecks, you will need to compare and contrast metrics for different events. For example, you may observe a large number of remote memory miss events at a region. However, latency metrics may reveal that even though a large number of misses are occurring in a given region, average latency time is short or total event latency time for that region is not significant.

Definitions

The following definitions will help you interpret SPP event terminology:

- A *read miss* is an instruction or data cache miss caused by a load.
- A *write miss* is a data cache miss caused by a store or a data cache miss caused by a load and clear.
- *Remote* refers to use of the Convex Toroidal Interface (CTI) to find needed data. Use of the CTI implies use of the SPP SCI rings.
- *Local* refers to memory found on the same hypernode as the processor.

Events

NOTE: Remote memory miss events apply to multinode configurations only.

Local memory misses	Collects the number of times that data not found in the processor cache was found in local memory (memory allocated to that processor's hypernode).
Remote memory misses	Collects the number of times that data not found in the processor cache was found in remote memory (memory allocated to another hypernode). Multinode configurations only.
Local and remote memory misses	Collects the number of loads and stores to and from memory into and out of vector registers.

C4 Series events

On C4 Series machines, memory access event counts include:

Data cache accesses	Collects the number of times that a value was accessed in the CPU data cache for the profiled regions of your program.
Data cache misses	Collects the number of times that a value was not found in the CPU data cache for the profiled regions of your program.
Page table cache misses	Collects the number of times that a value was not found in the CPU page table cache for the profiled regions of your program.
Instruction cache misses	Collects the number of times that an instruction was not found in the CPU instruction cache for the profiled regions of your program.
Vector loads and stores	Collects the number of times that a previous value in a vector register had to be written to memory for the profiled regions of your program.

Latency metrics are not available for C4 Series machines.

Related Topics	Selecting metrics in line mode Selecting metrics in X window mode	
Related Windows	Executable Manager window	Profile Selection dialog
Related Commands	`collect` `set events`	`select`

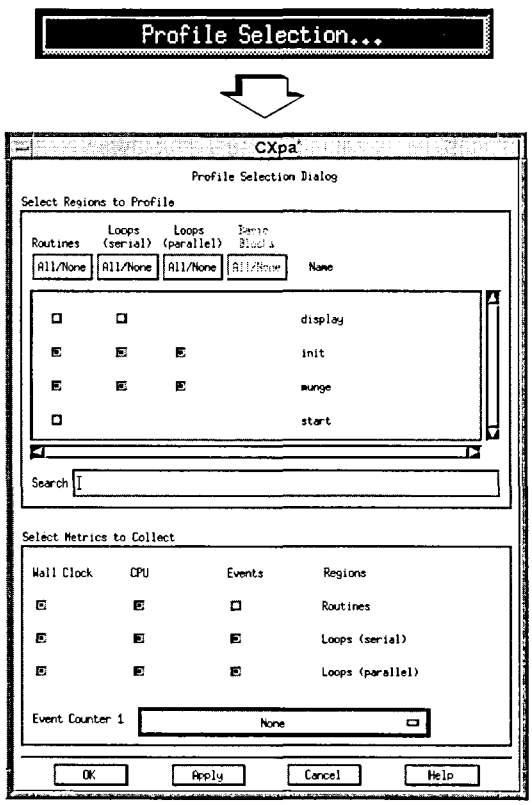
Selecting metrics in X window mode

Once you have selected the regions in your program you want to profile, select the types of metrics you want to collect at those regions when you run your program. By default, CXpa collects CPU time and iteration/execution counts for the regions in your program you have selected for profiling.

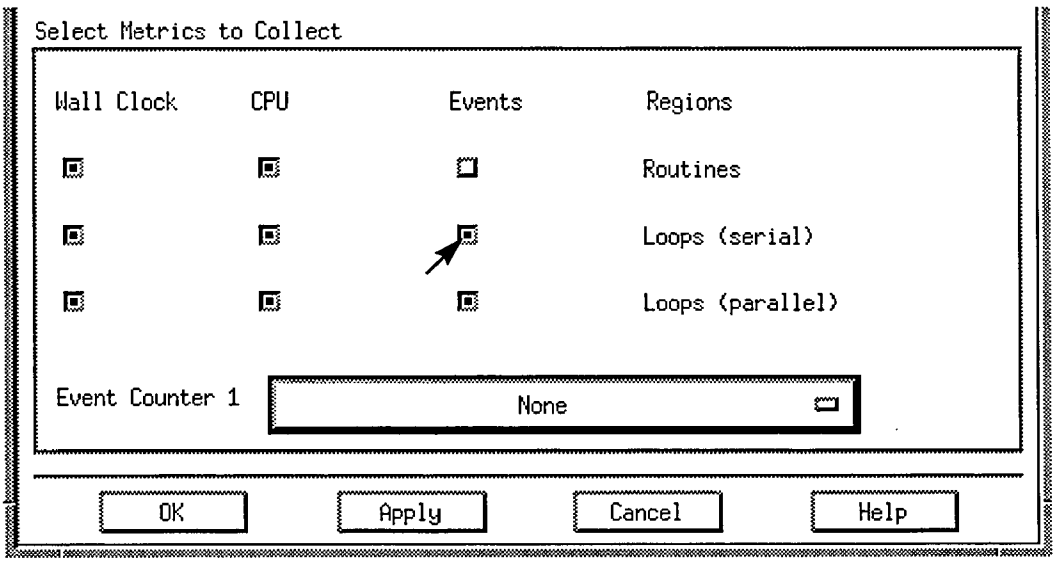
Additional metrics you can collect differ according to machine architecture. Refer to the "Introducing metrics" reference topic for a description of available metrics and the architectures on which they are available. This section describes the procedure for choosing metrics to collect using CXpa's X window interface.

To choose metrics to collect in X window mode:

1. Press the Profile Selection button in the Executable Manager window to bring up the Profile Selection dialog.



2. Make sure that you have selected the regions of your program you want to profile as described in "Selecting regions in X window mode."
3. Specify the metrics that you want to collect by clicking the toggle buttons in the Select Metrics to Collect section of the Profile Selection dialog, as shown in the following figure. CXpa collects these metrics at the regions of your program that are selected for profiling.



Events metrics are only available for SPP and C4 Series systems. If you wish to collect events metrics, you must first choose Events as one of the metrics in the Select Metrics to Collection section of this dialog.

Only one type of event can be collected per run of your program. The Event Counter options menu shows the type of event currently selected.

Refer to the next section, "Selecting events (C4 and SPP Series only)," for instructions on using the Event Counter options menu to select the type of event to collect.

Refer to the "Introducing metrics" online help topic or section in this book for a discussion of available events metrics for C4 and SPP Series systems.

4. Press the OK button to apply the changes and close the Profile Selection dialog.

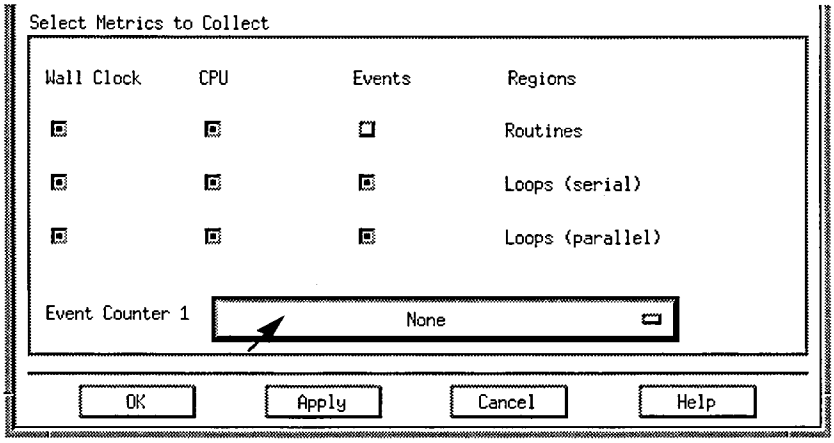
Selecting events (C4 and SPP Series only)

To specify the type of event you want to collect:

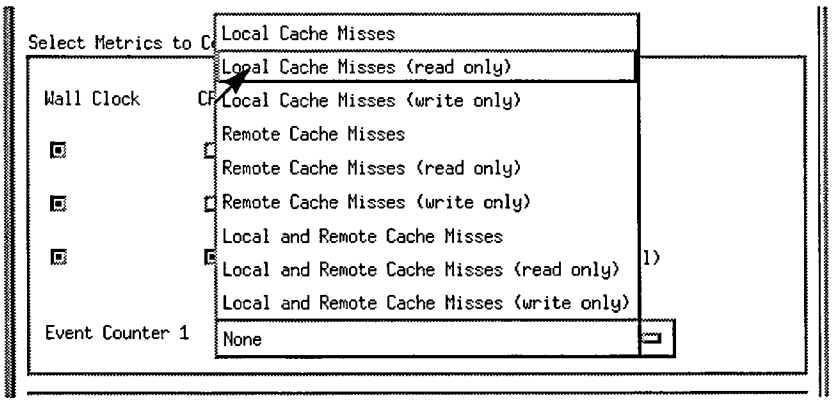
1. Choose Events as one of the metrics in the Select Metrics to Collect section of the Profile Selection dialog. The Event Counter options menu is now available for you to select an event type.

   For SPP Series machines, the default event selection is Local Cache Misses. For C4 Series machines, the default event selection is Data Cache Misses.

2. Click the left mouse button on the Event Counter options menu display a list of available event types. The types of events available vary between C4 and SPP Series machines, as shown in the following figure.

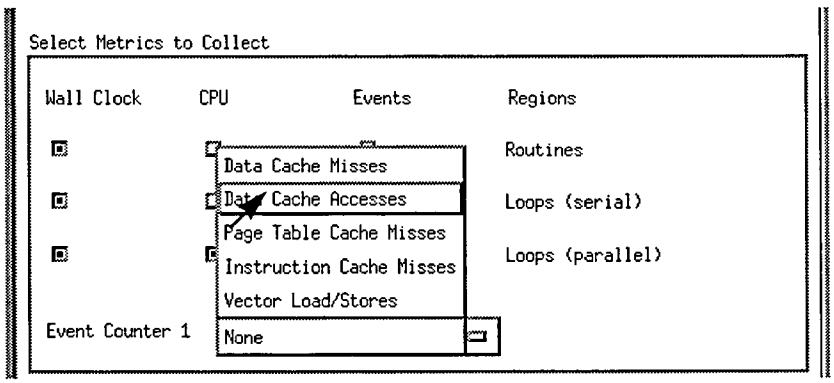


SPP Series events



or

C4 Series events



NOTE: Events metrics are only available on C4 and SPP computers. Because each architecture has a different set of events to choose from, the Event Counter options menu for C4 computers displays different options from the SPP computers.

3. Position the mouse cursor over the type of event you want to select. Only one event can be selected per run of your program.

   Refer to the section "Introducing metrics" for more information about the types of events that can be collected on each architecture.

4. Press the OK button to apply the changes and close the Profile Selection dialog.

Related Topics

Introducing metrics

Related Windows

Executable Manager window Profile Selection dialog

Selecting metrics in line mode

Once you have selected the regions in your program you want to profile, you must specify the types of metrics you want to collect at those regions.

In line mode, you specify the types of metrics you want to collect using the `collect` command and, if you have chosen to collect events, the `set events` command.

CPU time, wall clock time, and iteration/execution count metrics can be collected on all architectures. Other metrics that you can collect (including events and latency time) differ according to machine architecture.

This section describes the procedure for selecting types of metrics to collect when running CXpa in line mode. Refer to the following online help topics or sections of the *CXpa Reference* for more information

- "Introducing metrics" reference page
- Reference page for the `collect` command
- Reference page for the `set events` command

Choosing metrics to collect at the command line

Perform the following steps to select the types of metrics you want to collect:

1. Select the regions in your program that you want to profile using a form of the `select` command. For example:

   ```
   (CXpa) select routine all
   ```

   The above command tells CXpa to select all routine regions in your program for profiling.

NOTE: If you do not select one or more source code regions in your program for profiling with the `select` command, CXpa will not collect any metrics.

2. Enter a form of the `collect` command at the CXpa prompt. For example:

   ```
   (CXpa) collect cpu wall_clock events
   ```

   The above command tells CXpa that you want to collect CPU time, wall clock time, and events at the regions of your program selected for profiling.

   You can collect events metrics on C4 and SPP Series systems only. If you chose to collect events, you must use the `set events` command.

If you have selected `events` with the `collect` command, enter the command `set events` *<event-type>* at the CXpa prompt where *<event-type>* specifies the type of event you want to collect.

The events you can collect differ according to whether you are running your program on a C4 Series or SPP Series machine. Refer to the next section for a list of events available for each architecture.

For example:

```
(CXpa) set events local_misses
```

The above command executed on an SPP Series system tells CXpa that you want to count the number of times that data missed in the processor cache was found in memory on that processor's hypernode (for the profiled regions of your program). By default, events are collected during read and write operations. The `local_misses` parameter is only valid on SPP Series machines.

NOTE: Each use of the `set events` command overwrites its previous setting; only one type of event can be selected per program run.

3. Enter the `run` command at the CXpa prompt to collect the metrics you have specified for the selected regions of your program:

```
(CXpa) run
```

Event types

The types of events you can collect differ between C4 and SPP Series machines. Available events for each architecture are listed in the following sections.

SPP Series events

Use one of the following parameters to the `set events` command to select the type of event you want to collect. You can select only one of the following per program run:

- **`local_misses`**—Number of times that the profiled regions of your program had to access local memory (memory on that processor's hypernode) because of a processor cache miss.

NOTE: Remote memory miss events apply to multinode configurations only.

- **`remote_misses`**—Number of times the profiled regions of your program had to access remote memory (memory on another hypernode) because of a processor cache miss. This implies use of the CTI (Convex toroidal interconnect) rings. This event is valid for multinode configurations only.

- **local_and_remote_misses**—Number of shared local and remote memory misses combined for the profiled regions of your program. This event is valid for multinode configurations only.

You can use one or both of the following parameters of the `set events` command to specify whether the type of event you have selected is collected during read operations only, write operations only, or both:

- **read_only**—A read miss is a data cache miss caused by a load.
- **write_only**—A write miss is a data cache miss cause by a store or a data cache miss caused by a load and clear.

The default is `read_only write_only`.

C4 Series events

Use one of the following parameters with the `set events` command to select the type of event you want to collect. You can select only one of the following per program run:

- `data_misses`—Number of times that a memory reference request could not be met in the CPU data cache for the profiled regions of your program.
- `data_accesses`—Number of times that a memory reference request was met in the CPU data cache for the profiled regions of your program.
- `instruction_misses`—Number of times that an instruction reference request was not met in the CPU data cache for the profiled regions of your program.
- `page_table_misses`—Number of page table entry misses that occurred in the profiled regions of your program.
- `vector_load_stores`—Collects the number loads and stores to and from memory into and out of vector registers for the profiled regions of your program.

Related Topics

Introducing metrics

Related Commands

`collect`
`select`
`set events`

Part 2 Reference pages

Part 2 of this book contains reference pages for:

- CXpa's performance reports
- The windows in CXpa's X window interface
- The commands for CXpa's line mode interface
- CXpa messages

You can also access this information through the CXpa online help system.

Reports

4

This chapter provides an overview of the textual reports that CXpa displays for the profiled regions of your program.

CXpa can display textual performance reports for the following types of source code regions:

- Routines
- Loops
- Parallel regions (parallel loops)
- Basic blocks

The metrics available in performance reports vary according to machine type, source code regions selected for profiling, and the options used when compiling your program. Types of performance reports that are available for profiled regions are as follows:

- **Computation**—Includes iteration/execution counts and CPU time.
- **Time to solution**—Includes wall clock time and CPU/wall clock time.
- **Events**—Includes memory access events and latency metrics. Memory access events are collectable on C4 Series and SPP Series machines only; latency metrics are only available on SPP Series machines.
- **Vector Register Utilization**—Includes vector spills, vector flops, chime counts, and estimated Mflops. Vector register utilization metrics are only available on C Series machines.

For profiled loop regions (including parallel loops), performance data is provided for both original loops and optimized loops, along with information about the types of optimizations that were performed by the compiler.

For a detailed discussion of each type of report, refer to the "Routine reports," "Loop reports," "Parallel Region reports," and "Basic Block report" online help topics or sections in this book.

Filtering report data

This section describes how you can filter report data in line mode and X window mode.

NOTE: Profiling data for system libraries instrumented for CXpa can only be collected and viewed in reports if you compiled your source files with the `-cxpalib` option, and the CXpa-instrumented libraries are installed on your system.

X window mode

In X window mode, you can filter the data from instrumented system libraries displayed in CXpa reports by changing the application/library region visibility settings. Use the following procedure:

1. Select the Visibility option from the Options menu in the Executable Manager or Analysis Control window to bring up the Visibility Selection dialog.
2. Click the toggle buttons to select and/or deselect application and library region visibility.
3. Click OK to apply the changes and close the dialog.

Line mode

In line mode, you can filter report data with the `set visibility` command:

- `set visibility application`—Filter out performance data displayed in reports for system libraries instrumented for use with CXpa.
- `set visibility application library`—Include performance data displayed in reports for both application routines and system libraries instrumented with CXpa. This is the default.
- `set visibility library`—Display performance data in reports for CXpa-instrumented system library routines only.
- `set visibility thread`—Display performance data on a per-thread basis for all reports.
- `set visibility process`—Display performance data on a per-process basis for all reports. This is the default.

To view the current CXpa application/library visibility settings in line mode, use the `info` command.

Viewing reports

Textual performance reports are available in X window mode and line mode.

X window mode

In X window mode, you can select the types of reports you want to view from the Analysis Report window. To create an Analysis Report window:

- Select the Report option from the button at the lower right corner of the Executable Manager window
- Press the Create Report button in the Analysis Control window
- Select the Report option from the Windows menu in the Executable Manager window or the Analysis Control window

Once you have created an Analysis Report window, use the toggle buttons to select the region level for which you want to create reports, then select and/or deselect appropriate metrics.

You can also press the Customize button in the Analysis Report window to bring up a dialog that allows you to select specific routines and region levels for customized performance reports.

Line mode

In line mode, use a form of the `analyze` command to display performance reports (for example, `analyze`, `analyze loop`, or `analyze pregion`). The output of the `analyze` command is paged; you can also redirect output from this command to a file.

Report header

When you generate a report, CXpa prefaces the report with a header that contains the following information:

- **Executable**—Executable name for the program being profiled.
- **Performance Data**—Performance data file (PDF) name.
- **Process State**—State of the process when the PDF was created or closed. The states include: `running`, `paused`, `terminated`, `exited`, and `not started`.

- **CPU Time**—Total CPU time in seconds for your program.
- **Wall Clock Time**—Total Wall Clock Time in seconds for your program.
- **Architecture**—CONVEX architecture where the PDF was created.

The following is an example of a CXpa report header:

```
================================================================================
                        CXpa Version 3.0 Profile

Executable       : /texec/smith/fcApplications/fry/fry.-O2.exe
Performance Data: /toolmaster/devtools/work3/pdf/fry.pdf
Process State    : finished
CPU Time         : 25.160  secs
Wall Clock Time : 30.025  secs
Architecture     : CONVEX C3400 Series (3 cpus)
================================================================================
```

Related Topics

Basic Block report
Introducing metrics
Reports
Interpreting Mflops data
Parallel Region reports
Routine reports

Related Windows

Analysis Report window
Visibility Selection dialog
Customize Report dialog

Basic Block report

Description

The Basic Block report provides performance information for the profiled basic blocks in your program. A basic block is a section of code that has a single entry point and single exit point. One Basic Block Performance Analysis report is displayed for each executed routine with profiled basic blocks. This report can be used to identify routines which have a significant amount of code that is never executed.

CXpa can only generate this report if you use the `-cxpab` option to compile source files that contain basic blocks you wish to profile.

Refer to the "Fields" section of this reference page for information about fields (columns) that can appear in Basic Block reports.

Report

Metrics displayed in Basic Block reports are as follows:

- Total number of times each basic block was executed
- The starting address for each block (PC Value)
- Percentage and number of total blocks executed

You can use the execution metrics to determine unused blocks. The following example contains a Basic Block report.

Basic Block report

```
================================================================================
                        Basic Block Performance Analysis
                                    For: start
================================================================================

 Line            PC Value           Times  Exec                              PS
-------        ------------       ---------------                            --
     5          0x0004d408                      1
     5          0x0004d448                      1
     5          0x0004d370                      1
     5          0x0004d3b4                      1
     9          0x0004d510                      2
     9          0x0004d4a0                      1
    10          0x0004d61c                      1
    19          0x0004d670                     10
    22          0x0004d6c8                      4
    25          0x0004d730                      6
    29          0x0004d79c                     10
    34          0x0004d7ec                      1
    37          0x0004d838                      0
    40          0x0004d898                      1
    45          0x0004d934                      1
    45          0x0004d8f4                      1

Total Blocks          : 16
Total Blocks Executed : 15 (93.8%)

================================================================================
```

Context

To view a Basic Block report in X window mode, bring up the Analysis Report window, then:

- Select All as the region level

  or

- Press the Customize button on the Analysis Report window to bring up the Customize Report dialog, select Basic Blocks as the region type, and click OK or Apply.

To view a Basic Block report in line mode, use the `analyze block` command

Fields

Basic Block Reports are output on a per routine basis, for all profiled routines in your program that contain routines, with a report header similar to the following:

```
================================================================================
                    Basic Block Performance Analysis
                              For: start
================================================================================
```

Fields that can appear in basic block reports are listed and described below:

- `Line`—Starting source file line number for the basic block. The starting line number helps to trace a basic block back to the original source code.

  Due to optimizations, it is possible for many basic blocks to begin at the same line number.

- `PC Value`—Starting address of the basic block.
- `Times Exec`—Number of times the basic block was executed.
- `Total Blocks`—Total number of basic blocks in this routine.
- `Total Blocks Executed`—Total number of blocks that were executed and the percentage of blocks executed.
- `PS`—Not used.

Related Commands

`analyze`

Related Concepts

Introducing metrics
Loop reports
Parallel Region reports
Reports
Routine reports

Related windows

Analysis Report window
Customize Report dialog

Loop reports

Description

Loop performance analysis reports display metrics for profiled loop regions in your program.

Loop reports are only available if:

- The source files you wish to profile at the loop level are compiled at optimization level `-O1` or greater with the `-cxpa` option.
- Your program contains loops, and they were selected for profiling.
- At least one profiled loop region was executed.

Depending on the architecture on which you created the PDF and the type of metrics collected, the following reports can be displayed for each routine containing profiled loop regions that were executed:

- Computation
- Time to Solution
- Events (SPP and C4 Series only)
- Vector Register Utilization (C Series only). You must compile your source files at optimization level `-O2` or greater to obtain this report.

Each loop report, with the exception of the Vector Register Utilization report, consists of two sections:

- **Original Loops** —Displays metrics corresponding to the loops in your original source code.

  The Original Loops section provides you with a convenient way to relate metrics collected for the actual loops generated by the compiler to the loops in your original source code.

- **Optimized Loops**—Displays metrics for the actual loops generated by the compiler along with a brief history of the transformations performed on each loop by the compiler. The Optimized Loops section provides the most accurate picture of loop performance for your program.

Refer to the "Reports" section of this reference page for descriptions and explanations of the reports listed above, along with sample loop reports.

Refer to the "Fields" section of this reference page for information about fields (colums) that can appear in loop reports.

Refer to the "Reports" section of the *CXpa Reference* or the reference page for the `set visibility` command for filtering options for reports.

Reports

For loop regions, four types of reports are available: Computation, Time to Solution, Events (C4 Series and SPP Series only), and Vector Register Utilization (C Series only). The output of these reports are described in the following sections, along with sample reports.

Computation

The metrics in the Computation report can be used to examine the following for profiled loop regions:

- Number of times the loop was executed
- Number of iterations per invocation (total, minimum, maximum, and average)
- Amount of CPU time spent executing each loop:
  - Excluding time spent in inner loops
  - Including time spent in inner loops

For optimized loops, the following additional information is provided:

- Resulting nesting level of the loop after optimization
- The history of the transformations applied by the compiler (shown in the Optimization column) for optimized loops. Refer to the "Fields" section of this reference page for an explanation of the abbreviations shown in the Optimization column.

For a complete discussion of automatic optimizations performed by Convex compilers and manual optimization techniques for Convex machines, refer to the Convex *C Optimization Guide* (Order No. DSW-089), *Fortran Optimization Guide* (Order No. DSW-034), or the *Exemplar Programming Guide* (Order No. DSW- 014).

Line numbers for optimized loops annotated with a lowercase letter indicate that the loop was split into two or more loops during optimization.

Loop computation reports are displayed for each routine containing profiled loop regions that were executed. A sample loop Computation report is shown in the following example.

```
================================================================================
                        Loop Performance Analysis
                              For: init
================================================================================

Original Loops:
                                    Computation
                          Times       Iteration Count          (plus inner)
 Line                      Exec      Min      Max      Avg       CPU Time      PS
-------                   -------- ------- ------- -------  ----------------  --
    22                         512       0    1024     512             1.330
    23                           1       0     512     256             1.336

Optimized Loops:
                                    Computation
                          Times       Iteration Count  (less inner)(plus inner)
 Line    NL Optimization  Exec     Min      Max    Avg   CPU Time    CPU Time   PS
-------  -- ----------- -------- ------ ------ ------ ---------  ---------  --
    23a   0 P,I,Ds             1    512    512    512      5.868m      1.336
    22a   1 S,I,Ds           512   1024   1024   1024      1.330      1.330

    23b   0 S,I,Ds             0      0      0      0      0.000      0.000
    22b   1 S,I,Ds             0      0      0      0      0.000      0.000
```

Time to Solution

The metrics in the Time to Solution report for loop regions can be used to examine the following (assuming you have collected wall clock time):

- Total wall clock time spent in loops
- CPU/wall clock time

For serial loops, if the CPU/wall clock percentage is high, the CPU is being utilized efficiently in the region. If this percentage is low, it indicates that there is some type of performance bottleneck (assuming the data is significant in light of the amount of CPU work available). Performance bottlenecks can be caused by:

- I/O calls (for example, read() or write() calls)
- System calls (for example, open() or close() calls)
- Memory accesses (for example, cache misses). You can compare event metrics and latency for these loops to find out if the bottleneck is due to memory accesses.

For parallel loops, the CPU/wall clock time is the concurrency factor for the parallel loop. Parallel loops are annotated with a "P" in the Optimization column.

Values for CPU/wall clock time for parallel loops that are less than 1 indicate that for the current size of the problem (that is, the number of processors or the size of the data set), no real benefit was achieved from parallelism for that region; values that approach *n*, where *n* is the number of processors, indicate good use of resources within a parallel loop.

For example, as you increase the number of processors or the amount of work (data set size), the concurrency factor should increase proportionately. This would indicate that the parallel loop region is scaling well in parallel.

Time to Solution reports for loops are displayed for each routine containing profiled loop regions that were executed. The following example shows a Time to Solution report.

```
================================================================================
                          Loop Performance Analysis
                                  For: init
================================================================================
```

(... lines of output omitted)

```
Original Loops:
                                  Time to Solution
                          Times                              (plus inner)
 Line                      Exec                          Wall Clock  CPU/Wall PS
-------                   --------                       ----------  -------- --
    22                           4                          4.605m      0.32
    23                           1                          0.010       0.24

Optimized Loops:
                                  Time to Solution
                          Times          (less inner)          (plus inner)
 Line    NL Optimization  Exec    Wall Clock  CPU/Wall  Wall Clock  CPU/Wall PS
-------  -- ----------- -------- ----------  --------  ----------  -------- --
    23a   0 P,I,Ds              1       5.397m      0.18        0.010       0.24
    22a   1 S,I,P-Ur:4,Ds        4       4.605m      0.32        4.605m      0.32
    22b   1 S,I,Ds              0       0.000       0.00        0.000       N/A

    23b   0 S,I,Ds              0       0.000       0.00        0.000       N/A
    22c   1 S,I,P-Ur:4,Ds        0       0.000       0.00        0.000       N/A
    22d   1 S,I,Ds              0       0.000       0.00        0.000       N/A
```

Events (C4 and SPP Series only)

If you have chosen to collect events, CXpa displays an Events report for original and optimized loops in profiled loop regions. The Events report heading indicates the type of event collected for that run of your program (only one type of event can be collected per program run).

The types of events collected vary according to machine architecture. On SPP Series machines, CXpa collects data cache miss counts and latency time for accessing memory hierarchies. On C4 Series machines, CXpa collects instruction, data, and page table cache miss counts and vector load/store operation counts.

Refer to the *Exemplar Architecture* (Order No. DHW-014 and the *C Series Architecture* (Order No. DHW-300) reference manuals for more information about these architectures.

Events metrics for loop regions enable you to examine:

- Total number of events that occurred in a loop region.
- Latency—The total wall clock time spent accessing memory to satisfy a data cache miss (SPP Series only).
- Percentage of CPU time spent accessing memory to satisfy a data cache miss. This is expressed as event latency/CPU time, as shown in the `% CPU` column (SPP Series only).

  The % CPU value indicates how much of your computational work is being performed vs. the memory latency you are incurring in a source region for the active thread. If the percentage is low, it means that the CPU found all of the data it needed to do its job in the cache (close at hand). If the percentage is high, it means the CPU had to spend a lot of time asking the memory system to retrieve the data it needed relative to the amount of work it had to do.

  This percentage must be looked at in light of the amount of CPU work available to make a significant observation about program performance in light of cache effects. For example, if the event latency/CPU (% CPU) is 75%, and the overall CPU time is 25 secs for the region, then it is probably safe to say that cache contention is occurring. You could then try to search out the offending memory access pattern in the region and correct it. If the % CPU value is 75%, but the overall CPU time is only 25 usecs, then the data is probably not significant.
- Balance of work distributed among threads for parallel loops—By looking at latency, chore counts, and cache miss counts for parallel loops, you can see whether an imbalance of work across loops is due to cache effects. (Loops that have been parallelized by the compiler are with a "P" in the Optimization column.)

The following example contains an Events report for loop regions that was generated on an SPP Series machine. For this program run, local cache misses were collected for both read and write operations. Local cache miss counts indicate the number of times the program had to access memory local to a processor's node due to a processor cache miss.

```
================================================================================
                          Loop Performance Analysis
                                For: munge
================================================================================
...lines of output omitted
Original Loops:
                                    Local Cache Misses
                          Times                    (plus inner)
 Line                      Exec     Number of       Latency        % CPU      PS
-------                   -------- -------------  -------------  -----------  --
    35                           4           916         0.528m       18.93%
    36                           1          1330         0.802m       24.66%

Optimized Loops:
                                    Local Cache Misses
                          Times                    (less inner)
 Line    NL Optimization  Exec     Number of       Latency        % CPU      PS
-------  -- -----------  -------- -------------  -------------  -----------  --
    36a   0 P,I,Ds               1           414         0.274m       59.05%
    35a   1 S,I,P-Ur:4,Ds        4           916          0.528m       18.93%
    35b   1 S,I,Ds               0             0         0.000         0.00%

    36b   0 S,I,Ds               0             0         0.000         0.00%
    35c   1 S,I,P-Ur:4,Ds        0             0          0.000         0.00%
    35d   1 S,I,Ds               0             0         0.000         0.00%

                          Times                    (plus inner)
 Line    NL Optimization  Exec     Number of       Latency        % CPU      PS
-------  -- -----------  -------- -------------  -------------  -----------  --
    36a   0 P,I,Ds               1          1330         0.802m       24.66%
    35a   1 S,I,P-Ur:4,Ds        4           916          0.528m       18.93%
    35b   1 S,I,Ds               0             0         0.000          N/A

    36b   0 S,I,Ds               0             0         0.000          N/A
    35c   1 S,I,P-Ur:4,Ds        0             0          0.000          N/A

================================================================================
```

Vector Register Utilization (C Series only)

The Vector Register Utilization report is only available if the source files containing loop regions selected for profiling have been compiled at optimization level `-O2` or greater, and the loops contain vector floating point operations. The metrics displayed in this report include:

- **Vector Spills**—Number of times that a previous value in a vector register had to be written to memory.
- **Vector Flops**—Number of arithmetic vector floating point operations in the loop. This count does not include load or store operations, and assumes vector options are not masked.
- **Chime Count**—Number of chimes occurring within a given loop. Chime is an acronym for chained instruction measurement. The greater the chime count, the more resource conflicts, and therefore the less vector chaining.
- **Estimated Mflops**—Estimated millions of floating point operations per second. There are two subcategories:
  - **Avg**—Average number of Mflops per loop invocation. This is based on the average CPU time per invocation of the loop.
  - **Peak**—Theoretical maximum number of Mflops for this loop on this architecture.

For more information on the interpretation and reporting of vectorized loops and floating point operations, refer to the "Interpreting Mflops data" online help topic or section in this book.

The following example contains a Vector Register Utilization report for loop regions that was generated on a C4 Series machine with four CPUs.

```
================================================================================
                          Loop Performance Analysis
                                For: munge
================================================================================

(... lines of output omitted)

Optimized Loops:
                          Vector Register Utilization
                 Static Profile                 Estimated  Mflops
              Vector  Vector  Chime  (less inner loops)  (plus inner loops)
 Line     NL Spills  Flops   Count      Avg       Peak      Avg       Peak    PS
------- --- ------ ------- -------- --------- --------- --------- --------- --
    35    0      0       0      0.0     0.000     0.000     0.492   270.270
    36    1      0       3      1.5     0.571   270.270     0.571   270.270
```

Fields

Fields that can appear in Loop Performance Analysis reports are listed and described below in this section.

NOTE: All CXpa reports express time in seconds unless annotated with the letter "m" for milliseconds.

- `Line`—Source line number of the loop.
- `Times Exec`—Number of invocations of the loop.
- `Iteration Count`—Min, Max, and Avg iteration counts. The Iteration Count is shown for Min, Max, and Avg because the iteration count can vary from one invocation of a loop to another:
  - `Min`—Lists the fewest number of iterations of this loop for a single invocation.
  - `Max`—Lists the highest number of iterations of this loop for a single invocation.
  - `Avg`—Lists the average number of iterations of this loop for a single invocation.
- `CPU Time (plus inner loops)`—Total CPU time for the loop, including time spent in inner loops.
- `CPU Time (less inner loops)`—Total CPU time for the loop, excluding time spent in inner loops.
- `CPU/Wall`—Ratio of CPU time to wall clock time.
- `NL`—Nesting level of the loop after optimization.
- `Optimization`—Abbreviation for the transformation(s) that the compiler performed on the loop. The abbreviations possible are:
  - `Blk:`*n*—Loop blocking; *n* is the blocking factor, which is the number of iterations that were blocked together.
  - `D`—Distributed
  - `Ds`—Dynamic selection
  - `Hs`—Hoisted
  - `I`—Interchanged
  - `P`—Parallel
  - `PS`—Parallel strip-mined
  - `P/V`—Parallel vectorized
  - `pV`—Partially vectorized
  - `S`—Scalar
  - `SM`—Strip-mined vector
  - `UL`—Uninstrumentable
  - `UR:`*n*—Loop unrolling; *n* is the loop unrolling factor, which is the number of iterations that were unrolled.

- V—Fully vectorized

  For more information about automatic optimizations performed by the compiler and manual optimization techniques for Convex machines, refer to the *Exemplar Programming Guide*, the *C Optimization Guide*, or the *Fortran Optimization Guide*

- PS—Lists the profiling status. If this column is blank, then the loop executed normally. Other possible profiling statuses are:

e	The program exited at this point.
g	This region could not be timed due to the granularity of the clock supported on the current architecture.
m	Invalid time management was detected for this region due to an unprofilable code construct, an unprofilable command such as an `exec` or `fork`, or incorrect instrumentation in your program or a library routine. To work around incorrect instrumentation, you can: • Deselect the regions in your routines that are listed with a profiling status of `m`. • Choose not to profile library routines if a library routine is listed with a profiling status of `m`. By default, library routines are not profiled, so this will only occur if you used the `-cxpalib` compiler option to include library routines.
p	The program was paused at this point, and the timing information is incomplete.
t	The program terminated at this point.
u	This region was uninstrumentable because it was too small to gather timing data, was optimized to a single instruction, or contains an unrecognized construct.
x	CPU time (excluding called routines and/or inner loops) and ratio(s) cannot be computed accurately.
y	Wall clock time (excluding called routines and/or inner loops) and ratio(s) cannot be computed accurately.

`z`	Latency time (excluding called routines and/or inner loops) and ratio(s) cannot be computed accurately.

- `Latency`—The total wall clock time spent accessing memory to satisfy a data cache miss (SPP Series only).
- `% CPU`—Percentage of CPU time spent accessing memory to satisfy a data cache miss, expressed as event latency/CPU time (SPP Series only).
- `Vector Spills`—Lists the number of times that a previous value in a vector register had to be written to memory (C Series only).
- `Vector Flops`—Lists the number of arithmetic vector floating point operations in the loop. This count does not include load or store operations, and assumes vector options are not masked (C Series only).
- `Chime Count`—Lists the number of chimes occurring within a given loop. Chime is an acronym for chained instruction measurement. The greater the chime count, the more resource conflicts, and therefore the less vector chaining (C Series only).
- `Estimated Mflops`—Lists the estimated millions of floating point operations per second (C Series only). There are two subcategories:
  - `Avg`—Lists the average number of Mflops per loop invocation. This is based on the average CPU time per invocation of the loop.
  - `Peak`—Lists the theoretical maximum number of Mflops for this loop on this architecture.

Context

To display Loop reports in X window mode from the Analysis Report window, select Loops as the region level.

To display Loop reports in line mode, use the `analyze loop` command.

Related Commands

`analyze`

Related Topics

- Basic Block report
- Introducing metrics
- Reports
- Interpreting Mflops data
- Parallel Region reports
- Routine reports

Related Windows

- Analysis Report window
- Customize Report dialog

C Series only

Interpreting Mflops data

The Vector Register Utilization report contains various metrics for vectorized loops, such as the chime count and estimated millions of floating point operations per second (Mflops).

This reference page introduces the metrics included in the report and explains how these metrics are calculated.

On CONVEX C Series machines, performance data for vectorized loops is based solely on *vector* floating point operations.

The information in the Vector Register Utilization report represents the theoretical best that your code can do on a given machine, and is an estimate of how the code actually performed.

The first section briefly introduces the Vector Register Utilization report and vector chaining. The remaining sections explain how CXpa calculates chime counts and vector Mflops.

The topics covered include:

- Vector Register Utilization report
- Vector chaining
- Chime counts
- Vector Mflops

Vector Register Utilization report

The Vector Register Utilization report displays metrics and performance measures for vectorized loops.

This loop performance analysis report only appears for loops compiled at optimization level `-O2` or higher that have been vectorized by the compiler. It only contains an Optimized Loop section; no Original Loops section is available. Mflops data is only generated for loops containing vector floating point operations.

If the Vector Register Utilization report does not appear, no vectorization was done by the compiler for that loop or nest of loops.

The following example contains a Vector Register Utilization report for loop regions that was generated on a C4 Series machine with four CPUs.

```
Optimized Loops:
                                Vector Register Utilization
                      Static Profile                 Estimated  Mflops
                 Vector  Vector  Chime  (less inner loops)  (plus inner loops)
 Line     NL Spills  Flops   Count     Avg       Peak      Avg       Peak    PS
------- --- ------ ------- ------- --------- --------- --------- --------- --
    47a   0      0       0     0.0     0.000     0.000     0.000     0.000
    48a   1      0       0     0.0     0.000     0.000     0.000     0.000

    47b   0      0       0     0.0     0.000     0.000     0.414    50.000
    48b   1      0       0     0.0     0.000     0.000     0.414    50.000
    50    2      0       2     1.0     0.419    50.000     0.419    50.000
```

The fields in the Optimized loop section are briefly discussed below:

- `Line`—Starting source line number of the loop.
- `NL`—Nesting level of the loop after optimization.
- `Vector Spills`—Number of times that a previous value in a vector register had to be restored.
- `Vector Flops`—Number of arithmetic vector floating point instructions in the loop. This count does *not* include vector load or store operations.
- `Chime Count`—A chime is an acronym for CHained vector Instruction MEasurement. As CXpa calculates chimes, this number represents a "vector chaining factor" for a given loop that takes into account the rate at which the operation is performed on a given architecture (which is not always 1 result per clock cycle). This produces a more accurate measure of the efficiency of vector chaining than a simple chime count, because most C Series machines are capable of producing 2 results per cycle for most single-precision operations (two exceptions are square root and divide operations). The greater the chime count that CXpa reports, the more contention for available vector functional units, and therefore the less vector chaining.

  Vector chaining, chimes, and functional units are explained in the following sections.

- `Estimated Mflops`—Estimated number of millions of vector floating point operations per second. The `(less inner loops)` columns do not include time spent in inner loops. The `(plus inner loops)` columns include time spent in inner loops. There are two subcategories:
  - `Avg`—Average number of Mflops per loop invocation. This is based on the average CPU time per invocation of the loop.
  - `Peak`—A theoretical maximum number of Mflops for this loop based on the instructions generated by the compiler on this architecture.
- `PS`—Profiling status.

Dividing Avg Mflops by Peak Mflops gives the percent utilization of the vector unit. However, full utilization of the vector unit does not imply well-written code or fastest possible time to solution. It simply represents the best the existing code can do on that machine.

Vector chaining

This section explains how vector chaining occurs and describes two situations that prevent vector chaining: functional unit reservation and register bank conflicts. This information will make it easier to understand how CXpa calculates chimes and how to utilize the vector units more effectively.

The CONVEX C Series architecture enables certain vector operations to be performed concurrently, or to be overlapped with other vector operations. This ability, known collectively as vector chaining, can enable vector operations to occur simultaneously, rather than one after another.

This concurrency offers significant performance improvement. The chime count for a loop is a theoretical measure of the efficiency of vector chaining. The efficiency of vector chaining also depends on the precision of the operands, as well as architecture-specific factors such as the number of functional units and the operations that can be performed by each unit.

Vector chaining and functional units

On a C Series machine, the vector processor enables the compiler to load 128 elements of an array into a vector register, and then operate on all 128 elements as a unit. Because of the speed at which vector register operations are performed, such vector operations are often assumed to be happening to all 128 elements simultaneously.

A vector functional unit operates on one element of a vector register at a time. A vector functional unit is an independent part of a CPU that performs a designated operation or set of operations, such as addition or multiplication.

Because of this, the first element from the result of a vector operation is ready before the second, which is ready before the third, and so on.

Vector chaining allows the output of one functional unit to be used as the input to the next functional unit before the entire vector has been processed.

Consider the following two vector instructions. For this first example, assume that the instructions are executed on a C200 Series machine:

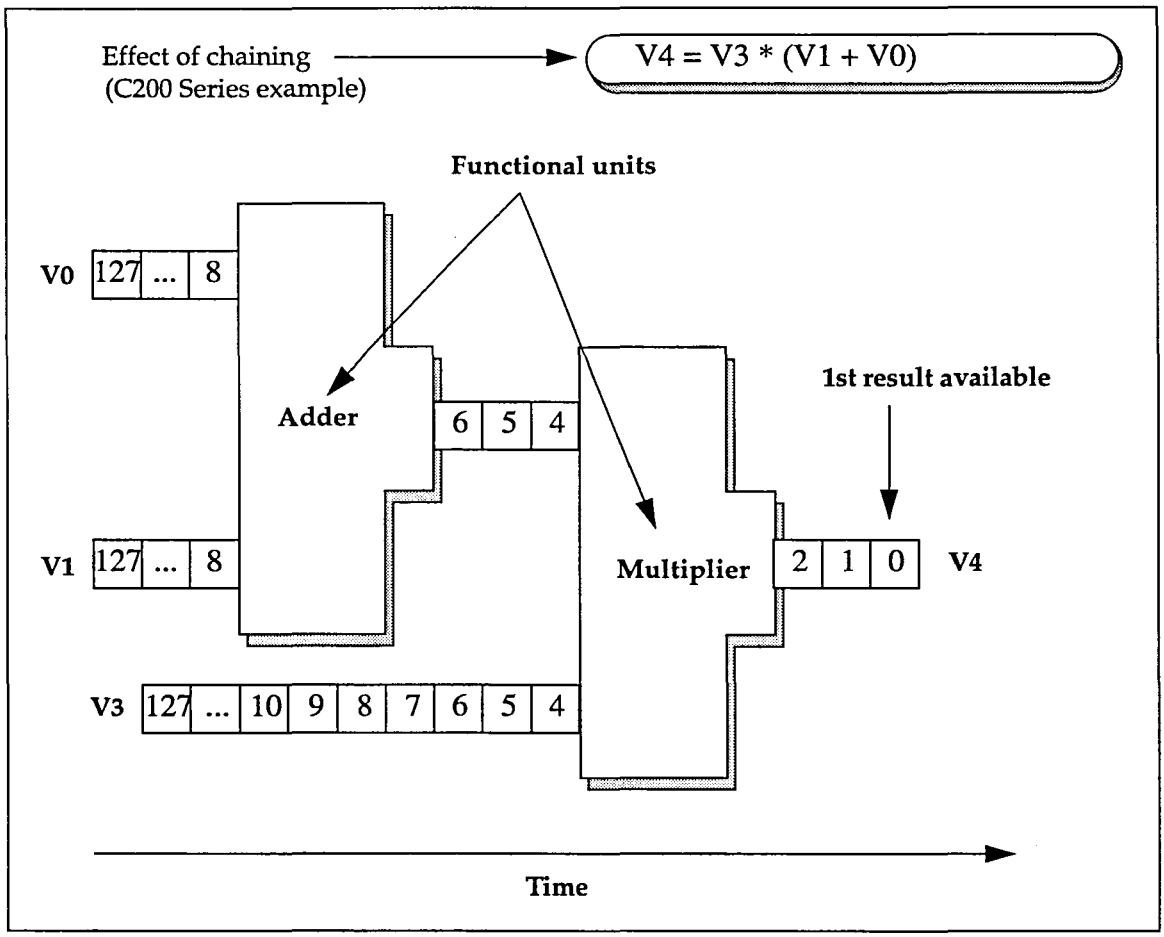
```
V2 = V1 + V0
V4 = V3 * V2
```

In this example, each element of V0 is added to the corresponding element of V1, and the result is stored into the corresponding element of V2. Then each element of V2 is multiplied with the corresponding element of V3, and the result is stored in V4. Without vector chaining, all of the elements of the first operation (the addition) would have to be completed before the multiplication operation could begin.

With vector chaining, the output of the addition operation can be fed directly into the multiplication operation. This is possible because the add operation and the multiply operation are calculated by two different functional units. This could be represented as:

```
V4 = V3 * (V1 + V0)
```

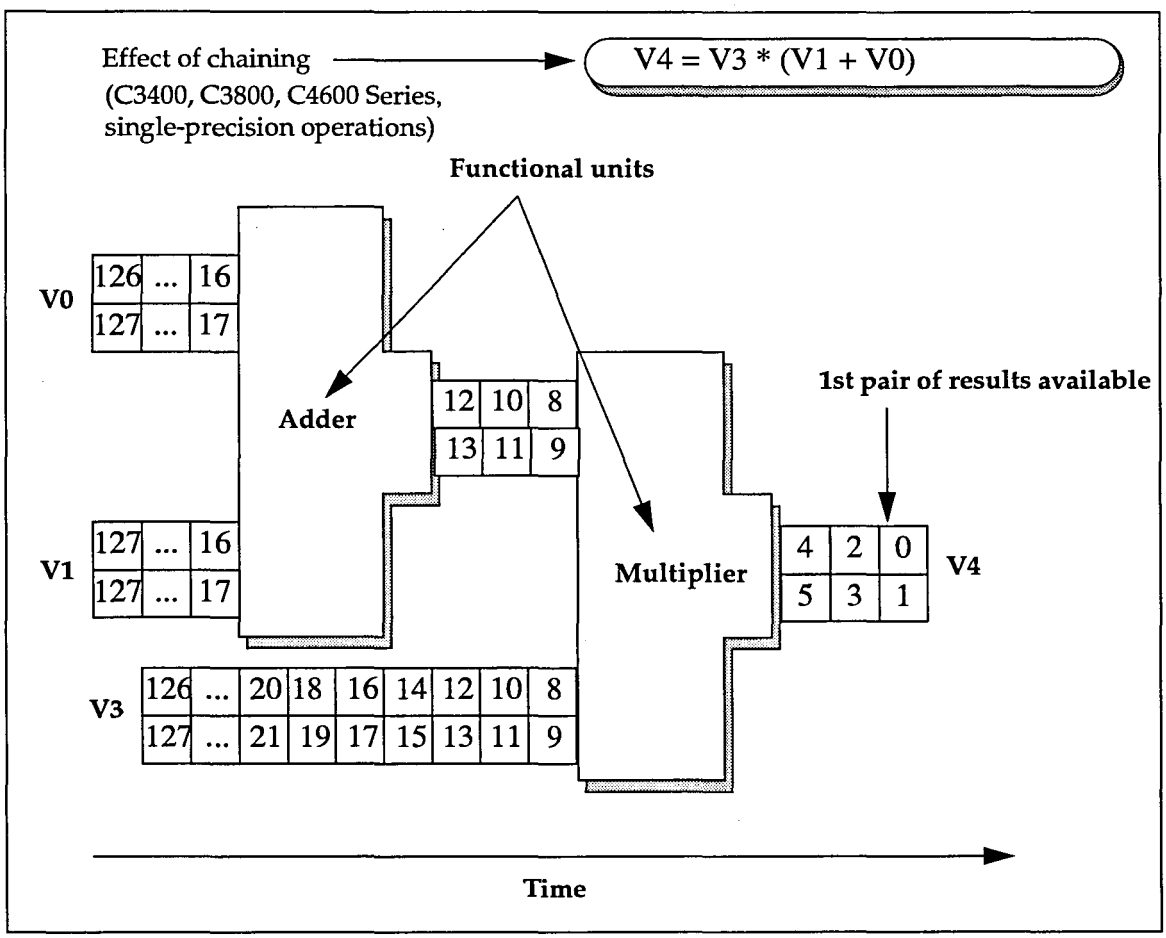
The following figure illustrates the effects of vector chaining for the vector instruction V4 = V3 * (V1 + V0), assuming that the code fragment is executing on a C200 Series machine.



As shown in the above figure, V0 and V1 are both passed to the Adder functional unit. As the results are output, they are fed directly into the Multiplier functional unit along with V3. As the results are generated, they are stored into V4.

If the values in V4 were to be stored into an array, this store could be chained with the previous two instructions. On the CONVEX C Series architecture, loading and storing of vector registers is handled by a third functional unit, called the Loader.

On some C Series machines, the efficiency of vector chaining can also be affected by the precision of the operation. C3400, C3800, and C4600 Series machines are capable of producing 2 results per clock cycle for operations in which both operands are less than 32 bits wide. The following figure shows how the vector instruction V4=V3 * (V1+V0) executes on a C3400, C3800, or C4600 Series machine when all of the elements are single-precision (word). Because each element is less than or equal to 32 bits, pairs of elements can be operated on simultaneously. If the statement shown in the example is the only one executed in a vector loop, CXpa reports the chime count as 0.5.



Functional unit reservation

Vector chaining is only possible when a functional unit is available. CONVEX C2 and C3 Series machines have three vector functional units per CPU; C4 Series machines have four. Each functional unit has a specific set of vector instructions it is capable of processing.

When a vector instruction cannot be chained with the previous instruction because the appropriate functional unit is not available, functional unit reservation occurs, and chaining is prohibited.

On the C200 and C3200 architectures, the following vector instructions could not be chained:

```
V2 = V1 + V1
V4 = V3 + V2
```

For these architectures, the add operation can only be performed by the Adder functional unit. The first add operation occupies the Adder functional unit. The second add operation must wait until the Adder is again available before beginning.

The ability of the CPU to chain vector instructions together depends on what vector operations can be performed by each vector functional unit of the CPU.

Vector functional units and available operations vary among C2, C3, and C4 Series machines. These are described in detail in the following sections.

C2 and C3 Series vector functional units

The vector functional units found in the CONVEX C2 and C3 Series architectures are listed in the following table. Names (Adder, Multiplier, Divider, or Loader) have been given to the functional units to identify the types of vector operations they perform. The table marks with a check each vector operation a vector functional unit can perform. You can use this table to determine whether functional unit reservation could be prevented by different instruction scheduling.

Generic operation	C200 and C3200 Vector func. units			C3400 Vector func. units			C3800 Vector func. units		
	Adder	Multiplier	Loader	Adder	Multiplier	Loader	Divider	Multiplier	Loader
Add/subtract	✔			✔	✔		✔	✔	
Logical compare	✔			✔	✔		✔	✔	
Bit count	✔			✔	✔		✔	✔	
Shift	✔			✔	✔		✔	✔	
Multiply		✔			✔			✔	
Divide		✔			✔		✔		
Type conversion	✔				✔		✔	✔	
Edit		✔		✔	✔		✔	✔	
Load/store			✔			✔			✔
Square root		✔			✔		✔		

C4 Series vector functional units

The vector functional units found in the CONVEX C4 Series architecture are listed in the following table. On C4 Series architectures, there are four vector functional units, which are referred to in the table as A, B, C, and the Loader. The A and B functional units perform all operations except divide, edit, and square root. Divide, edit, and square root operations can only be performed by functional unit C. Operations other than divide, square root, and edit are passed first to vector functional unit A, if it is available, followed by B, then C.

Note that three additional operations are supported: dot product, AXPY, and XYPA.

The table marks with a check each vector operation a vector functional unit can perform. You can use this table to determine whether functional unit reservation could be prevented by different instruction scheduling.

Generic operation	C4 Series vector functional units			
	A	B	C	Loader
Add/subtract	✔	✔	✔	
Logical compare	✔	✔	✔	
Bit count	✔	✔	✔	
Shift	✔	✔	✔	
Multiply	✔	✔	✔	
Type conversion	✔	✔	✔	
Dot product	✔	✔	✔	
AXPY/XYPA	✔	✔	✔	
Divide			✔	
Square root			✔	
Edit			✔	
Load/store				✔

Redundant operations and vector functional units

If an operation can be performed on more than one functional unit, it is said to be redundant. For example, the add operation is redundant on C3400 and C3800 Series CPUs.

On the C3400 Series CPU, redundant operations are passed to the Adder functional unit, because it does not perform any unique operations (a subsequent add could be passed to the Multiplier).

On the C3800 Series CPU, redundant operations are passed to the Divider functional unit, if it is available. Because divide operations are less common than multiply operations, it is less likely that the Divider functional unit will be needed by the next vector operation.

On a C4 Series CPU, all operations except divide, edit, square root, and load/store can be performed on more than one functional unit. Only vector functional unit C (divide, edit, and square root) and the Loader (load/store) perform unique operations. Redundant operations are passed first to vector functional unit A, if it is available, followed by B, then C.

Register bank conflicts

Vector chaining is prevented when a vector instruction tries to access a vector register that is not available. This is known as a register bank conflict. Also, vector instructions cannot be chained together if doing so would cause more reads or writes than possible to a single register bank.

C2 and C3 Series register banks

On C2 and C3 Series machines, each vector register bank holds a pair of registers. The pairs are as follows:

- V0 and V4
- V1 and V5
- V2 and V6
- V3 and V7

A maximum of two reads and one write can be made to a vector register bank during a single clock cycle.

C4 Series register banks

On C4 Series machines, each vector register bank contains four registers, which are grouped as follows:

- V0, V4, V8, and V12
- V1, V5, V9, and V13

- V2, V6, V10, and V14
- V3, V7, V11, and V15

A maximum of three reads and one write can be made during a single clock cycle.

On C3400, C3800, and C4 Series CPUs, two reads to the same register in a bank count as a single read.

For example, the following two instructions cause register bank reservation on all CONVEX C Series architectures:

```
V0 = V2 + V1
V4 = V3 * V1
```

The first instruction requires one write to the V0/V4 register bank. The second instruction cannot be chained with the first because of the additional write to the V0/V4 register bank.

Chimes

A loop whose vector instructions are chained together executes more quickly because chaining enables multiple instructions to occur simultaneously.

A CHained Instruction MEasurement, or chime, represents a vector instruction that could not be chained with the previous instructions, due to functional unit reservation or register bank reservation. Every vectorized loop will have a minimum of one chime.

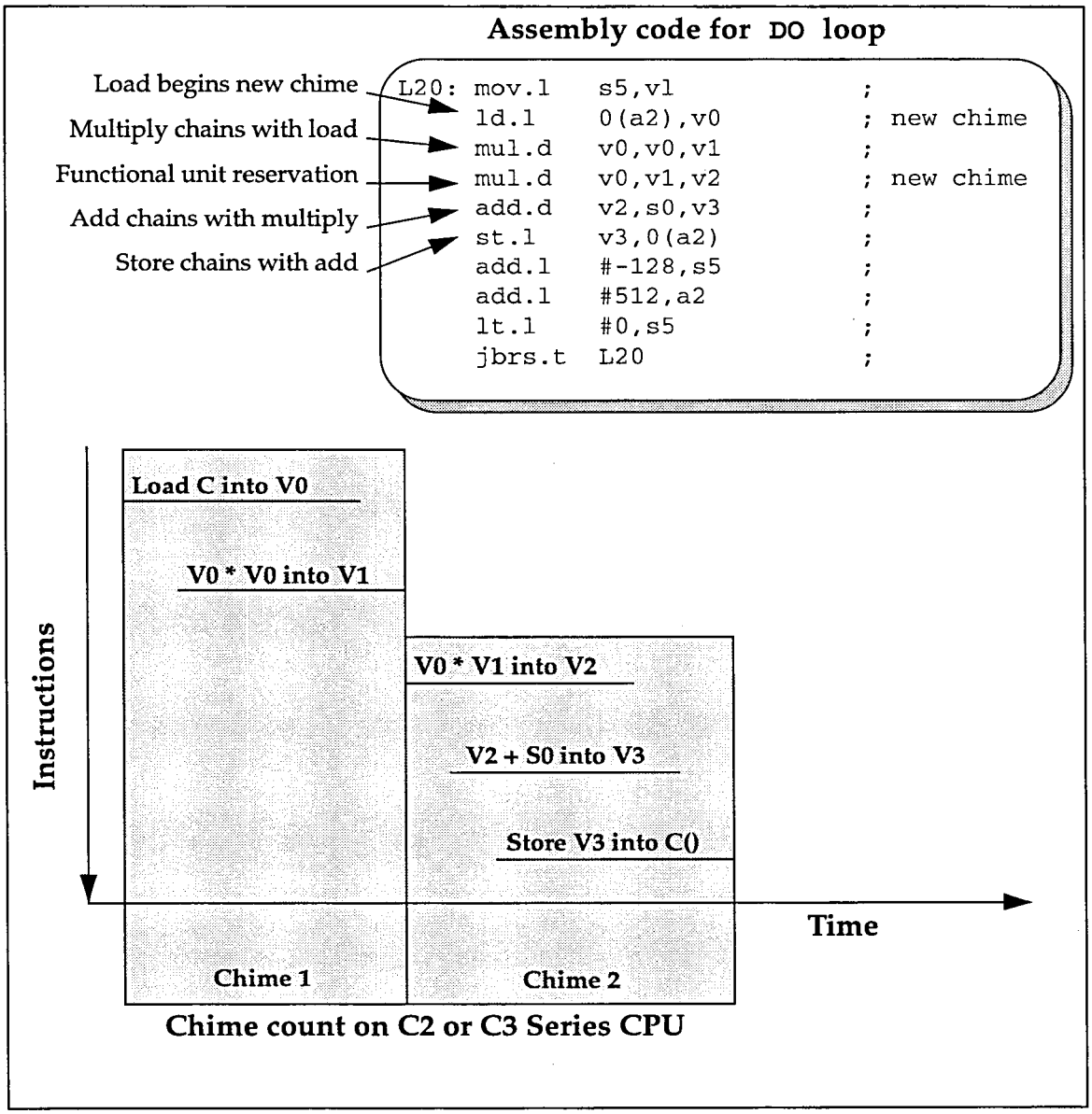
Ideally, each vector instruction would be chained together, thus resulting in a single chime. However, this is seldom possible. The higher the chime count for a loop, the less vector chaining achieved. Therefore, the number of chimes in a loop, called the chime count, can be an effective performance indicator for vectorized loops.

Calculating chimes

To demonstrate the calculation of a loop's chime count, consider the following FORTRAN loop:

```
DO I = 1, 256
   C(I) = C(I) * C(I) * C(I) + 3.0
ENDDO
```

The chime count for this loop when executed on a C2 or C3 Series machine is 2, as shown in the following figure.



In the above figure, the first multiply instruction (`mul.d v0,v0,v1`) can chain with the load instruction. On a C2 or C3 Series CPU, the second multiply cannot chain with the first because the multiply functional unit is not available. The second multiply cannot begin until the first multiply completes and the multiply functional unit is again available. Thus, the chime count is 2.

The chime count for the same set of instructions executing on a C4600 Series CPU is also 2, but the second chime would begin with the store instruction. The two multiply operations can chain, because on a C4600, the multiply operation is redundant. However, the store could not chain with the add, because the loader functional unit is busy.

Vector Mflops

Most scientific and engineering applications use floating point numbers. Due to differences in hardware and software, the performance of such code is better represented by millions of floating point operations performed per second (megaflops or Mflops) than by millions of instructions per second (Mips).

A floating point operation is simply an arithmetic instruction operating on one or more floating point numbers.

CXpa distinguishes two types of floating point operations:

- **Scalar**—Operating on a single floating point number. For example, taking the square root of 3.14.
- **Vector**—Operating on a vector of floating point numbers. For example, subtracting 3.14 from the first 128 elements of an array of real numbers.

CXpa reports both the estimated vector Mflops and the peak vector Mflops. These values are reported for the loop with and without nested loops.

The estimated number of vector Mflops indicates how well the loop ran on that machine. The closer the estimated Mflops is to the peak, the greater the utilization of the vector functional units available on the machine.

Calculating estimated Mflops

The estimated flops for a loop is calculated by dividing the total number of executed vector floating point operations in the loop by the total number of seconds spent in the loop.

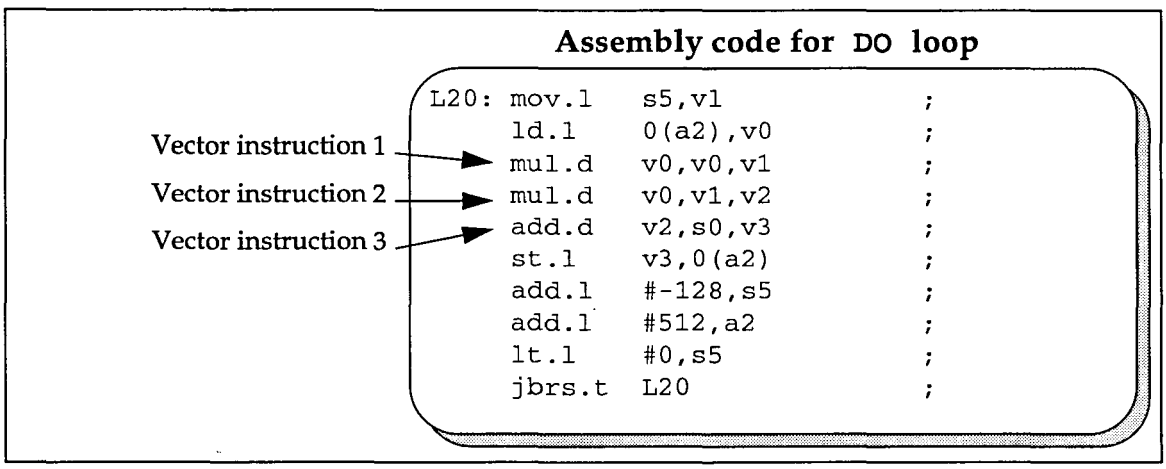
The estimated flops for a loop is calculated as:

$$estimated\_flops = \frac{floating\_point\_operations(flop) \times trip\_count}{total\_time(sec)}$$

Returning to the previous example, you can calculate the estimated Mflops for the loop:

```
DO I = 1, 256
   C(I) = C(I) * C(I) * C(I) + 3.0
ENDDO
```

There are 256 iterations of the loop, so the trip count is 256. You can determine the number of vector floating point operations in the loop by looking at the assembly code. As shown below, there are three vector floating point instructions inside the loop.



NOTE: The vector floating point instruction count does not include load/store operations, nor does it subtract for masked components (that is, assumes no mask).

Assume that the loop took .000026 seconds to execute on a C200. The calculation becomes:

$$estimated\_flops = \frac{3\ \text{flop} \times 256}{0.000026\ \text{sec}} = 29538462\ \text{flop/sec}$$

$$estimated\_Mflops = 29.538$$

The estimated Mflops for this loop would be 29.538 Mflops.

Calculating peak Mflops

Peak Mflops represents the maximum floating-point performance for the for a particular loop (based on the instructions generated by the compiler) on a particular machine. To calculate the peak Mflops, use the following equation:

$$peak\_flops = \frac{floating\_pt\_ops}{chime\_count} \times \frac{1}{machine\_clock\_cycle(sec)}$$

The clock rate of the machine rate per head is as follows:

- C200—40 ns
- C3400—40 ns
- C3800—16.67 ns
- C4600—7.4 ns

For the loop used in the previous example (executing on a C200), the peak rate becomes:

$$peak\_flops = \frac{3 \text{ flops}}{2} \times \frac{1}{40 \times 10^{-9} \text{ sec}}$$

$$peak\_Mflops = 37.5$$

On a C200, the peak Mflops for this loop (based on the instructions generated by the compiler) is 37.5 Mflops.

Related Topics

Loop reports
Parallel Region reports
Reports
Routine reports

Related Windows

Analysis Report window
Customize Report dialog

Parallel Region reports

Description

Parallel region performance analysis reports display metrics for profiled parallel loop regions in your program.

Parallel region reports are only available if the source files containing regions you wish to profile at the parallel region level are compiled at optimization level `-O3` with the `-cxpa` option. At optimization levels other than `-O3`, the compiler does not parallelize loops.

Depending on the architecture on which you created the PDF and the type of metrics collected, the following reports can be displayed for each routine executed in your program that contains profiled parallel regions:

- Time to Solution
- Events (SPP Series and C4 Series only)

Each report contain two sections:

- An Original Loops section, which displays metrics on per process. By summing the metrics for all threads of the process, CXpa provides a view of the metrics collected at runtime that corresponds to your original source code.
- An Optimized Loops section, which displays metrics for each thread that executed in the parallel region.

Refer to the "Fields" section of this reference page for information about fields (columns) that can appear in parallel region reports.

Reports

For parallel regions, two types of reports are available: Time to Solution and Events. The output of these reports are described in the following sections, along with sample reports.

Time to Solution

The metrics in the Time to Solution report for parallel regions can be used to examine the following:

- Total wall clock time spent in parallel regions
- Concurrency factor for parallel regions, expressed as the ratio of CPU time to wall clock time.

Values for CPU/wall clock time that are less than 1 indicate that for the current size of the problem (that is, the number of processors or the size of the data set), no real benefit was achieved from parallelism for that region; values that approach *n*, where *n* is the number of processors, indicate good use of resources within a parallel region.

For example, as you increase the number of processors or the amount of work (data set size), the concurrency factor should increase proportionately. This would indicate that the region is scaling well in parallel.

- Distribution of work across threads—By looking at CPU time, wall clock time, and/or chore counts, you can spot parallel regions where the load does not appear to be balanced across threads. You can also examine event metrics for those regions to see whether performance bottlenecks are due to cache effects.

A sample time-to-solution report for a parallel region is shown below:

```
================================================================================
                    Parallel Region Performance Analysis
                                   For: init
================================================================================

Original Loops:
                                Time to Solution
 Line     Times Exec        CPU         Wall Clock       CPU/Wall                PS
-------  ----------  -------------  -------------  -------------                --
    23            1          1.336          1.501           0.89

Optimized Loops:
                                Time to Solution
 Line     Thread Id         CPU         Wall Clock        Chores                 PS
-------  ----------  -------------  -------------  -------------                ---
    23             0          0.317          1.363             1
                   1          0.353          1.055             1
                   2          0.315          1.497             1
                   3          0.351          1.019             1
================================================================================
```

Events (C4 and SPP Series only)

If you have chosen to collect events, CXpa displays an Events report for original and optimized loops in profiled parallel regions. The Events report heading indicates the type of event collected for that run of your program (only one type of event can be collected per program run).

The types of events collected vary according to machine architecture. On SPP Series machines, CXpa collects data cache miss counts and latency time for accessing memory hierarchies. On C4 Series machines, CXpa collects instruction, data, and page table cache miss counts and vector load/store operation counts.

Refer to the *Exemplar Architecture* (Order No. DHW-014) and the *C Series Architecture* (Order No. DHW-300) reference manuals for more information about these architectures.

Events metrics for parallel regions enable you to examine:

- Total number of events that occurred in a parallel region or regions.
- Latency—The total wall clock time spent accessing memory to satisfy a data cache miss (SPP Series only).
- Percentage of CPU time spent accessing memory to satisfy a data cache miss. This is expressed as event latency/CPU time, as shown in the `% CPU` column (SPP Series only).

  The % CPU value indicates how much of your computational work is being performed vs. the memory latency you are incurring in a source region for the active thread. If the percentage is low, it means that the CPU found all of the data it needed to do its job in the cache (close at hand). If the percentage is high, it means the CPU had to spend a lot of time asking the memory system to retrieve the data it needed relative to the amount of work it had to do.

  This percentage must be looked at in light of the amount of CPU work available to make a significant observation about program performance in light of cache effects. For example, if the event latency/CPU (% CPU) is 75%, and the overall CPU time is 25 secs for the region, then it is probably safe to say that cache contention is occurring. You could then try to search out the offending memory access pattern in the region and correct it. If the event latency/CPU (% CPU) is 75%, but the overall CPU time is only 25 usecs, then the data is probably not significant.
- Balance of work distributed among threads—By looking at latency, chore counts, and cache miss counts for events across threads, you can see whether an imbalance of work across threads is due to cache effects.

A sample Events report (generated on an SPP Series machine) is shown below. For this program run, local cache misses were collected for both read and write operations. Local cache miss counts indicate the number of times the program had to access memory local to a processor's node due to a processor cache miss.

```
================================================================================
                    Parallel Region Performance Analysis
                                For: munge
================================================================================
...lines of output omitted
Original Loops:
                          Local Cache Miss Events
 Line     Times Exec     Number of      Latency        % CPU                   PS
-------  ----------  -------------  -------------  --------                   --
    36            1         394634          0.196      14.2%

Optimized Loops:
                          Local Cache Miss Events
 Line     Thread Id      Number of       Latency        Chores       % CPU     PS
-------  ----------  -------------  -------------  -------------  --------  --
    36             0         394419          0.196              1     56.6%
                   1         394504          0.192              1     56.2%
                   2         391414          0.194              1     56.4%
                   3         389436          0.201              1     57.2%
================================================================================
```

Fields

Parallel Region reports are output by default on a per-routine basis, for all profiled routines in your program that contain parallel regions, with a report header similar to the following:

```
================================================================================
                    Parallel Region Performance Analysis
                                For: calc
================================================================================
```

Fields that can appear in Parallel Region reports are listed and described below:

- `Line`—Starting source file line number for the parallel region. Typically, this is the line number in your source code that corresponds to the starting point for the parallel loop.
- `Times Exec`—Number of times the parallel region was executed.

NOTE: All CXpa reports express time in seconds unless the time is annotated with the letter "m" for milliseconds.

- `CPU`—For original loops, this is the total CPU time spent for all threads executing the parallel region. For optimized loops, this is the total CPU time for each thread executing the parallel region.
- `Wall Clock`—For original loops, this is the time to solution for all threads executing the parallel region. For optimized loops this is the time to solution for each thread executing the parallel region. Wall clock time includes time when the process is idle.
- `CPU/Wall`—Concurrency factor for the parallel region. The concurrency factor is the wall clock time divided by the CPU time and indicates the benefit achieved through parallelism vs. serial execution for the region.
- `Thread ID`—Thread ID number for each thread executing the parallel region.
- `Latency`—The total wall clock time spent accessing memory to satisfy a data cache miss (SPP Series only).
- `% CPU`—Percentage of CPU time spent accessing memory to satisfy a data cache miss, expressed as event latency/CPU time (SPP Series only).
- `Chore Count`—Lists the total number of chores executed by the thread executing the parallel region.

  A chore is one unit of work performed by a single thread in a parallel region. This unit of work usually corresponds to a single iteration of the body of a parallelized loop.

  On C Series computers, you can invoke CXpa with the `-f` option to run CXpa in fixed scheduling mode, reserving all processors for its use. Running CXpa with fixed scheduling produces a more accurate chore count and gives a better indication of a program's parallel efficiency.
- `PS`—Displays the profiling status. If this column is blank, then the region executed normally. Other possible profiling statuses are as follows:

Profiling status	Meaning
`e`	The program exited at this point.
`g`	This region could not be timed due to the granularity of the clock supported on the current architecture.

`m`	Invalid time management was detected for this region due to an unprofilable code construct, an unprofilable command such as an `exec` or `fork`, or incorrect instrumentation in your program or in a library routine. To work around incorrect instrumentation, you can: • Deselect the regions in your routines that are listed with a profiling status of `m`. • Choose not to profile library routines if a library routine is listed with a profiling status of `m`. By default, library routines are not profiled, so this will only occur if you used the `-cxpalib` compiler option to include library routines.
`p`	The program was paused at this point, and the timing information is incomplete.
`t`	The program terminated at this point.
`u`	This region was uninstrumentable because it was too small to gather timing data, was optimized to a single instruction, or contains an unrecognized construct.
`x`	CPU time (excluding called routines and/or inner loops) and ratio(s) cannot be computed accurately.
`y`	Wall clock time (excluding called routines and/or inner loops) and ratio(s) cannot be computed accurately.
`z`	Latency time (excluding called routines and/or inner loops) and ratio(s) cannot be computed accurately.

Context

To view parallel region reports from the Analysis Report window in X window mode, select Parallel Regions as the region level.

To view parallel region reports in line mode, use the `analyze pregion` command.

Related Commands	`analyze`
Related Concepts	Introducing metrics Reports Loop reports Routine reports
Related windows	Analysis Report window Customize Report dialog

Routine reports

Description

Routine performance analysis reports display metrics for profiled routines in your program.

Routine reports are only available if the source files containing regions you wish to profile at the routine level are compiled with the `-cxpa` or `-cxpar` option.

Depending on the architecture on which you created the PDF and the type of metrics collected, the following reports can be displayed for each routine executed in your program that contains profiled routines:

- Computation
- Time to Solution
- Events (SPP and C4 Series only)

For all report types, metrics are displayed

- Excluding values for called routines (less children)
- Including values for called routines (plus children)

Refer to the "Fields" section of this reference page for information about fields (columns) that can appear in routine reports.

Reports

For routines, three types of reports are available: Computation, Time to Solution, and Events. The output of these reports is described in the following sections, along with sample reports.

Computation

The metrics in the Computation report for routines can be used to examine:

- Number of times each routine was executed
- Total CPU time spent and percentage of program's total CPU time for each routine.

Analyzing this information at the routine level can help you identify which routines require the greatest portion of total execution time; you can then profile and analyze these routines in greater detail. A sample computation report for routines is shown below:

```
================================================================================
                        Routine Performance Analysis
================================================================================

                                    Computation
   (less children)       (plus children)       Times
  CPU Time      %       CPU Time      %        Exec      PS    Routine Name
- ------------  -----  ------------  -----  ----------- -- --------------------
       1.382   50.6%         1.382   50.6%           1     munge
       1.336   48.9%         1.336   48.9%           1     init
       0.011    0.4%         0.011    0.4%           3     display
       1.935m   0.1%         2.730  100.0%           1 e   start

(e) incomplete, contains exit.
================================================================================
```

Time to solution

The metrics in the Time to Solution report for routines can be used to examine the following:

- Total wall clock time spent in routines
- The ratio of CPU time to wall clock time

For routines that execute serially, if the CPU/wall clock percentage is high, the CPU is being utilized efficiently in the region. If this percentage is low, it indicates that there is some type of performance bottleneck (assuming the data is significant in light of the amount of CPU work available). Performance bottlenecks can be caused by:

- I/O calls (for example, read() or write() calls)
- System calls (for example, open() or close() calls)

- Memory accesses (for example, cache misses). You can compare event metrics and latency for these loops to find out if the bottleneck is due to memory accesses.

For routines that execute in parallel, the CPU/wall clock metric represents the concurrency factor, which indicates the benefit achieved from parallelism. For this case, values for CPU/wall clock time that are less than 1 indicate that, for the current size of the problem (i.e., the number of processors or the size of the data set), no real benefit was achieved from parallelism for that routine. Values that approach *n*, where *n* is the number of processors, indicate good use of resources within a routine.

For example, as you increase the number of processors or the amount of work (data set size), the concurrency factor should increase proportionally.

Routines that execute in parallel can also exhibit performance bottlenecks due to I/O calls, system calls, and memory accesses, as described above.

A sample time-to-solution report for a routine is shown below:

```
================================================================================
                        Routine Performance Analysis
================================================================================

                                Time to Solution
        (less children)                    (plus children)
 Wall Clock      %     CPU/Wall    Wall Clock      %     CPU/Wall    PS Routine Name
- ------------  -----  --------- ------------  -----  ---------  -- ------------
       1.501   60.4%       0.89         1.501   60.4%       0.89     init
       0.894   36.0%       1.55         0.894   36.0%       1.55     munge
       0.029    1.2%       0.07         2.443   98.3%       1.12  e  start
       0.019    0.8%       0.56         0.019    0.8%       0.56     display

(e) incomplete, contains exit.
================================================================================
```

Events (C4 and SPP Series only)

If you have chosen to collect events, CXpa displays an Events report for routines. The Events report heading indicates the type of event collected for that run of your program (only one type of event can be collected per program run).

The types of events collected vary according to machine architecture. On SPP Series machines, CXpa collects data cache miss counts and latency time for accessing memory hierarchies. On C4 Series machines, CXpa collects instruction, data, and page table cache miss counts and vector load/store operation counts.

Refer to the *Exemplar Architecture* (Order No. DHW-014) and the *C Series Architecture* (Order No. DHW-300) reference manuals for more information about these architectures. For more details on the types of metrics, refer to the "Introducing metrics" online help topic or section of the *CXpa Reference*.

Events metrics for routines enable you to examine:

- Total number of events that occurred in a routine or routines.
- Latency (SPP Series only)—The total wall clock time spent accessing memory to locate data not found in the cache.
- Percentage of CPU time spent accessing memory to locate data not found in the cache. This is expressed as event latency/CPU time, shown in the % CPU column (SPP Series only).

  The % CPU value indicates how much of your computational work is being performed vs. the memory latency you are incurring in a source region for the active thread. If the percentage is low, it means that the CPU found all of the data it needed to do its job in the cache (close at hand). If the percentage is high, it means the CPU had to spend a lot of time asking the memory system to retrieve the data it needed relative to the amount of work it had to do.

  This percentage must be looked at in light of the amount of CPU work available to make a significant observation about program performance in light of cache effects. For example, if the event latency/CPU (% CPU) is 75%, and the overall CPU time is 25 secs for the routine, then it is probably safe to say that cache contention is occurring. You could then try to search out the offending memory access pattern in the region and correct it. If the event latency/CPU (% CPU) is 75%, but the overall CPU time is only 25 usecs, then the data is probably not significant.
- Balance of work distributed among threads—By looking at latency and cache miss counts for events across threads, you can find routines whose work is not shared equally across threads. You can also examine event metrics for those routines to see whether performance bottlenecks are due to cache effects.

A sample Events report (generated on an SPP Series machine) is shown below. For this program run, local cache misses were collected for both read and write operations. Local cache miss counts indicate the number of times the program had to access memory on a processor's node due to a processor cache miss.

```
================================================================================
                        Routine Performance Analysis
================================================================================

                        Local Cache Miss Events
        (less children)                    (plus children)              Routine
  Number of     Latency    % CPU    Number of     Latency    % CPU  PS  Name
- ------------ ----------  -----  ------------ ----------  -----  --  -------
       395101       0.221   16.6%       395101       0.221   16.6%     init
       394777       0.196   14.2%       394777       0.196   14.2%     munge
         1802       0.926m  47.8%       794315       0.420   15.4% e   start
         2635       1.312m  12.5%         2635       1.312m  12.5%     display

(e) incomplete, contains exit.

================================================================================
```

Fields

Routine reports are output on a per routine basis, for all profiled routines in your program that contain routines, with a report header similar to the following:

```
================================================================================
                        Routine Performance Analysis
================================================================================
```

Fields that can appear in Routine reports are listed and described below:

- `Times Exec`—Number of times the routine was executed.

NOTE: All CXpa reports express time in seconds unless the time is annotated with the letter "m" for milliseconds.

- `CPU Time`—This is the total CPU time spent for all threads executing the routine.
- `Wall Clock`—This is the time to solution for all threads executing the routine. Wall clock time includes time when the process is idle.

- `CPU/Wall`—Concurrency factor for the routines. The concurrency factor is the wall clock time divided by the CPU time and indicates the speedup achieved through parallelization.
- `Number of`—Lists the number of times that the selected hardware event occurred.
- `Latency`—Lists the estimated time the process had to stop while waiting for the missing information. This information is an estimate on C4 Series computers and is exact on SPP Series computers.
- `%`—Lists the percent of CPU time for each routine.
- `% CPU`—Lists the percent of CPU time that was spent waiting for missing information.
- `Routine Name`—Lists the name of the routine where the miss occurred.
- `PS`—Displays the profiling status. If this column is blank, then the region executed normally. Other possible profiling statuses are as follows:

Profiling status	Meaning
`e`	The program exited at this point.
`g`	This region could not be timed due to the granularity of the clock supported on the current architecture.
`m`	Invalid time management was detected for this region due to an unprofilable code construct, an unprofilable command such as an `exec` or `fork`, or incorrect instrumentation in your program or in a library routine. To work around incorrect instrumentation, you can: • Deselect the regions in your routines that are listed with a profiling status of `m`. • Choose not to profile library routines if a library routine is listed with a profiling status of `m`. By default, library routines are not profiled, so this will only occur if you used the `-cxpalib` compiler option to include library routines.
`p`	The program was paused at this point, and the timing information is incomplete.
`t`	The program terminated at this point.

`u`	This region was uninstrumentable because it was too small to gather timing data, was optimized to a single instruction, or contains an unrecognized construct.
`x`	CPU time (excluding called routines and/or inner loops) and ratio(s) cannot be computed accurately.
`y`	Wall clock time (excluding called routines and/or inner loops) and ratio(s) cannot be computed accurately.
`z`	Latency time (excluding called routines and/or inner loops) and ratio(s) cannot be computed accurately.

Context

To display Routine reports in X window mode, bring up the Analysis Report window and choose Routines as the region level.

To display Routine reports in line mode, use the `analyze routine` command.

Related Commands

`analyze`

Related Concepts

Introducing metrics
Parallel Region reports
Loop reports
Reports

Related windows

Analysis Report window
Customize Report dialog

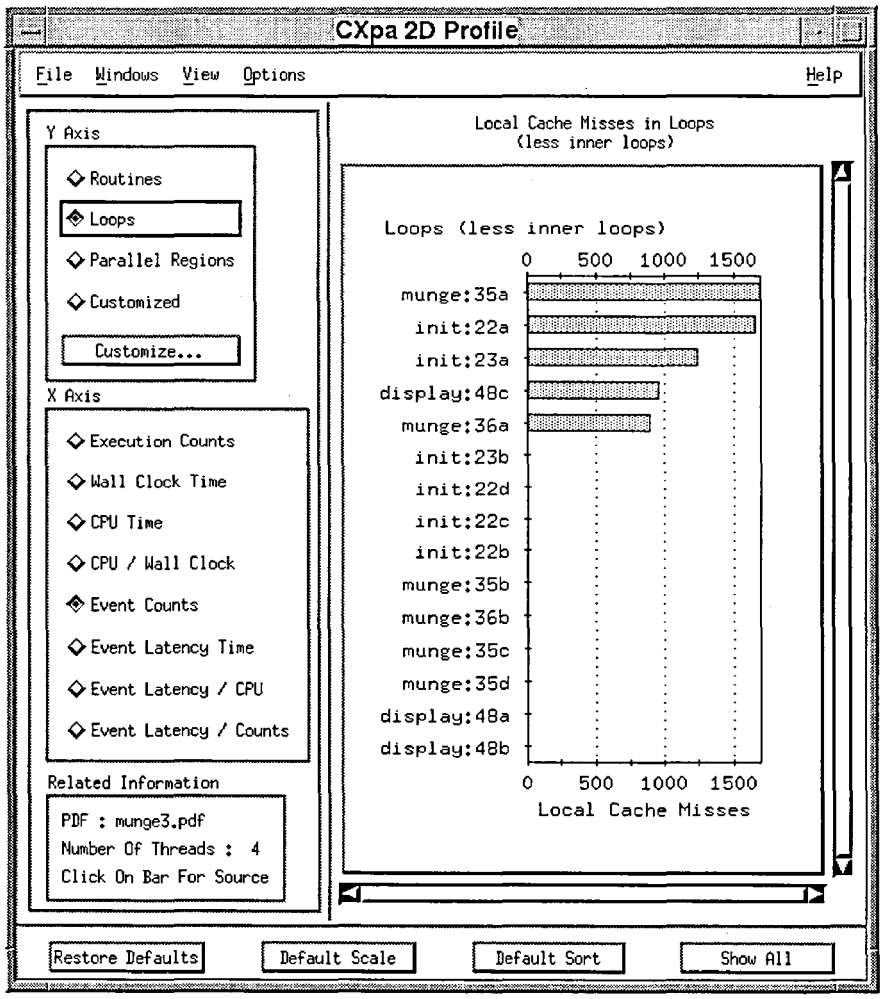
Windows

5

This chapter contains help pages for all CXpa windows and dialogs. These pages are in alphabetical order by title. Each help page is divided into the following sections:

- **Description**—Explains the purpose and functionality of the window or dialog.
- **Fields**—Describes the purpose of each field that appears in the window or dialog.
- **Buttons**—Describes the function of each button that appears in the window or dialog.
- **Context**—Describes the action that makes this window or dialog appear.
- **Related Windows, Topics, or Commands**—Lists sections of this document online help topics that contain related information.

2D Profile window



Description

The 2D Profile window displays a two-dimensional graph of various kinds of performance data for the selected regions of your program. This graph displays data for the entire process. To display performance data for individual threads, use the 3D Profile window.

2D Profile window

The data for 2D profile graphs can be sorted alphabetically, by load order, from lowest to highest, or from highest to lowest.

Scaling options for the 2D profile graph allow you to view or specify the range (X axis) or number/percentage (Y axis) of data values displayed at one time in the 2D Profile window. This can be useful when there is a large number of data items to graph and you want to focus on a subset of the data. You can save 2D profile graphs in PostScript, xwd, or ASCII formats.

Source code correlation

To display the source code associated with a bar in the 2D profile graph, click on that bar with the left mouse button.

Menus

Name	Meaning
File	Contains options for saving the graph in a PostScript, ASCII, or Xwd file and closing the window.
Windows	Contains options for creating additional CXpa windows for viewing 2D or 3D profile graphs, performance reports, or source files.
View	Contains options for sorting and/or scaling the data in the graph.
Options	Contains an option (Filter Profile) that enables you to include or exclude values for called routines and/or inner loops in the graph.
Help	Contains various help options for invoking the CXpa help system.

Y Axis

Contains buttons that change the type of program region to graph on the Y (vertical) axis.

Name	Meaning
Routines	Displays a graph of routine performance.
Loops	Displays a graph of loop performance.
Parallel Regions	Displays a graph of parallel region (parallel loop) performance.
Customized	Indicates that the window is displaying a customized graph created using the Profile Customization dialog.

Customize	Displays the Profile Customization dialog that enables you to choose the data that appears in the graph.

X Axis

Contains options to change the type of metrics this window graphs on the X (horizontal) axis. The available X-axis items represent metrics that can be collected. The metrics options displayed differ according to machine architecture, and are listed and described below.

Metric	Definition
All architectures	
Execution Counts	The number of times a loop, routine, or parallel region is executed.
Wall Clock Time	Time to solution and includes time when the process is idle.
CPU Time	Time when the CPU works on the process.
CPU/Wall Clock	Concurrency factor for parallel regions, which is an indicator of the speed-up achieved through parallelization. For serial regions, this represents the percentage of wall clock time spent in computation (vs. time spent waiting for I/O, system calls, or accessing memory).
C Series only	
Vector Mflops	Millions of floating point operations per second performed by vector functional units.
C4 Series and SPP Series only	
Event Counts	The number of times a selected event occurred (for example, the number of data cache misses that occurred in selected regions of your program). The types of events graphed vary between C4 Series and SPP Series machines.
SPP Series only	
Event Latency Time	The time the process spent in wait latency for the type of memory event you are collecting.
Event Latency/CPU	The percentage of CPU time the process spent in wait latency for the type of memory event you are collecting.

Event Latency/Counts — The average time the process spent in wait latency per event for the type of memory event you are collecting.

The following table shows the availability of metrics by architecture and the relationship between available metrics, compiler options, and source code regions that can be selected for profiling.

X-axis option (metrics)	Collectable on these architectures	Collectable for these regions	Required compiler options
Execution Counts	All	All	`-cxpa` or `-cxpar`
Wall Clock Time	All	All	`-cxpa` or `-cxpar`
CPU Time	All	All	`-cxpa` or `-cxpar`
CPU/Wall Clock	C2, C3, C4, SPP Series	All	`-cxpa or -cxpar`
Vector Mflops	C2, C3, C4 Series	Loop only	`-O2` or `-O3`
Event Counts	C4, SPP Series	Routines	`-cxpa` or `-cxpar; -O1` or greater
		Loops	`-cxpa; -O2 or -O3`
		Parallel regions	`-cxpa -O3`
Event Latency Time	SPP Series only	Routines	`-cxpa` or `-cxpar; -O1` or greater
		Loops	`-cxpa; -O2` or `-O3`
		Parallel regions	`-cxpa -O3`
Event Latency/ CPU	SPP Series only	Routines	`-cxpa` or `-cxpar; -O1` or greater
		Loops	`-cxpa; -O2` or `-O3`
		Parallel regions	`-cxpa -O3`
Event Latency/ Counts	SPP Series only	Routines	`-cxpa` or `-cxpar; -O1` or greater
		Loops	`-cxpa; -O2` or `-O3`
		Parallel regions	`-cxpa -O3`

Related Information

Displays the name of the PDF, the number of threads, and directions for displaying source code associated with any bar in the 2D profile graph.

Buttons

Name	Meaning
Restore Defaults	Sets the sort and scale values to their default.
Default Scale	Sets the scale value to the default.
Default Sort	Sets the sort value to the default.
Show All	Shows the entire graph. The entire graph may not appear in the window if the graph is very large.

Context

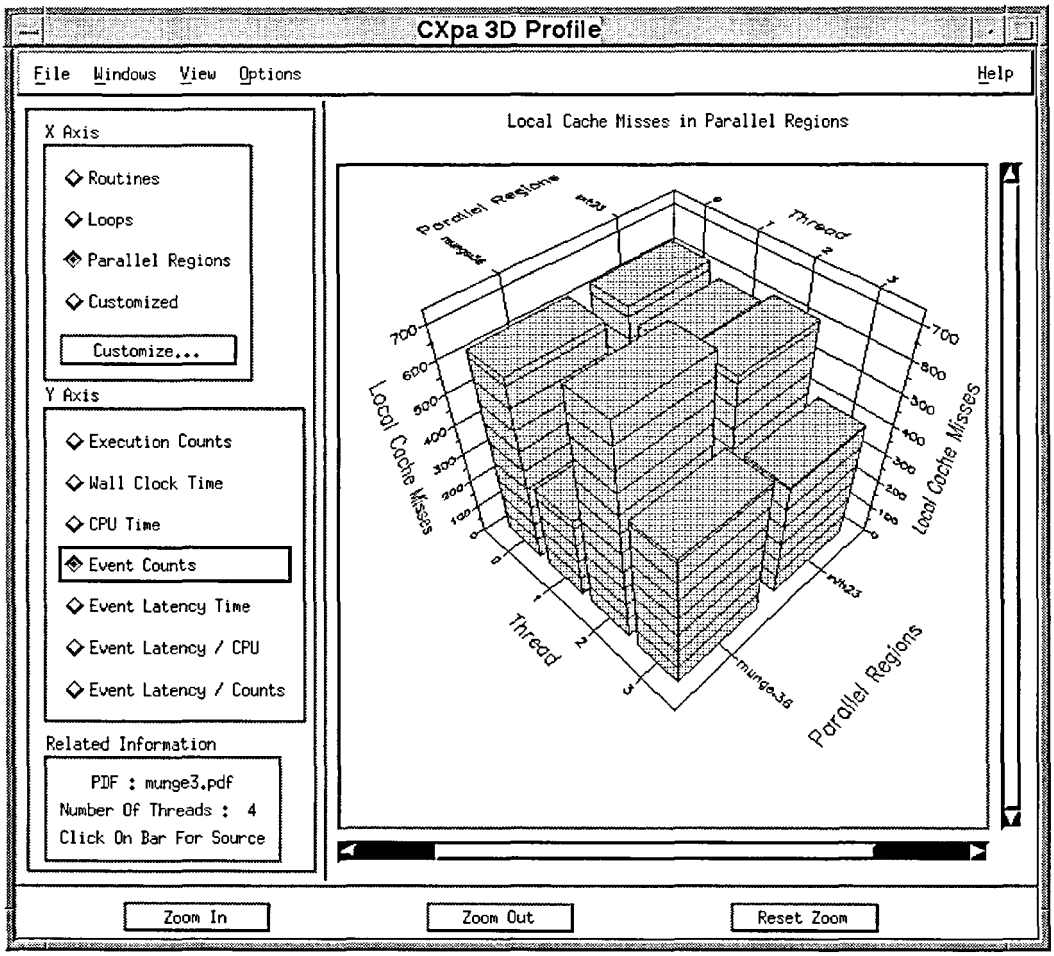
The 2D Profile window appears when you:

- Press the Create 2D Profile button in the Analysis Manager window or the 2D Profile button in the Executable Manager window.
- Select the 2D Profile option from the Windows menu in any CXpa window.
- Invoke CXpa with the name of a PDF file only (without specifying an executable), if you have set the X application resource `Cxpa*defaultWindow` to `2DProfile`.

Related Windows

3D Profile window
Analysis Control window
Analysis Report window
Executable Manager window
Filter Profile dialog
Profile Customization dialog
Profile Selection dialog
Save Profile dialog
Scale dialog
Sort dialog
Visibility Selection dialog

3D Profile window



Description

The 3D Profile window displays a three-dimensional graph of various kinds of parallel performance data (which vary according to architecture) for the profiled regions of your program. Performance data is graphed on a per-thread basis. The data for 3D profile graphs can be sorted alphabetically, by load order, from lowest to highest, or from highest to lowest. The default is from highest to lowest. You can save 3D profile graphs in PostScript, xwd, or ASCII formats.

Source code correlation

You can display the source code for a routine, loop or parallel region by clicking with the left mouse button on the corresponding bar in the graph.

Graph rotation

You can rotate the graph by placing the mouse pointer over the graph and holding down the middle mouse button. To return the graph to its original position, press the Reset Zoom button.

Menus

Name	Meaning
File	Contains options for saving the graph in a PostScript, ASCII, or Xwd file and closing the window.
Windows	Contains options for creating additional CXpa windows for viewing 2D and 3D profile graphs, performance reports, and source files.
View	Contains options for sorting the data in the 3D graph.
Options	Contains an option for including or excluding values for called routines and/or inner loops in the graph.
Help	Contains various help options that invoke the CXpa help system.

X Axis

Contains buttons that change the type of program region to graph on the X (horizontal) axis.

Region level option	Action
Routines	Displays a graph of routine performance.
Loops	Displays a graph of loop performance.
Parallel Regions	Displays a graph of parallel region performance.
Customized	Indicates that the window is displaying a customized graph created using the Profile Customize dialog.
Customize	Displays the Profile Customization dialog that enables you to choose the data that appears in the graph.

Y Axis

Contains options for changing the type of metrics this window graphs on the Y (vertical) axis. The available Y-axis items represent metrics that can be collected. The metrics options displayed differ according to machine architecture, and are listed and described below.

Metric option	Description
All architectures	
Execution Counts	Number of times a loop, routine, or parallel region is executed.
Wall Clock Time	Time to solution, including time when the process is idle.
CPU Time	Time that the CPU works on the process.
C Series only	
Vector Mflops	Millions of floating point operations per second performed by vector functional units.
C4 Series and SPP Series only	
Event Counts	Number of times a selected event occurred (e.g., the number of data cache misses that occurred in profiled regions of your program). The types of events graphed vary between C4 Series and SPP Series machines.
SPP Series only	
Event Latency Time	Time the process spent in wait latency for the type of memory event you are collecting.
Event Latency/ CPU	Percentage of CPU time the process spent in wait latency for the type of memory event you are collecting.
Event Latency/ Counts	Average time the process spent in wait latency per event for the type of memory event you are

The following table shows the availability of metrics by architecture and the relationship between available metrics, compiler options, and source code regions that can be selected for profiling.

X-axis option (metrics)	Collectable on these architectures	Collectable for these regions	Required compiler options
Execution Counts	All	All	`-cxpa, -cxpar, or -cxpab`
Wall Clock Time	All	All	`-cxpa, -cxpar, or -cxpab`
CPU Time	All	All	`-cxpa, -cxpar, or -cxpab`
Vector Mflops	C2, C3, C4 Series	Loop only	`-02 or -03`
Event Counts	C4, SPP Series	Routines	`-cxpa or -cxpar; -01, -02 or -03`
		Loops	`-cxpa; -02 or -03`
		Basic blocks	`-cxpab`
		Parallel regions	`-cxpa -03`
Event Latency Time	SPP Series only	Routines	`-cxpa or -cxpar; -01, -02 or -03`
		Loops	`-cxpa; -02 or -03`
		Basic blocks	`-cxpab`
		Parallel regions	`-cxpa -03`
Event Latency/ CPU	SPP Series only	Routines	`-cxpa or -cxpar; -01, -02 or -03`
		Loops	`-cxpa; -02 or -03`
		Basic blocks	`-cxpab`
		Parallel regions	`-cxpa -03`
Event Latency/ Counts	SPP Series only	Routines	`-cxpa or -cxpar; -01, -02 or -03`
		Loops	`-cxpa; -02 or -03`
		Basic blocks	`-cxpab`
		Parallel regions	`-cxpa -03`

Related Information

Displays the name of the PDF, the number of threads, and directions for displaying source code associated with any bar in the 3D profile graph.

Buttons

Name	Meaning
Zoom In	Zooms in (enlarges) the upper-left corner of the graph.
Zoom Out	Makes the graph smaller.
Reset Zoom	Returns the graph to its original zoom and rotation settings.

Context

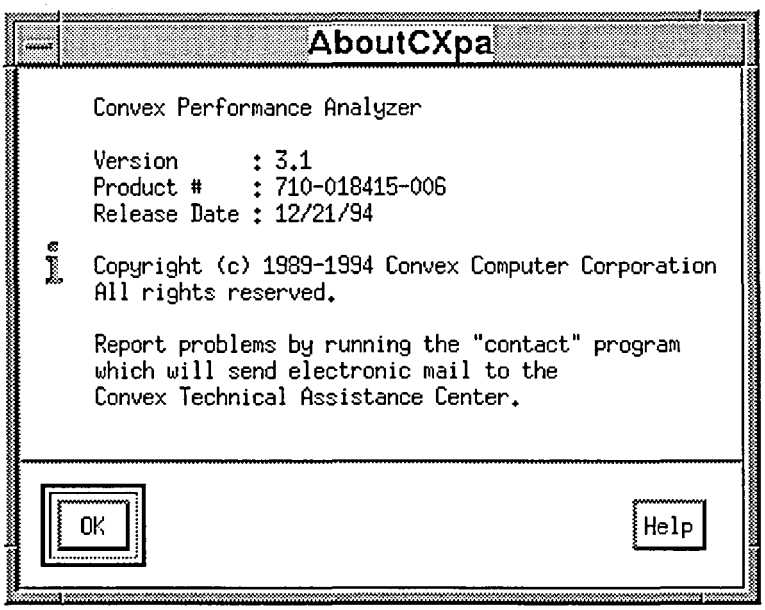
The 3D Profile window appears when you:

- Press the Create 3D Profile button in the Analysis Manager window or the 3D Profile button in the Executable Manager window.
- Select the 3D Profile option in the Windows menu of any CXpa window.
- Invoke CXpa with the name of a PDF file only (without specifying an executable), if you have set the X application resource `Cxpa*defaultWindow` to `3DProfile`.

Related Windows

2D Profile window
Analysis Report window
Filter Profile dialog
Profile Selection dialog
Sort dialog
Visibility Selection dialog
Analysis Control window
Executable Manager window
Profile Customization dialog
Save Profile dialog
3D Profile window

About CXpa



Description

The About CXpa dialog displays release information:

- CXpa's version number
- The release date of this version of CXpa
- The product number
- Copyright information
- Information about using the `contact` utility to report problems

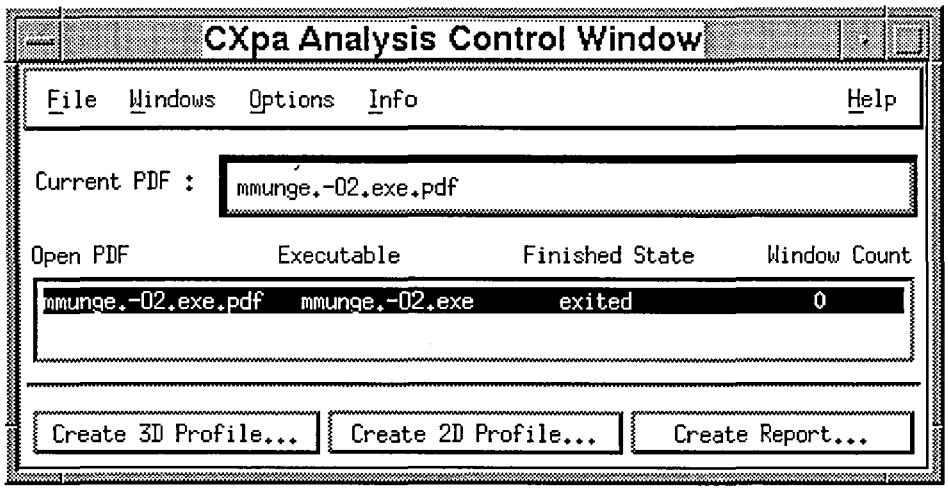
Buttons

Name	Action
OK	Closes this dialog.
Help	Displays a help page for this dialog.

Context

The About CXpa dialog appears when you select the Product Information option from the Help menu on any CXpa window.

Analysis Control window



Description

The Analysis Control window appears when you invoke CXpa with the name of a PDF file only. From the Analysis Control window you can:

- Create and view performance reports or 2D and 3D profiles from the information in any number of PDFs, including PDFs created on different architectures, from different executables, or during previous CXpa sessions
- Open multiple profile graphs and reports per PDF to compare performance analysis information
- View source files for your application

Invoking CXpa with a PDF only

If you just want to view previously collected performance information, you can invoke CXpa with the PDF file name (without specifying an executable):

`cxpa` *<pdf-name>*

The Analysis Control window appears.

NOTE: You can use a CXpa X application resource to specify automatic creation of a profile or report window when you invoke CXpa with a PDF file only (in "analysis mode").

The resource is `Cxpa*defaultWindow`, and it can accept one of four values: `2DProfile`, `3DProfile`, `Report`, or `NoWindow`. The default is `NoWindow`.

Choosing a current PDF

You can choose a current PDF by clicking on a PDF in the Open PDF column. Press the buttons at the bottom of this window to create a 2D or 3D profile graph or a text report of the data in the current PDF.

Opening PDFs

To open a new PDF, select the Open PDF option from the File menu. A new entry appears in the PDF list. You can also open a PDF and make it current by typing its name directly in the Current PDF field. If you do not specify a relative or full path, CXpa looks for the PDF file in the current directory.

Menus

Name	Meaning
File	Contains options to open an existing PDF or to exit CXpa.
Windows	Contains options for creating additional windows for viewing 2D and 3D profile graphs, performance reports, or source files.
Options	Contains options for changing CXpa's source code search path or setting visibility for library and application routines.
Info	Contains an option to display information on the current PDF or your current session.
Help	Contains options for invoking the CXpa help system.

Fields

Name	Meaning
Current PDF	Displays the name of the PDF used when you press one of the Create buttons at the bottom of the window or from a menu. You can also enter a PDF name in this field to open a new PDF.
Open PDF	Lists the names of available PDFs. The current PDF is always highlighted.

Executable	Lists the name of the executable that created the PDF.
Finished State	Lists the state of the executable when it finished executing.
Window Count	Lists the number of windows on your display that are associated with the listed PDF.

Buttons

Name	Action
Create 3D Profile	Displays a window containing a 3D profile graph.
Create 2D Profile	Displays a window containing a 2D profile graph.
Create Report	Displays a window containing textual performance reports.

Context

The Analysis Control window appears when you invoke CXpa with the name of a PDF file only.

CXpa looks for a.out.pdf in the current directory and displays the Analysis Control window.

Related Windows

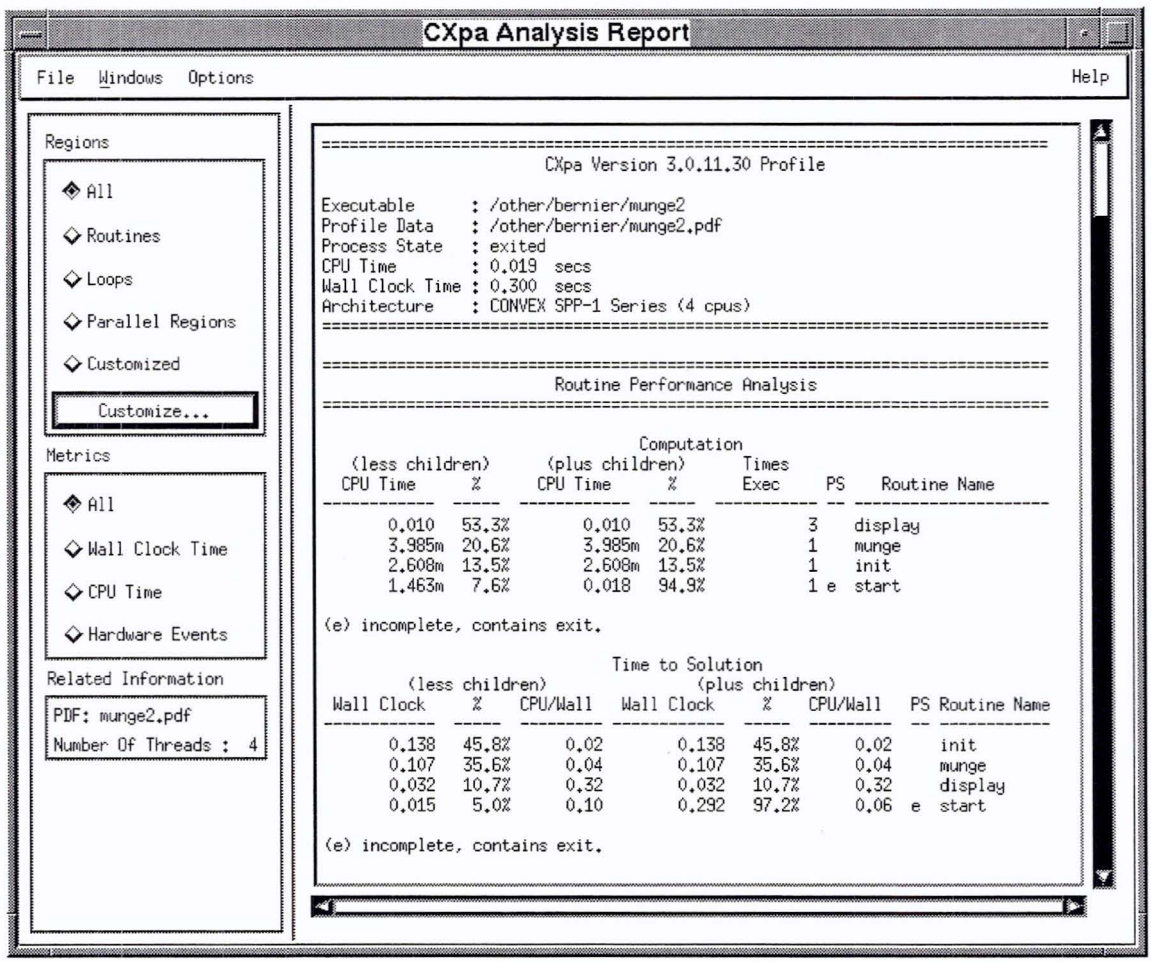
2D Profile window
Analysis Control window
Open PDF dialog
Info Session dialog
Source Code window
Visibility Selection dialog
3D Profile window
Analysis Report window
Info PDF dialog
Source Code Selection dialog
Source Search Path dialog

Related Topics

Analyzing PDFs only
Setting the PDF

Analysis Control window

Analysis Report window



Description

The Analysis Report window allows you to display textual performance reports generated from the data in the active PDF. You can create multiple Analysis Report windows, allowing you to compare information among several PDFs or to examine different types of analysis reports for a single PDF. By default, when you first bring up the Analysis Report window, all available metrics are displayed for all profiled regions of your program.

Analysis Report window

You can display specific metrics for various types of source code regions by clicking one of the buttons in the Regions section and a button from the Metrics section. You can also create customized reports (refer to the reference topic for the "Customize Report dialog" for more information).

You can select whether you want to display performance data in reports per process (the default) or per thread, by selecting the Filter Report option from the Options menu. Threaded data is displayed on a per-routine basis for each thread, with the number of the thread displayed in the report heading for each routine.

For more information on the types of reports available at each region level and the metrics displayed in each type of report, refer to the following online help topics or sections in this book: "Reports", "Loop reports", Parallel Region reports, "Routine reports," "Basic Block report," and "Interpreting Mflops data."

Menus

Name	Options
File	Contains options to save the report in the window to a file or close the window.
Windows	Contains options for creating additional windows for viewing 2D and 3D profile graphs, performance reports, or source files.
Options	Contains an option for filtering performance report data per process (the default) or per thread.
Help	Contains various help options that invoke the CXpa help system.

Regions

Contains options to change the type of source code region for which reports are displayed.

Region level option	Action
All	Displays reports for all profiled regions.
Routines	Displays Routine reports.
Loops	Displays Loop reports.
Parallel Regions	Displays Parallel Region reports.
Customized	Indicates that the report in the window is a customized report created by clicking the Customize button.

Buttons

Name	Action
Customize	Displays the Customize Report dialog that allows you to create a customized report on specific routines at selected region levels (that is, for routines, loops, parallel regions, or basic blocks).

Metrics

Contains options to change the metrics displayed in CXpa reports for the selected source code regions.

Metric option	Action
All	Displays all available reports and metrics. This is the default.
CPU Time	Displays Computation reports containing CPU time and iteration/execution counts.
Wall Clock Time	Displays Time to Solution reports containing wall clock time.
Hardware Events	(C4 Series and SPP Series only) Displays hardware event reports which includes event counts and, for SPP Series only, latency times for events.
Vector Mflops	(C Series only) Displays a Vector Register Utilization report, which includes vector spills, vector flops, chime counts, and estimated Mflops.

Related Information

Label	Meaning
PDF	Displays the name of the file in which the current performance data is stored.
Number of Threads	Displays the number of threads in your program.

Analysis Report window

Context

The Analysis Report window appears when you:

- Press the Report button on the Executable Manager window
- Press the Create Report button on the Analysis Control window
- Select the Report option from the Windows menu in any CXpa window

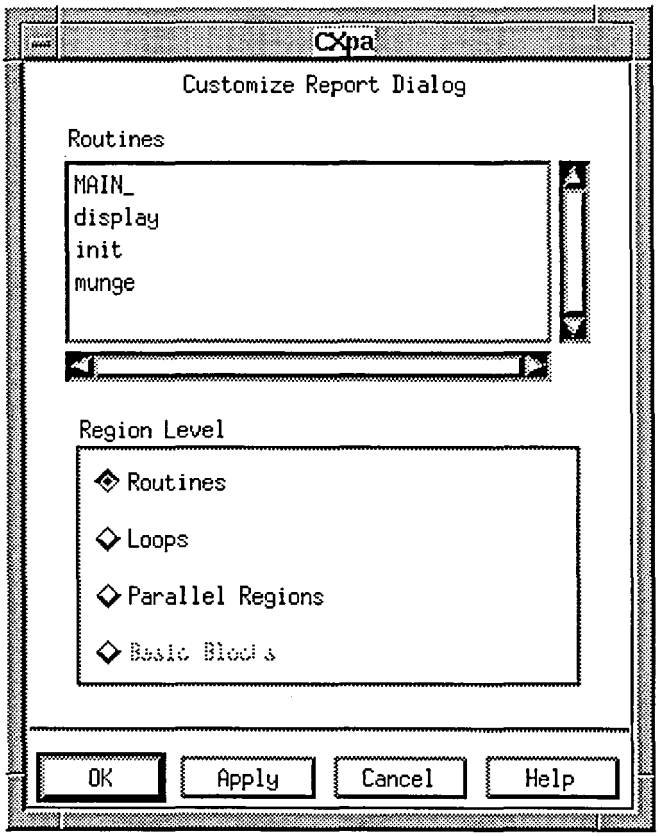
Related Windows

2D Profile window
3D Profile window
Analysis Control window
Customize Report dialog
Executable Manager window
Filter Report dialog
Profile Selection dialog
Save Report dialog

Related Topics

Basic Block report
Interpreting Mflops data
Introducing metrics
Loop reports
Parallel Region reports
Reports
Routine reports
Selecting metrics in X window mode

Customize Report



Description

The Customize Report dialog allows you to create a customized report for specific routines at a selected region level. Use this dialog when you want to display reports for a subset of routines.

To display a customized report:

1. Choose the region type for which you want to customize a report. (You can select only one.)
2. Select the routines you want to include in the report by highlighting routine names in the Routines column with the mouse. (You can select multiple routines.)

Customize Report

3. Press the OK button to display the new report in the Analysis Report window.

Fields

Heading	Meaning
Routines	Allows you to select the routines whose performance information you want to include in the report. Multiple routines can be selected.
Region Level	Allows you to select the type of source code region for which you want to display performance reports. You may select only one of these at a time.

Buttons

Name	Action
OK	Displays a customized report in the Analysis Report window and closes this dialog.
Apply	Displays a customized report in the Analysis Report window without closing this dialog.
Cancel	Closes this dialog without updating the report in the Analysis Report window.
Help	Displays a help page for this dialog.

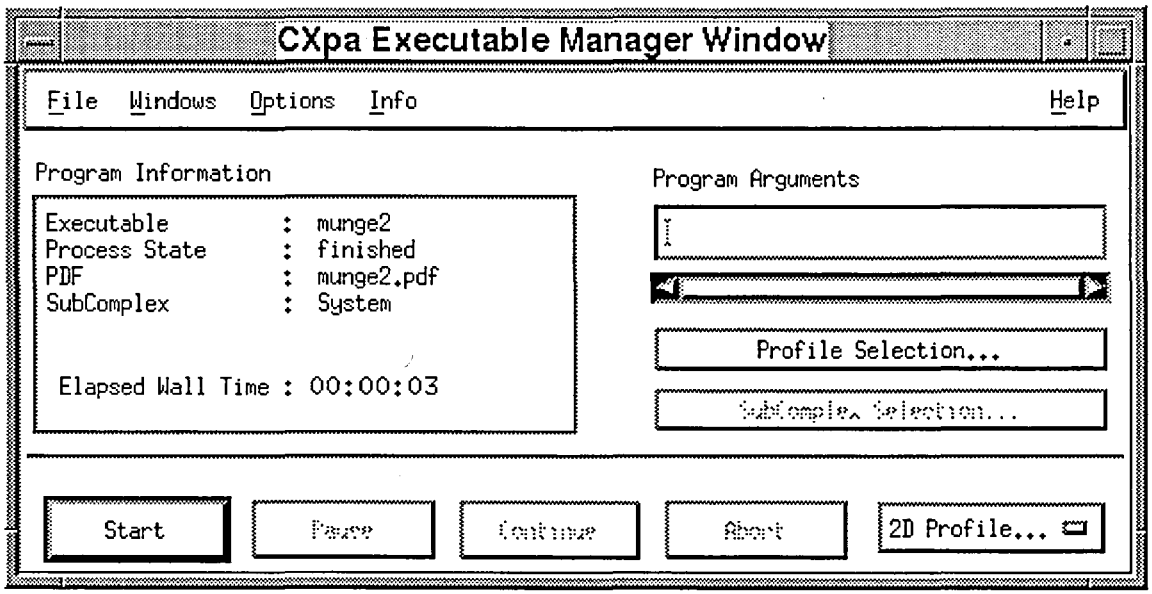
Context

The Customize Report dialog appears when you press the Customize button on the Analysis Report window.

Related Windows

Analysis Report window
Filter Report dialog
Save Report dialog

Executable Manager window



Description

The Executable Manager window appears when you invoke CXpa with an executable. From this window, you can profile a program that has been compiled with one of the profiling compiler options (`-cxpa`, `-cxpar`, `-cxpab`).

From the Executable Manager window you can:

- Select source code regions for profiling and metrics to collect
- Specify program arguments
- Execute and profile your program
- Display textual performance reports and 2D or 3D profile graphs
- View source files for your application
- Specify a new PDF name to prevent overwriting an existing PDF
- Overwrite any existing PDF by specifying the name of that PDF

From the Executable Manager window, you cannot display performance reports and profiles from PDFs created in a previous session, generated from a different executable, or generated on a different architecture.

To display performance reports and profiles from previously created PDFs, including PDFs created on other architectures or with different executables, invoke CXpa without specifying an executable (or with the name of a PDF file only) to display the Analysis Control window.

Profiling a program

After you have compiled your program with one of the profiling options, perform the following steps to profile your program:

1. Invoke CXpa with the name of your executable:

   % `cxpa a.out`

   The Executable Manager window appears.

2. If you are running CXpa on an SPP Series system and want to run your process on a different subcomplex, press the Subcomplex Selection button to bring up the Subcomplex Selection dialog. Select the name of the subcomplex on which you want your process to run.

3. Enter any arguments to your program in the Program Arguments field.

4. By default, CXpa collects CPU time and execution counts for all routines in your program. To use these defaults, skip to step 6. To select different source code regions (such as loops and parallel regions only) for profiling and/or collect different metrics, continue with step 5.

5. Press the Profile Selection button. The Profile Selection Dialog appears. Perform the following actions to select different metrics and regions for profiling:

   - Select the source code regions and routines you want to profile by clicking the toggle buttons in the Select Regions to Profile section of the dialog.
   - To change the default metrics (CPU time and execution counts), click the toggle buttons in the Select Metrics to Collect section of the dialog to select/deselect metrics.
   - On SPP Series and C4 Series machines, you can collect hardware event metrics. If you wish to collect event metrics, select Events as one of your metrics choices and use the Event Counter options menu to select the type of event that you want to collect.
   - Press the OK button in the Profile Selection dialog to apply the region and metrics settings and close the dialog. You are now ready to profile your program.

6. Press the Start button to run the program you are profiling. A window appears that displays your program's input and output.

   While your program runs, CXpa collects performance data. When your program completes execution, the Process State field shows a state of `finished`.

   You can use the Pause button at the bottom of the Executable Manager window to suspend your program's execution during profiling.

NOTE: To obtain the most accurate profiling data, you should not pause your program during profiling. For best results, run the program to completion and then analyze the results. You will also see more accurate results if you run the program in a standalone environment.

   Once your program is paused, you can:

   - View an intermediate 2D or 3D profile graph or text report, as shown in step 7.
   - Abort your program
   - Continue your program's execution

7. To display performance information, select one of the options (2D Profile, 3D Profile, or Report) from the options menu button at the bottom-right corner of the window.

   Choose an option by releasing the mouse button while the pointer is over the option you want.

   You can also display performance information by choosing an option from the Windows menu.

8. Exit CXpa by choosing the Exit option from the File menu.

Menus

Name	Meaning
File	Contains options to choose a new PDF or exit CXpa.
Windows	Contains options for creating additional windows for viewing 2D and 3D profile graphs, performance reports, or source files.
Options	Contains options that allow you to change the source code search path or choose whether CXpa-instrumented library and/or application routines are visible to CXpa during profiling or analysis (application/library visibility settings).

Info	Lists options that display information about the PDF, the executable being profiled, and your current CXpa session.
Help	Contains various options that invoke the CXpa help system.

Fields

Name	Meaning
Executable	Lists the executable you are profiling.
Process State	List the process state for the program you are profiling. These states include: Running, Paused, Finished, and Terminated.
PDF	Lists the name of the performance data file (PDF) where performance data will be stored.
Scheduling	(C Series only) Lists whether scheduling is set to Fixed or Time Share. Fixed Scheduling reserves all processors for use by the program you are profiling.
SubComplex	(SPP Series only) Lists the name of the subcomplex on which your executable will run.
Elapsed Wall Time	Presents the actual time that has passed since you pressed the Start button.
Program Arguments	Allows you to specify command line arguments to your program and file redirection.

Buttons

Name	Action
Profile Selection	Displays the Profile Selection dialog that allows you to select regions to profile and metrics to collect.
SubComplex Selection	(SPP Series only) Displays the Subcomplex Selection dialog that allows you to choose the subcomplex that your executable will run on.
Start	Runs the executable and initiates profile data collection.
Pause	Pauses the executable and profile data collection.

Continue	Continues a paused program and resumes profile data collection.
Abort	Terminates the program and stops profile data collection.
2D Profile	Click this button to display an options menu for displaying additional windows. The choices are 2D Profile, 3D Profile, and Report. The default is 2D Profile.

Context

The Executable Manager window appears when you start CXpa with an executable file. If you do not specify any options, CXpa looks for a.out in the current directory.

Related Windows

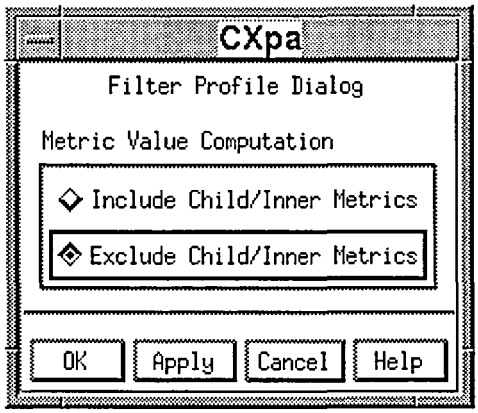
2D Profile window
Analysis Control window
Info Executable dialog
New PDF dialog
Profile Selection dialog
Source Code window
Subcomplex Selection dialog
3D Profile window
Analysis Report window
Info PDF dialog
Info Session dialog
Scheduling Selection dialog
Source Search Path dialog
Visibility Selection dialog

Related Topics

Analyzing PDFs only
Learning CXpa quickly
Introducing metrics
Introducing source code regions
Reports
Selecting metrics in X window mode
Selecting regions in X window mode
Setting the PDF

Executable Manager window

Filter Profile



Description

The Filter Profile dialog contains toggle buttons for specifying whether metrics graphed in the 2D Profile and 3D Profile windows include values for called routines (child routines) and/or inner loops. By default, they are excluded (Exclude Child/Inner Metrics).

Buttons

Heading	Meaning
Metric Value Computation	
Include Child/Inner	For routines, include metrics for called routines (child routines) when computing values for the 2D and 3D Profile graphs. For loops, include metrics for inner loops when computing values.
Exclude Child/Inner	For routines, exclude metrics for called routines (child routines) when computing values for the 2D and 3D Profile graphs. For loops, exclude metrics for inner loops when computing values.

Buttons

Name	Action
OK	Accepts the new settings and closes the dialog. Updates the graphs displayed in the 2D and 3D Profile windows, if they are currently open.

Apply	Apply the new settings without closing the dialog. Updates the graphs displayed in the 2D and 3D Profile windows, if they are currently open.
Cancel	Closes this dialog without making any changes.
Help	Displays a help page for this dialog.

Context

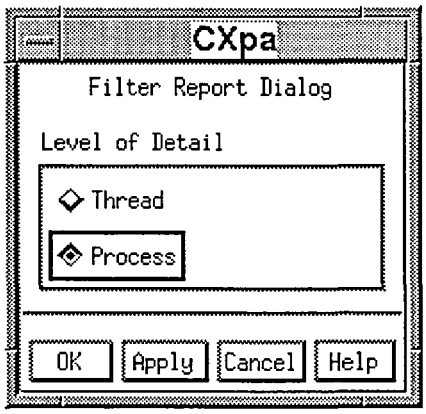
The Filter Profile dialog appears when you choose the Filter Profile option from the Options menu on the 2D Profile window or the 3D Profile window.

Related Windows

3D Profile window

2D Profile window

Filter Report



Description

The Filter Report dialog contains toggle buttons for specifying whether information presented in CXpa reports is presented at the process level or the thread level. The default setting is Process.

Buttons

Name	Action
Level of Detail	
Thread	Sets the level of detail for information presented in performance analysis reports to include data for individual threads in all reports.
Process	Sets the level of detail for information presented in performance analysis reports to the process level. Performance data is summed across all threads of the process (except in parallel region reports).
OK	Accepts the new settings and closes the dialog. Updates the reports displayed in existing Report windows.
Apply	Applies the new settings without closing the dialog. Updates the reports displayed in the Analysis Report window.

Cancel	Closes this dialog without making any changes.
Help	Displays a help page for this dialog.

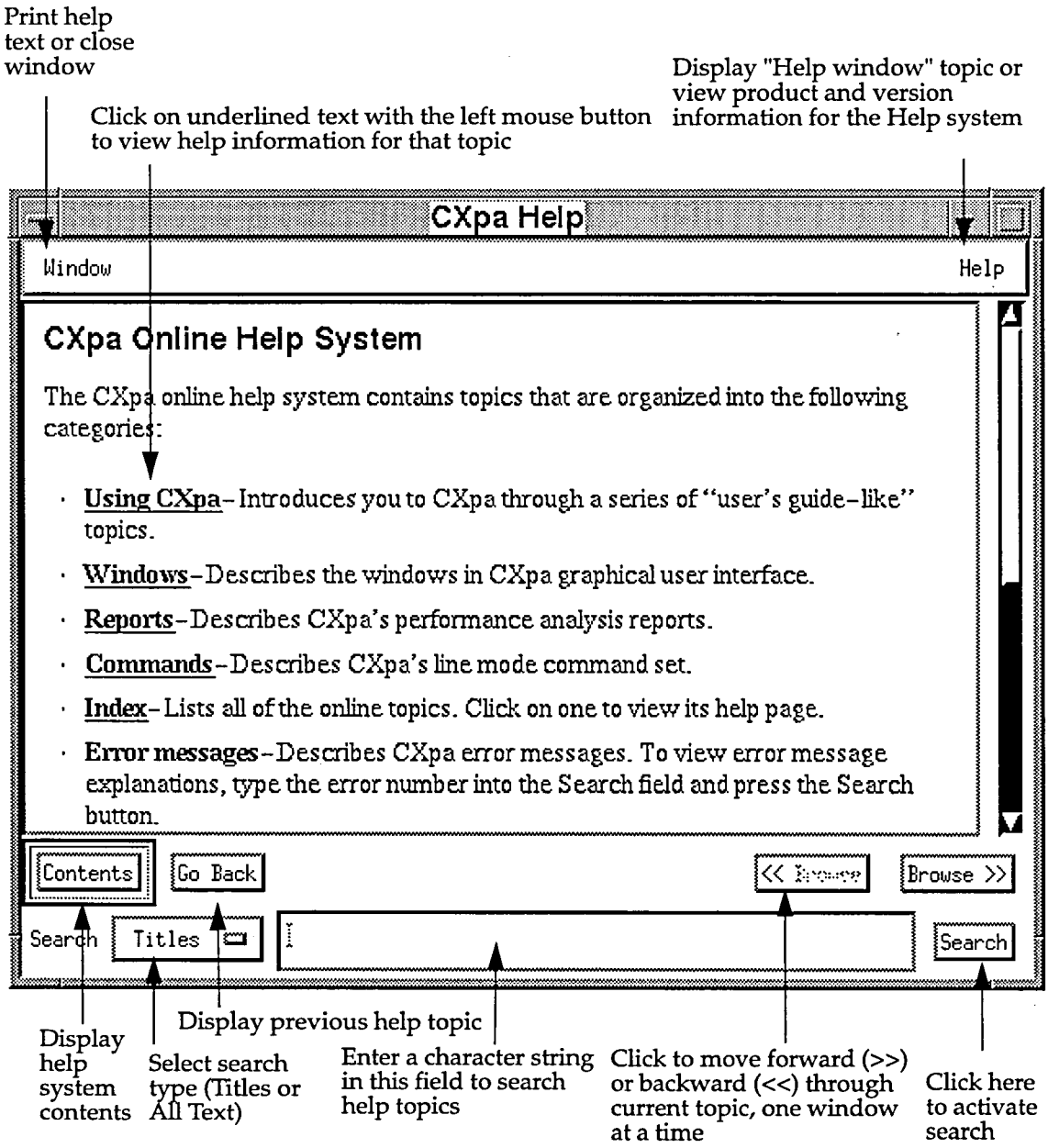
Context

The Filter Report dialog appears when you choose the Filter Report option from the Options menu on the Analysis Report window.

Related Windows

Analysis Report window
Customize Report dialog
Save Report dialog

Help window


Print help text or close window
Click on underlined text with the left mouse button to view help information for that topic
Display "Help window" topic or view product and version information for the Help system
CXpa Help
Window
Help
CXpa Online Help System
The CXpa online help system contains topics that are organized into the following categories:
· Using CXpa – Introduces you to CXpa through a series of "user's guide-like" topics.
· Windows – Describes the windows in CXpa graphical user interface.
· Reports – Describes CXpa's performance analysis reports.
· Commands – Describes CXpa's line mode command set.
· Index – Lists all of the online topics. Click on one to view its help page.
· Error messages – Describes CXpa error messages. To view error message explanations, type the error number into the Search field and press the Search button.
Contents
Go Back
<< Browse
Browse >>
Search
Titles
Search
Display help system contents
Select search type (Titles or All Text)
Display previous help topic
Enter a character string in this field to search help topics
Click to move forward (>>) or backward (<<) through current topic, one window at a time
Click here to activate search

Help window

Description

Using the CXpa Help window, you can navigate the CXpa online help system, print help text, search help topics (by title or full text), display instructions for using the Help system, and display version information for the help system.

To display online help for CXpa messages, enter the message number (for example, A35) in the Search string field.

Menus

Name	Items
Window	Contains the following items: **Print Text**—Pipes a formatted ASCII text version (without graphics) of the current help topic to the `lpr` command. The `lpr` command looks for the printer specified in your PRINTER environment variable. If this variable is not set, nothing is printed. **Close**—Closes the Help window.
Help	Contains the following items: **On Help**—Displays instructions for using the CXpa Help system. **On Version**—Displays a Product Information dialog that identifies the version of the CXpa Help system (Convex Interactive Documentation Toolkit) you are using.

Buttons

Name	Action
Contents	Displays a hyperlinked table of contents for the CXpa online help system.
Go Back	Displays the previous topic stored in the help history. The help history contains all the topics you viewed during the current session.
Browse >>	Lets you move forward through the current help topic, one window length at a time. This button is deactivated when you reach the end of a topic.
<< Browse	Lets you move backward through the current help topic, one window length at a time. This button is deactivated when you reach the beginning of a topic.

Search	Searches for the character string you entered in the Search field.
Titles/All Text	Lets you select whether to search only the topic titles or the full text of all topics.

Context

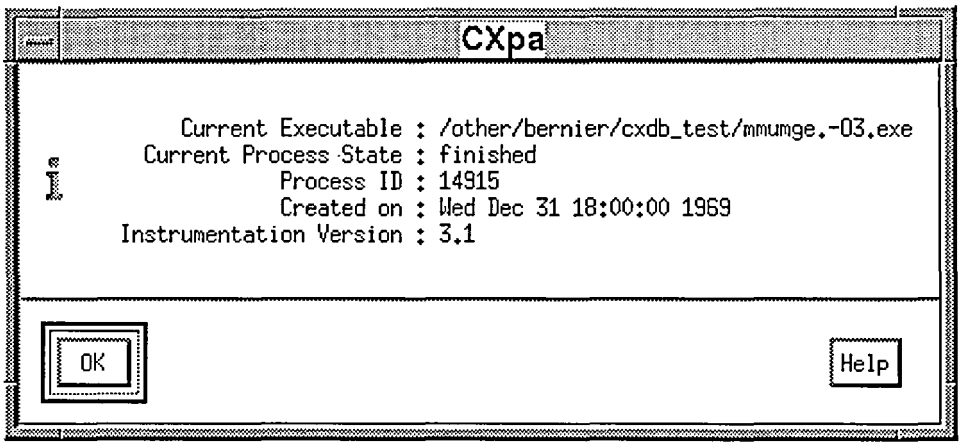
The Help window appears when you:

- Select the Window Overview, CXpa Overview, Using Help, or Tutorial item from the Help menu in the upper right corner of any CXpa window
- Click the Help button on any CXpa dialog

Related Topics

Using online help

Info Executable



Description

The Info Executable dialog displays information about the executable you are profiling.

Fields

Name	Meaning
Current Executable	Lists the executable that you are profiling.
Current Process State	Lists the current state of the process you are profiling. The states include `running`, `paused`, `terminated`, `exited`, and `finished`.
Process ID	List the process ID for the program you are profiling.
Created on	Lists the date that the executable was created.
Instrumentation Version	Lists the version of the CXpa profiling routines (cxpamon.o) linked into your program with a CXpa compiler option (`-cxpa`, `-cxpar`, `-cxpab`).

Info Executable

Buttons

Name	Action
OK	Closes this dialog.
Help	Displays a help page for this dialog.

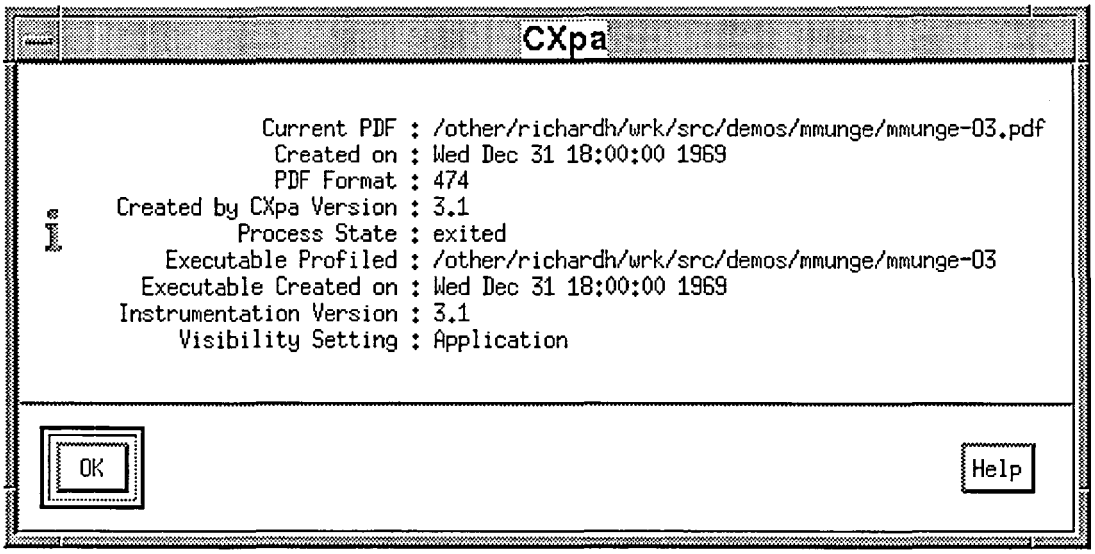
Context

The Info Executable dialog appears when you select the Executable option from the Info menu in the Executable Manager window.

Related Windows

Executable Manager window
Info Session dialog
Open PDF dialog
Info PDF dialog
New PDF dialog
Visibility Selection dialog

Info PDF



Description

The Info PDF dialog displays information on the performance data file (PDF).

Fields

Name	Meaning
Current PDF	Lists the name of the current performance data file (PDF).
Created on	Lists the date and time that the PDF was created.
PDF Format	Lists the format version number of the PDF.
Created by CXpa Version	Lists the version number of CXpa that created the PDF.
Process State	Lists the process state recorded in the PDF.
Executable Profiled	Lists the name of the executable you are profiling.
Executable Created on	Lists the date that the executable was created.

Instrumentation Version	Lists the version of the CXpa profiling routines linked to the executable when the PDF was created.
Visibility Setting	Lists the routines that are visible to your CXpa session. The possible values are Application, Library, or both.

Buttons

Name	Action
OK	Closes this dialog.
Help	Displays a help page for this dialog.

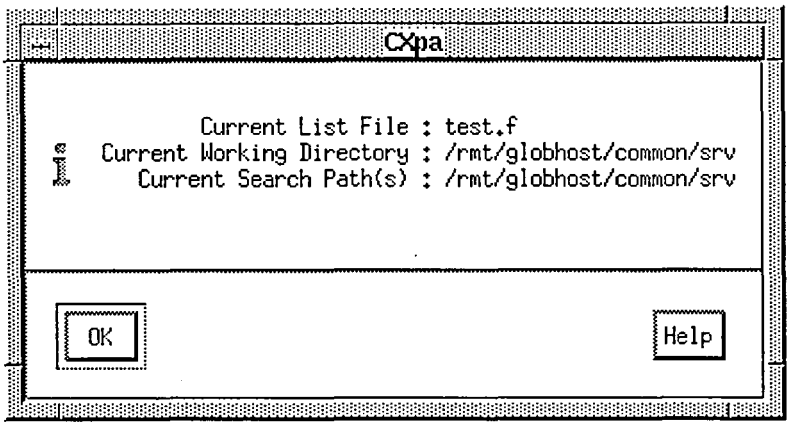
Context

The Info PDF dialog appears when you select the PDF option from the Info menu in the Executable Manager or Analysis Control window.

Related Windows

Analysis Control window
Executable Manager window
Info Executable dialog
Info Session dialog
New PDF dialog
Open PDF dialog
Visibility Selection dialog

Info Session



Description

The Info Session dialog displays information about your CXpa session.

Fields

Name	Meaning
Current List File	Lists the name of the file used when you display the Source Code window.
Current Working Directory	Lists the current directory.
Current Search Path(s)	Lists the directories CXpa uses to find source files to display in the Source Code window.

Buttons

Name	Action
OK	Closes this dialog.
Help	Displays a help page for this dialog.

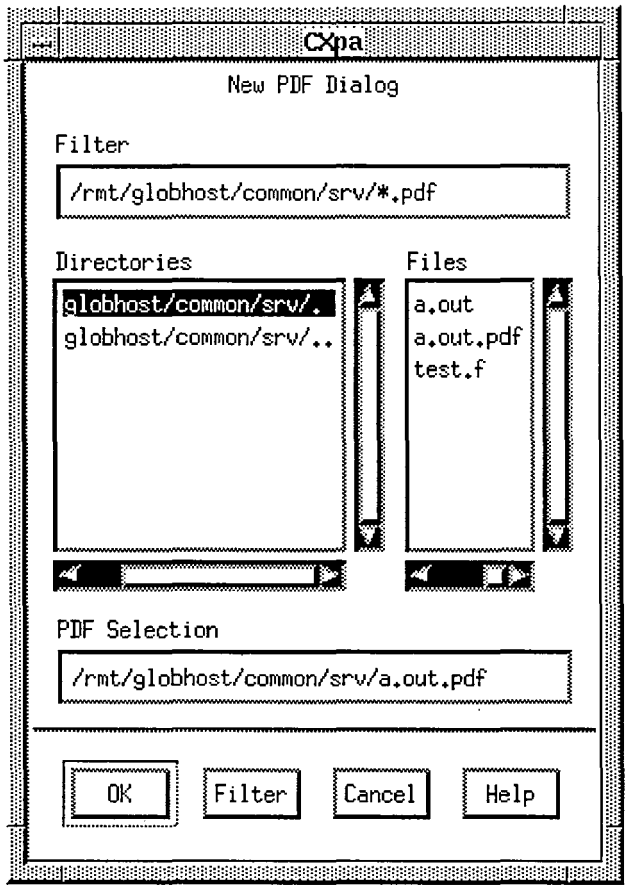
Context

The Info Session dialog appears when you select the Session option from the Info menu in the Executable Manager or Analysis Control window.

Related Windows

Analysis Control window
Info Executable dialog
Executable Manager window
Info PDF dialog

New PDF



Description

The New PDF dialog allows you to choose the name of the performance data file (PDF). A PDF is a file in which CXpa stores performance data for a single run of your program. CXpa uses the data in a PDF to create 2D and 3D Profile graphs and textual performance reports.

If you have invoked CXpa with the name of an executable file, when you run your program, CXpa overwrites all of the data in the PDF. If you do not want this to occur, you must change the name of the PDF between runs of your program using the New PDF dialog.

If the file you specify does not exist, CXpa creates it. If you do not set the PDF name, CXpa uses the default PDF name of *<executable>*.pdf, where *<executable>* is the name of the executable file for your program.

Filtering files

When the file selection dialog appears, the Filter field will contain the path to the current directory with *.pdf appended to it. Use this feature to find only files (from all files in the Files area) that match the search pattern shown in the filter field. To use the filter feature:

1. Alter the path name in the Filter field by typing in the field or by clicking on one of the directories. You can use wildcards (* for multiple matches or ? for a single match).
2. Press the Filter button. The Files list displays files and the Directories list displays directories that match the specification in the Filter field (usually *.pdf).

Selecting a file

You can select a PDF in the following ways:

- Double-click on a file name in the Files list.
- Highlight a file in the Files list and press the OK button.
- Type the full path name to the file in the PDF Selection field and press the OK button or press **RETURN**.

Fields

Heading	Meaning
Filter	Shows a path name, usually containing wildcards, that you set to determine what files appear in the Files and Directories lists.
Directories	Lists all subdirectories in the directory specified in the Filter field. You can change the directory to be searched by selecting the desired directory. You can double-click on a directory to search for all files that match the search pattern.
Files	Lists all files and/or subdirectories that match the search pattern in the Filter field.
PDF Selection	Contains the file name to use as the PDF. Collected profiling data is placed in this file.

Buttons

Name	Action
OK	Accepts the file name in the PDF selection field as the new PDF and closes dialog.
Filter	Displays in the Files area the directories and files that match the Filter directory.
Cancel	Closes this dialog without making any changes.
Help	Displays a help page for this dialog.

Context

The New PDF dialog appears when you select the New PDF option from the File menu on the CXpa Executable Manager window.

Use the New PDF dialog when you do not want to use the default PDF name of <*executable*>.pdf.

Related Windows

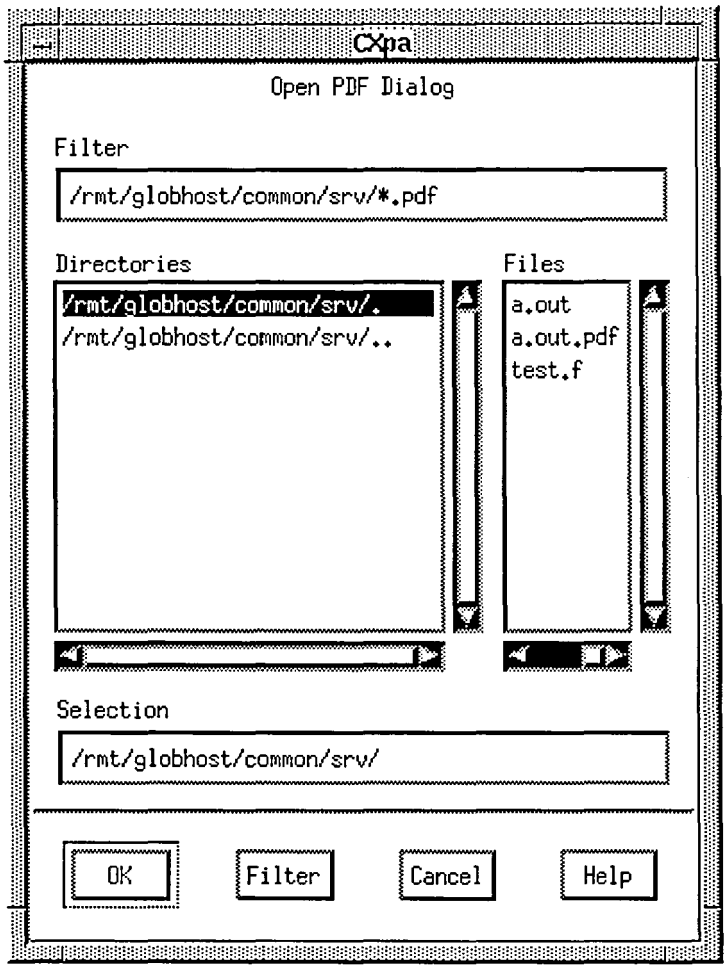
Analysis Control window
Executable Manager window
Filter Profile dialog
Open PDF dialog

Related Topics

Analyzing PDFs only
Setting the PDF

New PDF

Open PDF



Description

The Open PDF dialog allows you to add a PDF to the PDF list in the Analysis Control window.

A PDF is a file that CXpa uses to store performance data for a single run of your program. CXpa uses the data in the PDF to calculate the performance information that appears when you display a performance report or a 2D or 3D profile graph.

You can select PDFs created in a previous CXpa session, including PDFs created on other architectures or PDFs created from different executables.

Filtering files

When the Open PDF dialog appears, the Filter field contains the path to the current directory with *.pdf appended to it. Use this feature to find only files (from all files in the Files area) that match the search pattern shown in the filter field. To use the filter feature:

1. Alter the path name in the Filter field by typing in the field or by clicking on one of the directories. You can use wildcards (* for multiple matches or ? for a single match).
2. Press the Filter button. The Files list displays files and the Directories list displays directories that match the specification in the Filter field (usually *.pdf).

Selecting a file

You can select a PDF in the following ways:

- Double-click on a file name in the Files list.
- Highlight a file in the Files list and press the OK button.
- Type the full path name to the file in the PDF Selection field and press the OK button or press **RETURN**.

Fields

Heading	Meaning
Filter	Shows a path name, usually containing wildcards, that you set to determine what files appear in the Files and Directories lists.
Directories	Lists all subdirectories in the directory specified in the Filter field. You can change the directory to be searched by selecting the desired directory. You can double-click on a directory to search for all files that match the search pattern.
Files	Lists all files and/or subdirectories that match the search pattern in the Filter field.
PDF Selection	Contains the file name to use as the PDF. Collected profiling data is placed in this file.

Buttons

Name	Action
OK	Adds the file name in the PDF Selection field to the PDF list in the Analysis Control window.
Filter	Displays in the Files area the directories and files that match the Filter directory.
Cancel	Closes this dialog without making any changes.
Help	Displays a help page for this dialog.

Context

The Open PDF dialog appears when you select the Open PDF option from the File menu on the CXpa Analysis Control window.

Related Windows

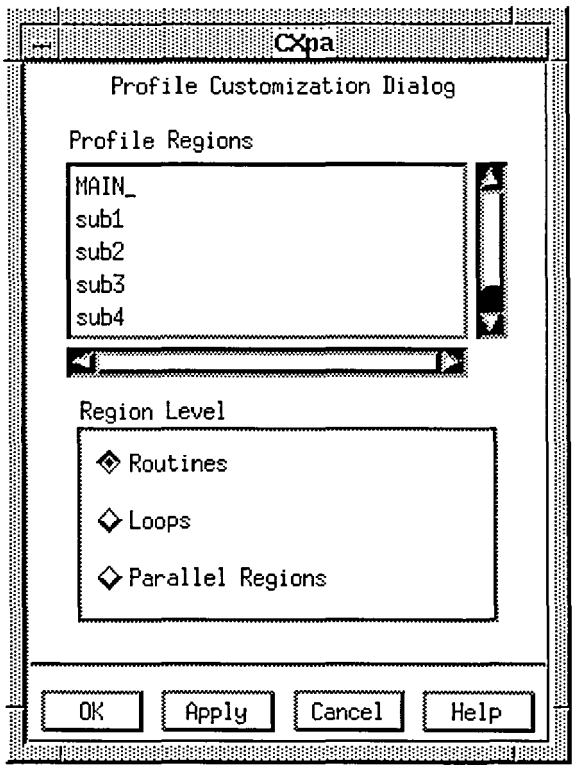
Analysis Control window
Filter Profile dialog
Info PDF dialog
New PDF dialog

Related Topics

Analyzing PDFs only
Setting the PDF

Open PDF

Profile Customization



Description

The Profile Customization dialog allows you to create a customized 2D or 3D profile graph for specific routines at a selected region level (routine, loop, or parallel region).

To display a customized profile graph:

1. Choose the region type for which you want to customize a profile. (You can select only one.)
2. Select the routines in the Profile Regions column that you want to include in the graph by highlighting them with the mouse. (You can select multiple routines.)
3. Press the Apply or OK button to display the new graph in the 2D or 3D Profile window.

Profile Customization

Fields

Heading	Meaning
Profile Regions	Allows you to select the routines whose performance information you want to include in the profile graph.
Region Level	Allows you to select the type of source code region for which you want to graph performance data.

Buttons

Name	Action
OK	Accepts the changes you have made, closes the dialog, and applies the changes to the graph.
Apply	Applies the changes to the profile without closing the dialog.
Cancel	Does not apply any of the changes you have made and closes the dialog.
Help	Displays a help page for this dialog.

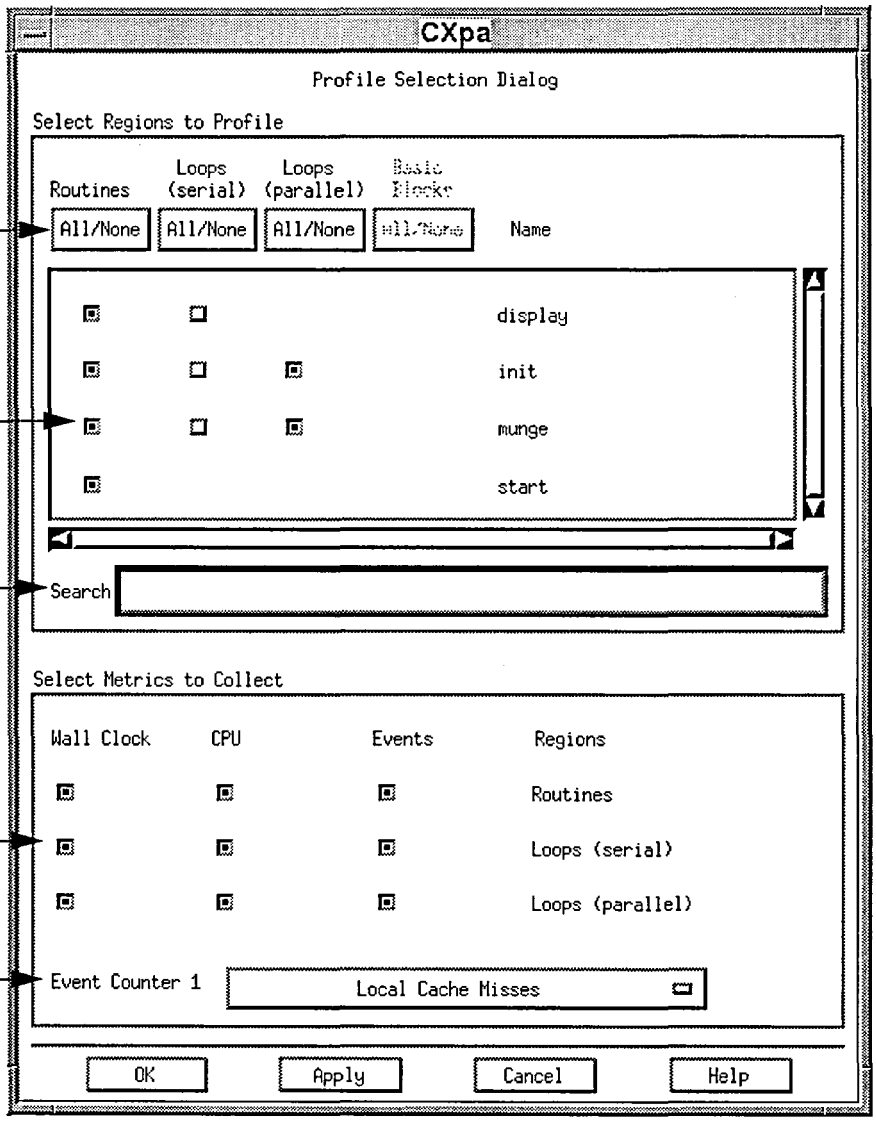
Context

The Profile Customization dialog appears when you press the Customize button on the 2D or 3D Profile windows.

Related Windows

2D Profile window
3D Profile window
Analysis Control window
Executable Manager window
Profile Selection dialog

Profile Selection



Click All/None buttons to select or deselect all routines containing the indicated type of source code region

Click toggle buttons to select/deselect regions in specific routines

Enter a routine name to search for (you can use * or ? wildcards)

Click toggle buttons to select/deselect metrics to collect

Select the type of event to collect for a single run of your program (SPP and C4 Series only)

Description

Use the Profile Selection dialog to select or deselect source code regions for profiling and to specify the types of metrics to collect for the these regions during profiling.

You can select all available source code regions in your program or a subset. You do not have to recompile your program to select or deselect regions for profiling.

Typically, you will want to select all routine regions the first time you profile your program (this is the default). This provides a complete picture of your program's performance.

After you have identified the routines whose performance you want to improve, you may want to select only those specific routines for profiling. With fewer source code regions selected, less time is spent in the timing routines CXpa uses to collect performance data. Selecting code regions for profiling does increase wall-clock time. Profiling time increases with the number of regions selected for profiling. In general selecting loop regions for profiling increases execution time more than selecting routine regions.

Selecting regions and metrics for profiling in all routines

From the Profile Selection dialog, perform the following steps to select regions and metrics for profiling in all routines:

1. Make sure that all routines are selected in the Select Regions to Profile section of the dialog (by default, all routines are selected when you first bring up the Profile Selection dialog). Toggle the Routines All/None button for routine regions to make sure that all routines are selected
2. Click the All/None toggle buttons for the types of source code regions you wish to profile in all routines. You can:
   - Perform routine-level analysis only for all routines (the default)
   - Profile all serial loops in all routines
   - Profile all parallel loops in all routines
   - Profile all basic blocks in all routines

   If a toggle button is not displayed for a particular region type, it means that no regions of that type can be profiled for that routine. For example, loop regions are not available for profiling unless your program is compiled at optimization level `-O1` or greater with the `-cxpa` option.
3. Specify the metrics that you want to collect by clicking the toggle buttons in the Select Metrics to Collect section of the Profile Selection dialog. CXpa collects these metrics at the regions of your program that are selected for profiling. You can choose to collect:
   - Wall clock time
   - CPU time
   - Events metrics (SPP Series and C4 systems only)

Events metrics are only available for SPP and C4 Series systems. If you wish to collect events metrics, you must first choose Events as one of the metrics in the Select Metrics to Collection section of this dialog.

Only one type of event can be collected per run of your program. The Event Counter options menu shows the type of event currently selected.

Refer to the "Introducing metrics" online help topic or section in this book for a discussion of available events metrics for C4 and SPP Series systems.

4. If you have chosen Events as one of the metrics in the Select Metrics to Collect section of the Profile Selection dialog, the Event Counter options menu is now available for you to select an event type.

   For both SPP Series machines, the default event type selected is Local Cache misses; for C4 Series machines, the default is Data Cache Misses.

5. Click and hold down the left mouse button on the Event Counter options menu to display a list of available event types. While holding down the left mouse button, position the mouse cursor over the desired event type, and then release the mouse button.

6. Press the OK button to apply the changes and close the Profile Selection dialog or press the Apply button to apply the changes without closing the dialog.

Selecting regions and metrics in individual routines

To select specific region types for profiling in individual routines:

1. Toggle the All/None button to deselect all routine regions.

2. Select and/or deselect the types of regions you want to enable in specific routines by clicking the toggle buttons to the left of the routine name.

   If your program contains a large number of routines, you can type the name of a routine in the Search field, then press **RETURN** to scroll the routine list so that the desired routine is displayed.

   If a toggle button is not displayed for a particular region type, it means that no regions of that type can be profiled for that routine. For example, loop regions are not available for profiling unless your program is compiled at optimization level `-O1` or greater with the `-cxpa` option.

3. Choose the metrics you want to collect at the regions of your program selected for profiling by clicking the appropriate toggle buttons in the Select Metrics to Collect section of the dialog.

You can collect:

- CPU time
- Wall clock time
- Events (SPP Series and C4 Series systems only)

4. If you have chosen to collect Events, select the type of event you want to collect for this run of your program from the Event Counter options menu.
5. Press the OK button in the Profile Selection dialog to apply the settings and close the dialog or Click the Apply button to apply the settings without closing the dialog.

Fields

Label	Meaning
Select Regions to Profile	Allows you to select regions of your program to profile.
Select Metrics to Collect	Allows you to choose metrics to collect for the regions of your program selected for profiling.
Name	Lists names of routines in your program that contain regions that can be selected for profiling. These are listed in alphabetical order.
Search	Allows you to search for a particular routine. Expressions with no wildcards will search for literal matches. Available wildcards include question marks (?) as a single match wildcard and asterisk (*) as a multiple match wildcard. To show search results, CXpa scrolls the list so that the first matching routine is at the top.
Event Counter	(SPP Series and C4 Series systems only) Displays an options menu where you can select the type of hardware event to collect. The types of events you can select differ between SPP Series and C4 Series system. Refer to the "Introducing Metrics" and "Selecting metrics in X window mode" online help topics or sections in this book for a description of event metrics available on each architecture.

Buttons

Name	Action
All/None	Selects/deselects all routines in your program that contain the indicated type of source code region (routines, loops, parallel loops, or basic blocks).
OK	Applies any changes you have made and closes this dialog.
Apply	Applies any changes you have made without closing the dialog.
Cancel	Closes this dialog without making any changes.
Help	Displays a help page for this dialog.

Context

The Profile Selection dialog appears when you click the Profile Selection button in the Executable Manager window.

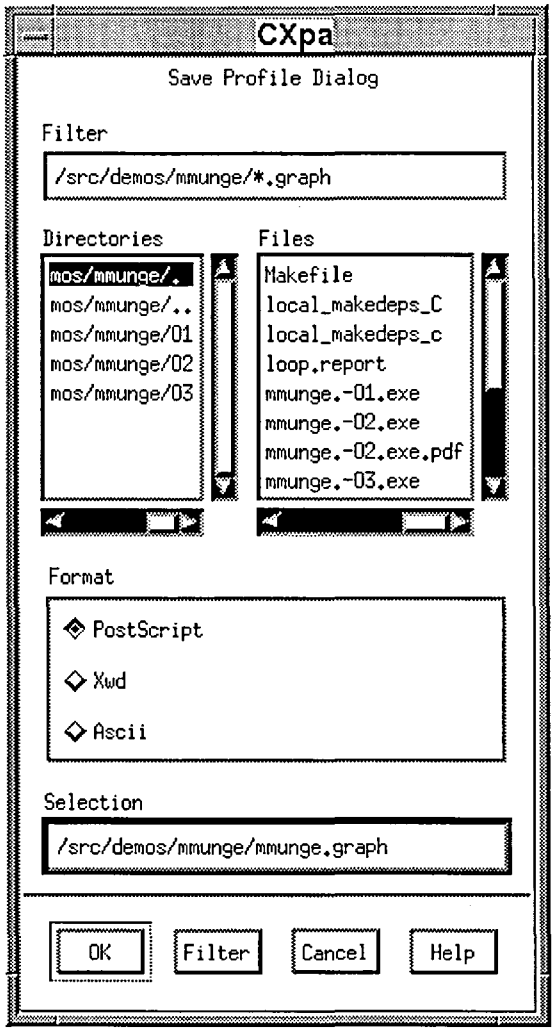
Related Windows

Analysis Report window
Executable Manager window
Profile Customization dialog
Visibility Selection dialog

Related Topics

Introducing metrics
Introducing source code regions
Selecting metrics in X window mode
Selecting regions in X window mode

Save Profile



Description

The Save Profile dialog allows you to save the graph in the 2D Profile or 3D Profile windows to a PostScript, ASCII, or Xwd file. These files are immediately available for use by other utilities that accept these formats, as shown below.

File format	Can be used with shell command	Can be imported to
PostScript	`lpr`	Document publishing or word processing applications
Xwd	`xwud, xpr`	Document publishing or word processing applications
ASCII	`lpr`	Spreadsheet or graphic applications

Filtering files

When the file selection appears, the Filter field will contain the path to the current directory with *.profile appended to it. Use this feature to find only files (from all files in the Files area) that match the search pattern shown in the filter field. To use the filter feature:

1. Alter the path name in the Filter field by typing in the field or by clicking on one of the directories. You can use wildcards (* for multiple match or ? for single match).
2. Press the Filter button. The Files list displays files, and the Directories list displays directories that match the specification in the Filter field (usually *.profile).

Selecting a file

You can select a file in the following ways:

- Double-click on a file name in the Files list.
- Highlight a file in the Files list and press the OK button.
- Type the full path name to the file in the Selection field and press the OK button or press **RETURN**.

Fields

Heading	Meaning
Filter	Shows a path name, usually containing wildcards, that you set to determine what files appear in the Files and Directories lists.

Directories	Lists all subdirectories in the directory specified in the Filter field. You can change the directory to be searched by selecting the desired directory. You can double-click on a directory to search for all files that match the search pattern.
Files	Lists all files and/or subdirectories that match the search pattern in the Filter field.
Selection	Contains the file name to save the profile to.

Buttons

Name	Action
OK	Saves the profile in the selected format to the file name in the Selection field.
Filter	Displays in the Files area the directories and files that match the Filter directory.
Cancel	Closes this dialog without making any changes.
Help	Displays a help page for this dialog.

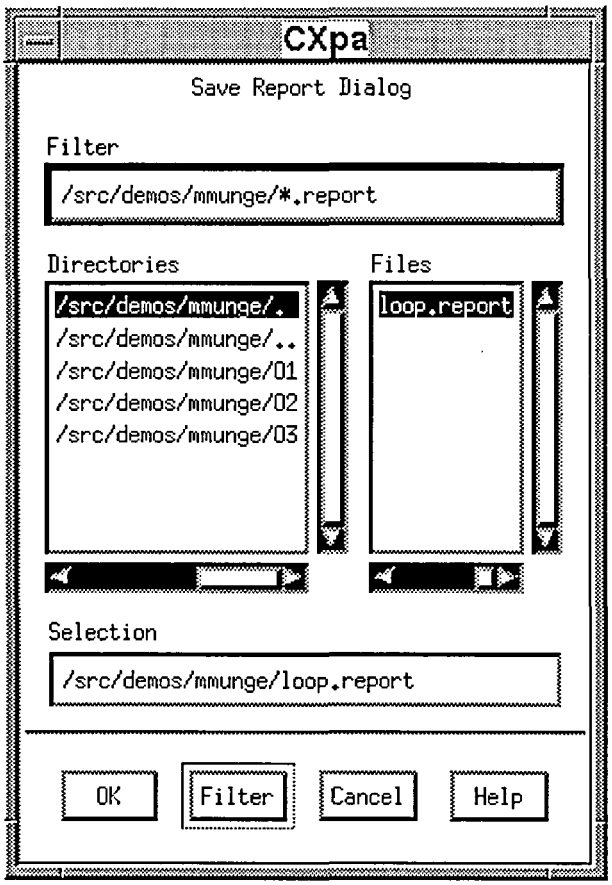
Context

The Save Profile dialog appears when you select the Save Profile option from the File menu on the 2D Profile or 3D profile windows.

Related Windows

2D Profile window
Filter Profile dialog
3D Profile window
Profile Customization dialog

Save Report



Description

The Save Report dialog allows you to save the report in the Analysis Report window to an ASCII file.

Filtering files

When the dialog appears, the Filter field will contain the path to the current directory with *.report appended to it. Use this feature to find only files (from all files in the Files area) that match the search pattern shown in the filter field.

To use the filter feature:

1. Alter the path name in the Filter field by typing in the field or by clicking on one of the directories. You can use wildcards (* for multiple matches or ? for a single match).
2. Press the Filter button. The Files list displays files and the Directories list displays directories that match the specification in the Filter field (usually *.report).

Selecting a file

You can select a file in the following ways:

- Double-click on a file name in the Files list.
- Highlight a file in the Files list and press the OK button.
- Type the full path name to the file in the Selection field and press the OK button or press **RETURN**.

Fields

Heading	Meaning
Filter	Shows a path name, usually containing wildcards, that you set to determine what files appear in the Files and Directories lists.
Directories	Lists all subdirectories in the directory specified in the Filter field. You can change the directory to be searched by selecting the desired directory. You can double-click on a directory to search for all files that match the search pattern.
Files	Lists all files and/or subdirectories that match the search pattern in the Filter field.
Selection	Contains the file name to save the report to.

Buttons

Name	Action
OK	Saves the report to the file name in the Selection field.
Filter	Displays in the Files area the directories and files that match the Filter directory.
Cancel	Closes this dialog without making any changes.
Help	Displays a help page for this dialog.

Context

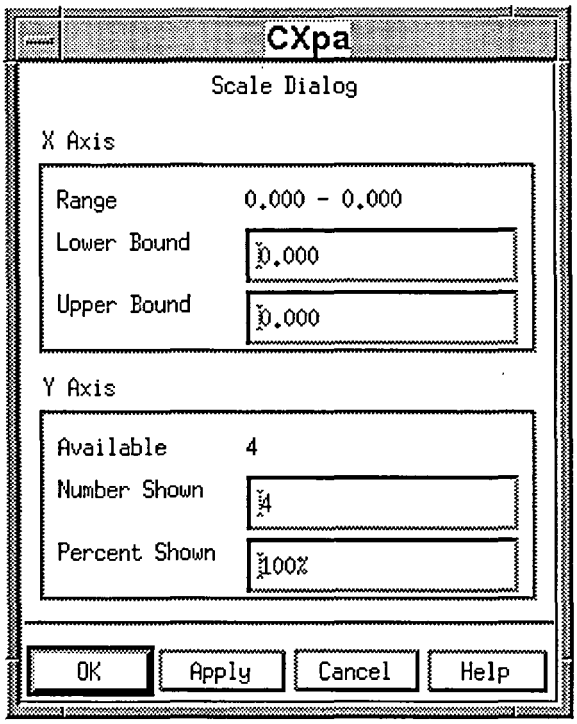
The Save Report dialog appears when you select the Save Report option from the File menu on the Analysis Report window.

Related Windows

Analysis Report window
Customize Report dialog
Filter Report dialog

Save Report

Scale



Description

Use the Scale dialog to view or specify the range (X axis) or number/percentage (Y axis) of data values displayed at one time in the 2D Profile window. This is especially useful in cases where there are a large number of data items to graph and you want to focus on a subset of the data.

X-Axis scaling

The X Axis panel shows the range of X axis data values that are currently displayed in the 2D graph. You can limit the range of items displayed at one time in the window by specifying a lower and/or upper bound. When you scroll the X axis, the Lower Bound and Upper Bound fields update to reflect the range of values displayed in the graph.

Y-Axis scaling

The Y Axis panel displays the total number of available regions to graph on the Y axis. You can limit the number of Y-axis items displayed at one time in the window by specifying either a number or a percentage to display. If you change the number shown, the Percent Shown field reflects the change; if you change the percentage of values shown, the Number Shown field reflects the changes.

X Axis

Field	Description/Valid values
Range	Lists the range of data values that can be shown in the graph.
Lower Bound	Sets the lower-bound of the X axis. This value must be less than the value specified in the Upper Bound field but can be smaller than the minimum data value.
Upper Bound	Sets the upper-bound of the X axis.This value must be greater than the value specified in the Lower Bound field but can be larger than the maximum data value.

Y Axis

Name	Description/Valid values
Available	Lists the number of available regions in the graph.
Number Shown	Shows the number of regions currently displayed. Valid values are integers from one to the number available. Values larger than the number available are ignored.
Percent Shown	Shows the percentage of available regions currently displayed on the Y axis. Valid values are percentages from 0 to 100. Values larger than 100 are ignored.

Buttons

Name	Action
OK	Scales the graph with the chosen options and closes this dialog.
Apply	Scales the graph with the chosen options. The dialog remains.

Cancel	Closes this dialog without making any changes.
Help	Displays a help page for this dialog.

Context

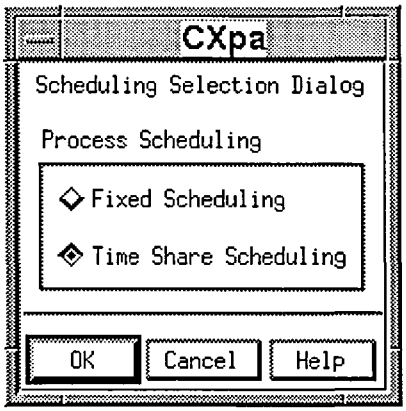
The Scale dialog appears when you choose the Scale option from the View menu of the 2D Profile window.

Related Windows

2D Profile window
Profile Customization dialog
Visibility Selection dialog
Filter Profile dialog
3D Profile window

C Series only

Scheduling Selection



Description

The Scheduling Selection dialog allows you to specify whether your process runs with fixed scheduling or time share scheduling. The default is time share scheduling.

Fixed scheduling ensures all CPUs on a machine are made available for each timeslice of a process when execution begins. This gives you the closest approximation to a standalone system in a time share environment. Enabling fixed scheduling only guarantees that all CPUs are made available when process execution begins, not that they will be used.

The current setting for process scheduling is displayed in the Scheduling field in the Program Information area on the Executable Manager window.

You can also enable fixed scheduling by specifying the `-f` option when you invoke CXpa from the shell.

Scheduling Selection

Fields

Heading	Meaning
Fixed Scheduling	When selected, your process runs with fixed scheduling.
Time Share Scheduling	When selected, your process runs with time share scheduling. This is the default.

Buttons

Name	Action
OK	Accepts the new setting.
Cancel	Closes this dialog without making any changes.
Help	Displays a help page for this dialog.

Context

The Scheduling Selection dialog appears when you choose the Scheduling option from the Options menu on the Executable Manager window.

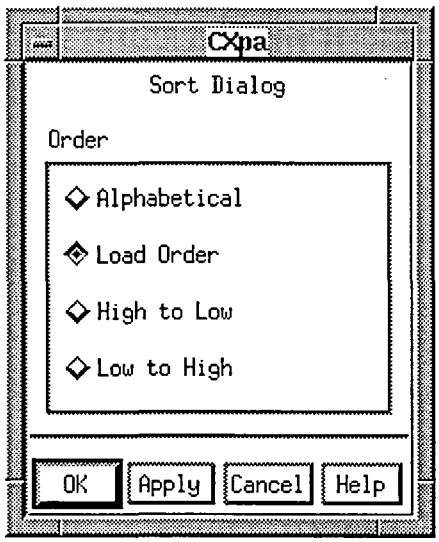
Related Windows

Executable Manager window

Related Commands

`cxpa`

Sort



Description

The Sort dialog allows you to sort the data in the 2D and 3D profile graphs by clicking on a sort order option and pressing the OK or Apply button.

You can sort the data alphabetically, by load order, or by the data values for the type of metrics displayed (from highest to lowest or from lowest to highest). The default is from highest to lowest. Routine names are sorted alphabetically by character strings.

Order

Lists the sorting choices.

Alphabetical	Sorts the data alphabetically.
Load order (default)	Lists the data in the order that the routines were given to the link editor.
High to Low	Sorts the data from highest to lowest.
Low to High	Sorts the data from lowest to highest.

Sort

Buttons

Name	Action
OK	Sorts the graph in the chosen order and closes this dialog.
Apply	Sorts the graph is the chosen order.
Cancel	Closes this dialog without making any changes.
Help	Displays a help page for this dialog.

Context

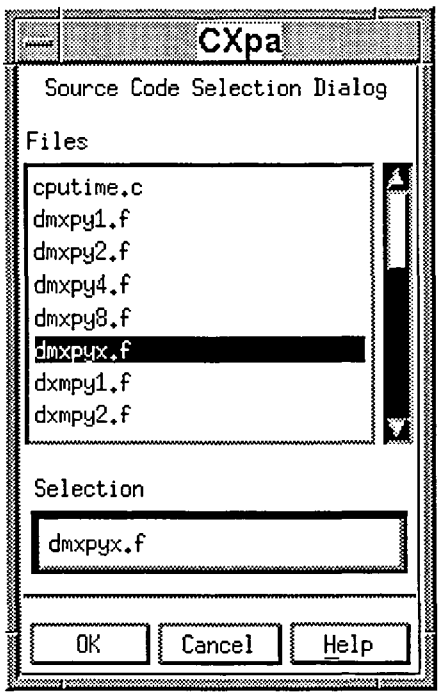
The Sort dialog appears when you choose the Sort option from the View menu in the 2D or 3D Profile windows.

Related Windows

2D Profile window

3D Profile window

Source Code Selection



Description

The Source Code Selection dialog allows you to choose which source file to display in the Source Code window.

Changing source files

To display another source file in the Source Code window:

1. Click the name of the source file that you want to display. The selected file name appears in the Selection field.
2. Click the OK button. The dialog disappears and the new source file appears in the Source Code window.

If CXpa cannot find the source file, change the search path by choosing the Source Search Path option from the Options menu.

NOTE: You cannot enter a full path name in the Selection field. To change paths:

1. **Add the desired directory path in the Source Search Path dialog and click the OK button.**
2. **Type the file name in the selection field of the Source Code Selection dialog.**

Fields

Heading	Meaning
Files	Lists all the source files used to create the executable file.
Selection	Lists the file that you want to display in the Source Code window

Buttons

Name	Action
OK	Accepts the file name in the selection field as the new source file to display and closes this dialog.
Cancel	Closes this dialog without making any changes.
Help	Displays a help page for this dialog.

Context

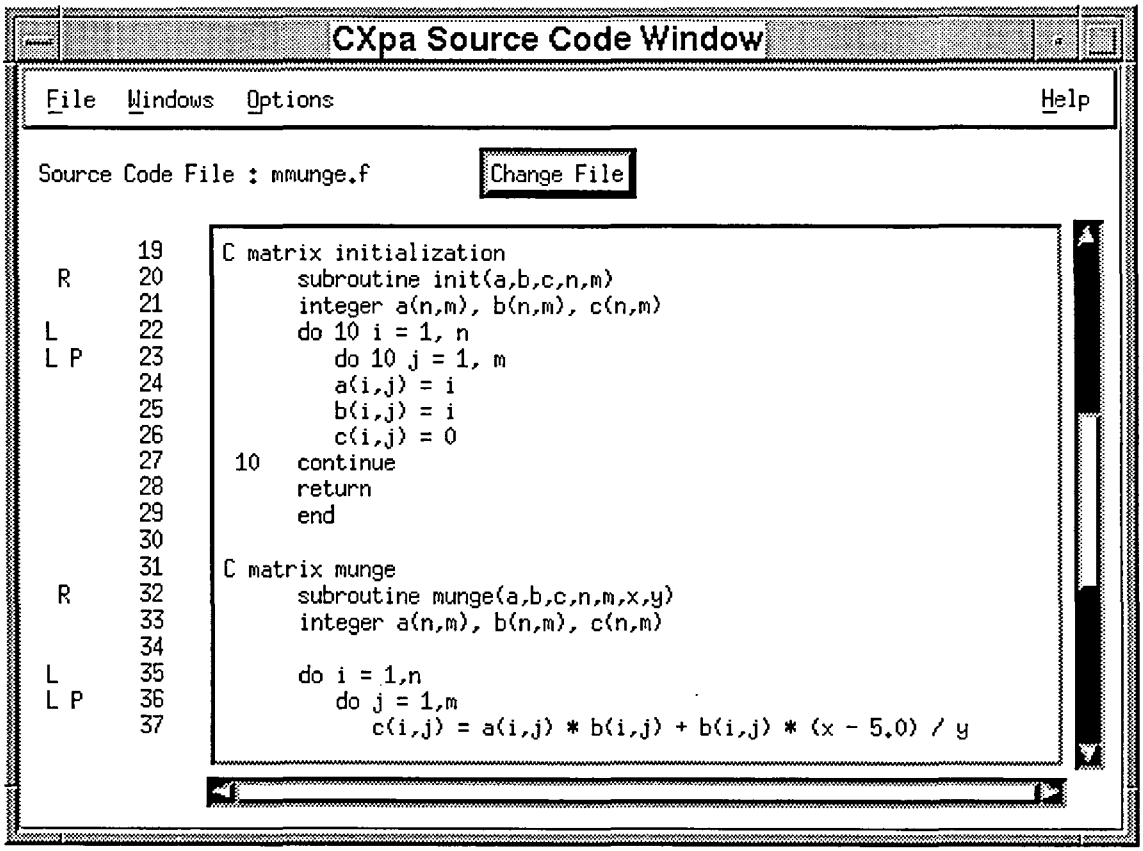
The Source Code Selection dialog appears when you press the Change File button on the Source Code window.

Related Windows

Source Search Path dialog

Source Code window

Source Code window



Description

The Source Code window displays the source code for your program. By default, the Source Code window displays the file that contains your main routine. You can open multiple Source Code windows displaying different files in your program or use the Change File button to display a different source file in the current window.

Annotations

To the left of the line numbers, CXpa displays annotations that identify source code regions that can be profiled. Uppercase letters indicate source code regions currently selected for profiling. Lowercase letters indicate source code regions that are not selected for profiling.

- R, r—Indicates routine regions.
- L, l—Indicates loop regions.
- P, p—Indicates parallel region regions.
- B, b—Indicates basic block regions.

The => symbol to the right of a line number indicates the beginning of a section of code that corresponds to a bar in the 2D Profile or 3D Profile window that you clicked to display source code.

Changing source files

To display another source file in this window:

1. Click the Change File button. The Source Code Selection dialog appears.
2. Click the name of the source file that you want to display. The selected file name appears in the Selection field.
3. Click the OK button. The dialog disappears and the new source file appears in the Source Code window.

   If CXpa cannot find the source file, change the search path by choosing the Search Path option from the options menu.

Menus

Name	Meaning
File	Contains an option to close the window.
Windows	Contains options to open additional CXpa windows for viewing 2D and 3D profile graphs, performance reports, or source files.
Options	Contains Source Search Path option
Help	Contains various help options that invoke the CXpa help system.

Buttons

Name	Meaning
Change File	Displays a dialog to choose another source file to display in the Source Code window.

Context

The Source Code window appears when you:

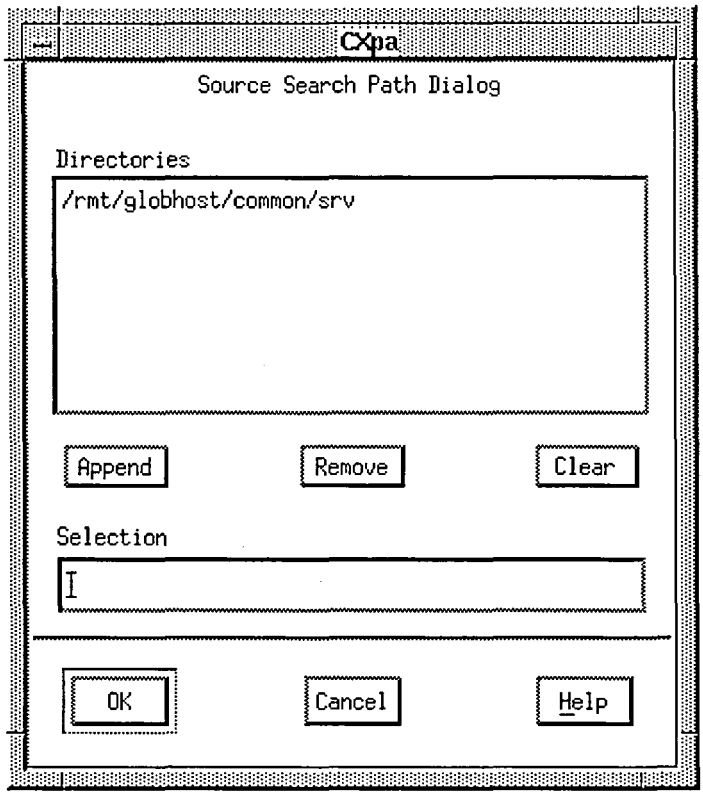
- Select the Source Code option from the Windows menu in any CXpa window
- Click on a bar in the graph of the 2D or 3D Profile windows

Related Windows

2D Profile window
3D Profile window
Analysis Control window
Analysis Report window
Executable Manager window
Source Search Path dialog
Source Code Selection dialog

Source Code window

Source Search Path



Description

The Source Search Path dialog allows you to change CXpa's search path. CXpa uses its search path to find source files when you list a source file (by clicking on graphs in the 2D or 3D Profile windows or by using the Source Code window). CXpa uses the location of the source files embedded in the executable by the compiler to look for source files, so you will need to modify the search path if you have moved the source files after compiling them.

Adding a directory to CXpa's search path

To add a directory to CXpa's search path:

1. Enter a directory path name in the Selection field.

2. Press the Append button. The path name appears in the Directories list.
3. Press the OK button to apply this change and close the dialog.

Removing a directory from CXpa's search path

To remove a directory from CXpa's search path:

1. Highlight the path name to remove in the Directories list.
2. Press the Remove button.
3. Press the OK button to apply this change and close the dialog.

Fields

Heading	Meaning
Directories	Lists the directories in CXpa's search path.
Selection	Allows you to enter a directory name to add to or delete from CXpa's search path.

Buttons

Name	Action
Append	Adds the directory in the Selection field to the Directories list.
Remove	Removes a highlighted directory from the Directories list.
Clear	Removes all the directories from the list.
OK	Accepts the directories in the Directories list as CXpa's search path and closes this dialog.
Cancel	Closes this dialog without changing CXpa's search path.
Help	Displays a help page for this dialog.

Context

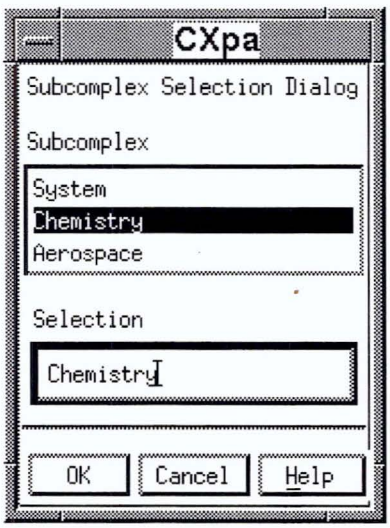
The Source Search Path dialog appears when you select the Source Search Path option from the Options menu on the Source Code, Executable Manager, or Analysis Control windows.

Related Windows

Analysis Control window
Executable Manager window
Source Code Selection dialog
Source Code window

SPP only

Subcomplex Selection dialog



Description

The Subcomplex Selection dialog allows you select the name of the subcomplex on which you want to execute your program. When you start CXpa, it queries the system for the number of subcomplexes configured and only makes this dialog available if there is more than one.

A *subcomplex* is a collection of processors and memory from one or more nodes of an SPP Series system (which define the boundary within which all threads belonging to a process execute).

The default value is the subcomplex from which you invoked CXpa. You only need to use this dialog if you want to run your program on a different subcomplex. You must have the appropriate permissions to run the process on the specified subcomplex.

Subcomplex Selection dialog

The names of all subcomplexes on your system are displayed in the Subcomplex field. To select a different subcomplex:

1. Click with the mouse to highlight the subcomplex you want to select. Your selection is then displayed in the Selection field. You can also type in the name of the subcomplex you want to select in this field
2. Click OK to select the new subcomplex and close the dialog.

For more information on subcomplexes on SPP Series systems, refer to the scm(1) and scm(4) man pages or the Convex *SPP-UX System Administration Guide* (DSW-853) or contact the system administrator at your site.

Fields

Field	Meaning
Subcomplex	Lists the names of all subcomplexes on your system.
Selection	When you first invoke the Subcomplex Selection dialog, this field is contains the name of the subcomplex from which you invoked CXpa. Otherwise, it displays the currently selected subcomplex.

Buttons

Name	Action
OK	Selects the subcomplex displayed in the Selection field and closes the dialog.
Cancel	Closes this dialog without applying the changes.
Help	Displays a help page for this dialog.

Context

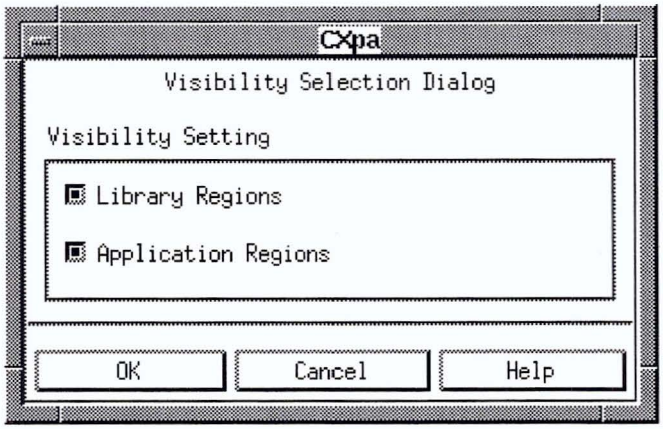
The Subcomplex Selection dialog appears when press the Subcomplex Selection button on the Executable Manager window. The Subcomplex Selection button is not available if there is only one subcomplex configured on your system.

CXpa queries the system for available subcomplexes at start-up, so if a subcomplex is added during a CXpa session, it will not be visible to CXpa during that session.

Related Windows

Executable Manager window

Visibility Selection



Description

The Visibility Selection dialog allows you choose whether application and/or library routines are visible to a CXpa analysis or profiling session.

When a routine is *visible* to CXpa, it means that it can be selected for profiling and that data collected during profiling can be viewed and analyzed in performance graphs and reports.

The term *library routines* refers to CONVEX-supplied libraries that have been instrumented for use with CXpa (such as the C and Fortran libraries, the operating system libraries, and the mathematical libraries).

NOTE: Profiling data for system libraries instrumented for CXpa can only be collected and viewed if:

- **You compile your source files with the `-cxpalib` option and with CONVEX C V6.0 or greater or CONVEX Fortran V9.0 or greater, and**
- **The CXpa-instrumented libraries are installed on your system.**

Without these conditions, you can set visibility for library regions, but CXpa will only collect or analyze profiling data for application source regions.

When you invoke CXpa with an executable to bring up the Executable Manager window, the Visibility Selection setting determines which routines you can select.

Visibility Selection

When you invoke CXpa with a PDF to bring up the Analysis Control window, the Visibility Selection setting determines which routines appear in the performance reports and graphs.

By default, visibility is set for both application and library regions so that system library routines and application routines are visible during profiling and analysis.

To change the visibility setting for library and/or application routines:

1. Click the regions that you want to include or exclude.
2. Click the OK button to apply the settings and close the dialog.

Fields

Heading	Meaning
Library Regions	When selected, makes the CXpa-instrumented system library routines linked with your program visible to CXpa during profiling or analysis.
Application Regions	When selected, makes the CXpa-instrumented routines in your program visible to CXpa during profiling or analysis.

Buttons

Name	Action
OK	Accepts the new settings.
Cancel	Closes this dialog without making any changes.
Help	Displays a help page for this dialog.

Context

The Visibility Selection dialog appears when you choose the Visibility option from the Options menu on the Executable Manager window or the Analysis Control window.

Related Windows

2D Profile window
Analysis Control window
Executable Manager window
Profile Selection dialog
3D Profile window
Analysis Report window
Profile Customization dialog
Source Search Path dialog

Commands

6

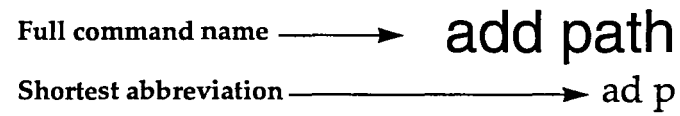
This chapter contains a reference page for each CXpa command. Commands can be entered in line mode (when you invoke CXpa with the `-nw` option) at the `(CXpa)` command prompt, used in CXpa script files, or executed in batch mode.

You can abbreviate CXpa commands. For example, the command `analyze cpu loop` can be abbreviated to `an c l`. The shortest unique abbreviation for each command is shown in the upper right corner of the first reference page for each command, immediately below the command name.

Each reference page displays its command name and shortest abbreviation at the top, followed by a one-line description. The rest of the page is divided into the following sections:

- **Syntax**—Lists the format rules for the command and its parameters.
- **Description**—Explains the purpose and functionality of the command.
- **Examples**—Shows one or more examples illustrating the use of the command.
- **Related Commands or Topics**—Lists CXpa commands or topics related to the command being described.

The heading at the top of each command description contains the following lines of information:



add path

ad p

Add directories to CXpa's search path for source files.

Syntax

add path *<directory-list>*

Parameter	Meaning
<directory-list>	Specifies one or more directories, separated by a space, to add to CXpa's search path.

Description

The `add path` command appends the specified list of directories to CXpa's search path. CXpa uses its search path to locate source files. By default, the search path contains the:

- Location of the original source files when the program was compiled
- Location of the executable and/or PDF
- Current working directory

CXpa uses the location of the source files embedded in the executable by the compiler to look for source files, so you will need use the `add path` command to modify the search path if you have moved the source files after compiling them.

Use the `info` command to list CXpa's current search paths. Use the `path` command to replace CXpa's current search path set.

You may find it convenient to place the `add path` command in CXpa's initialization file, .cxpainit, to automatically set CXpa's search path each time you invoke CXpa.

Examples

The following examples show typical uses of the `add path` command.

```
(CXpa) add path c_progs
(CXpa) info
```

(Skipping command output not relevant to this example.)

```
Current Search Path(s) : /mnt/user/progs/
                         /mnt/user/progs/c_progs
```

The above example of the `add path` command appends the relative path of `c_progs` to CXpa's search path.

```
1.(CXpa) list selectable SUB2
             File prog.f not found in search path.
2.(CXpa) info

(Skipping command output not relevant to this example.)

     Current Search Path(s) : /mnt/dev_tree/wrk

3.(CXpa) add path /mnt/dev_tree/progs
4.(CXpa) list selectable SUB2
       R  1 SUBROUTINE SUB2(MATRIX, VAR3)
      L P 5 DO I=1,5
      L   6    DO J=1,5
```

The above example shows a scenario for using the `add path` command when CXpa cannot find a source file you are trying to list. The line numbers are for reference only.

1. When the `list selectable SUB2` command is issued, CXpa cannot find the file that contains the routine `SUB2`.
2. The `info` command displays CXpa's current search path at the bottom of the `info` command's output.
3. The `add path` command adds /mnt/dev_tree/progs to CXpa's search path.
4. Now that CXpa's search path has been modified, CXpa finds the needed source file when the `list selectable SUB2` command is issued.

Related Commands

`info` `list`
`list selectable` `path`

analyze

an

Display performance reports for profiled source code regions.

Syntax

`analyze` [*<metric-list>*] [*<region-type>*] [*<routine-list>*] [*<i/o_redirection>*]

Parameter	Meaning

When parameters are used with this command, they must be specified in the order shown in the syntax statement above.

<metric-list>

Restricts the output of this command to the specified metrics. If no metrics are specified, all available metrics are displayed.

When used, this parameter should precede any other parameter. Separate multiple metrics with a space. Valid values are as follows:

`cpu`
`wall_clock`
`events` (C4 Series and SPP Series only)
`vector_flops` (C Series only)

<region-type>

Specifies the type of source code region for which you want to display reports. Valid values are `routine`, `loop`, `pregion`, and `block`. If you do not specify a region, reports are displayed for all profiled regions.

<routine-list>

For the selected source code region, specifies one or more routines. Separate multiple routines with a space. If you do not specify a *region-type* with the *routine-list* parameter, routine-level analysis reports are displayed.

<i/o_redirection>

Redirects this command's standard output or error to the specified file when you include one of the redirection operators (<, >, >>, >&, >>&).

Description

Use the `analyze` command to create and display textual performance reports. When you execute the `analyze` command without specifying any parameters, all available reports and metrics are displayed for all profiled regions in your program.

CXpa displays reports in line mode using the pager specified with your PAGER environment variable. If the PAGER environment variable is not set, CXpa uses the `more` command to page output. You can also redirect output to a file using redirection operators.

The `analyze` command generates the reports from the data in the performance data file (PDF), so a PDF must exist before you can display a performance report. CXpa creates a PDF when you select region types with a `select` command and execute the program with the `run` command. If you have not specified a PDF name in this profiling session, CXpa uses the default file name *<executable>*.pdf.

When you invoke CXpa with the name of a PDF file only, you can use the `analyze` command to look at reports from multiple PDFs or PDFs created in previous CXpa sessions (including PDFs created on different architectures). Then, use the `set pdf` command to change the name of PDF being analyzed.

NOTE: If you invoked CXpa with the name of an executable, you can only analyze PDFs that were created during the current CXpa session and generated from the executable you are currently profiling.

To generate reports for a specific source code region level, use the `analyze` command followed by a *region-type* parameter: `analyze block`, `analyze routine`, `analyze loop`, or `analyze pregion`.

You can use the *metric-list* and *routine-list* parameters to display reports for a subset of metrics and/or routines, respectively.

To view intermediate profiling results, press **CTRL-c** to pause the program and then using the `analyze` command.

Reports

The `analyze` command can display reports for the following source code regions under the listed conditions:

- **Routine reports**—Use the `analyze` or `analyze routine` command to display routine reports. Source files must be compiled with `-cxpa` or `-cxpar` and routine regions selected for profiling with the `select all` or `select routine` commands.
- **Loop reports**—Use the `analyze` or `analyze loop` command to display loop reports. Source files compiled with `-cxpa` at optimization level `-O1` or higher, and loop regions must be selected for profiling with the `select all` or `select loop` commands.
- **Parallel Region reports**—Use the `analyze` or `analyze pregion` command to display parallel region reports. Source files must be compiled with `-cxpa` at optimization level `-O3`, and parallel regions must be selected for profiling with the `select all` or `select pregion` commands.

- **Basic Block report**—Use the `analyze block` command to display a basic block report. Source files must be compiled with `-cxpab`, and block regions must be selected for profiling with the `select all` or `select block` commands.

For a detailed description of CXpa reports and the metrics displayed in each report, refer to the topics listed in the "Related Topics" section.

Examples

The examples in this section show how to use the `analyze` command.

```
1.(CXpa) select routine all
2.(CXpa) collect cpu wall_clock events
3.(CXpa) set events local_misses
4.(CXpa) run
   (Program runs to completion, and any output is displayed.)
5.(CXpa) analyze
   (Performance reports are displayed.)
```

The above scenario shows how you would normally use the `analyze` command in conjunction with other CXpa commands to generate performance analysis reports. The line numbers are for reference only.

1. The `select routine all` command tells CXpa to select all routine regions in your program for profiling.
2. The `collect cpu wall_clock events` command tells CXpa which metrics to collect. Events metrics are only available on C4 and SPP Series machines. In this case, CPU time, wall clock time, and events will be collected. You must then use the `set events` command to specify the type of event to collect.
3. The `set events local_misses` command tells CXpa to collect the number of times that a memory reference had to be satisfied from memory on the processor's node due to a miss in the processor's data cache. The `local_misses` parameter is only available on SPP Series machines.
4. The `run` command executes the program and initiates profile data collection. Profile data collection ends when the program runs to completion.
5. The `analyze` command calculates and displays all possible performance reports from data collected and stored in the PDF file.

The following examples show how to use the `analyze` command after regions and metrics have been selected for profiling and profiling data has been collected and stored in a PDF.

```
(CXpa) analyze events loop sub2
```

The above command creates and displays loop performance analysis reports for events metrics collected at the profiled loop regions executed in subroutine `sub2`. The order of the parameters is significant. The *metric-list* parameter (in this case, `events`) must be specified first, and the routine specifier `sub2` must follow the `loop` *region-type* parameter.

```
(CXpa) analyze loop > report_output
```

The above command creates loop performance analysis reports containing all metrics collected for all profiled regions in the program and redirects the output to a file named report_output. Any existing data in report_output is overwritten.

```
(CXpa) analyze events cpu
```

The above command creates and displays all available performance reports containing events and CPU time metrics collected for all profiled regions in the program.

```
(CXpa) analyze loop init display
```

The above command creates and displays loop performance reports containing all collected metrics for the profiled loop regions executed in subroutines `init` and `display`.

Related Commands

`deselect`
`rerun`
`run`
`select`
`set pdf`

Related Topics

Basic Block report
Introducing metrics
Introducing source code regions
Loop reports
Reports
Routine reports
Parallel Region reports

collect
col

Specify the metrics that you want to collect while profiling your program.

Syntax

`collect` *<metric-list>*

Parameter	Meaning
<metric-list>	Specifies one or more metrics to collect. The choices are: `cpu`—Collects CPU time and iteration/execution counts. `events`—(C4 Series and SPP Series only) Collects events metrics that you can configure with the `set events` command. `vector_flops`—(C Series only) Collects data on vector register utilization, including estimated average and peak vector Mflops, chime counts, and vector spills. `wall_clock`—Collects wall clock time.

Description

The `collect` command tells CXpa which metrics to collect at the regions of your program that have been selected for profiling. Each use of this command replaces the values previously set.

NOTE: Unless you have selected source code regions to profile with the `select` command, no region-level analysis will be possible.

NOTE: If you specify event collection with the `events` parameter of the `collect` command, you must use the `set events` command to specify the type of event collected. Only one type of event can be collected per run of your program.

Examples

The examples in this section show various ways to use the `collect` command, assuming your program is compiled appropriately for the regions selected for profiling.

```
1.(CXpa) select loop
2.(CXpa) collect cpu wall_clock events
3.(CXpa) set events local_misses
4.(CXpa) run
   (Program runs to completion, and any output is displayed.)
5.(CXpa) analyze
   (Performance reports are displayed.)
```

The above scenario shows how you would normally use the `collect` command in conjunction with other CXpa commands to specify the metrics you want to collect. The line numbers are for reference only

1. The `select loop` command tells CXpa to select all available loop regions in your program for profiling.
2. The `collect cpu wall_clock events` command tells CXpa which metrics to collect. Events metrics are only available on C4 and SPP Series machines. In this case, CPU time, wall clock time, and events will be collected. You must then use the `set events` command to specify the type of event to collect.
3. The `set events local_misses` command tells CXpa to collect the number of times that a memory reference had to be satisfied from local memory (memory on the same node as the processor) due to a miss in the processor's data cache. The `local_misses` parameter is only available on SPP Series machines.
4. The `run` command executes the program and initiates profile data collection. Profile data collection ends when the program runs to completion.
5. The `analyze` command calculates and displays all possible performance reports from data collected and stored in the PDF file.

```
(CXpa) select routine all
(CXpa) collect wall_clock
```

The above commands select all routine regions in your program for profiling and enable CXpa to collect wall clock time for routines.

Related Commands

`analyze`
`run`
`select`
`set events`
`set pdf`

Related Topics

Introducing metrics

Selecting metrics in line mode

continue

con

Continue profiling a paused program.

Syntax

```
continue
```

Description

The `continue` command resumes the profiling of a paused program.

When you continue profiling, the selected program resumes execution, and CXpa resumes profile data collection.

Examples

The following example shows a likely scenario in which you would use the `continue` command. The line numbers are for reference only.

```
1.(CXpa) run
2.(CXpa) CTRL-c
   (Pressing CTRL-c pauses program execution.)
3.(CXpa) analyze
   (Intermediate performance reports are displayed.)
4.(CXpa) continue
   (Program runs to completion, and data collection ends.)
5.(CXpa) analyze
   (Complete performance reports are displayed.)
```

The following lines explain the example above by number:

1. The `run` command runs the program and initiates profile data collection.
2. Pressing **CTRL-c** pauses program execution.
3. The `analyze` command displays the performance information that has been collected to this point.
4. The `continue` command resumes the program execution and data collection.
5. When the program has run to completion, CXpa is finished collecting profile data. The `analyze` command is used again to display the final profile results of this run.

Related Commands `run` `stop`

cxpa

Invoke the CONVEX performance analyzer.

Syntax

cxpa [*<executable>*] [**-f**] [**-help**][**-nw**] [**-nx**] [[**-path** *<dir>*] ...]
[[**-pdf**] *<filename>*] [**-pid**] [**-stack** *<bytes>*] [**-w**] [**-x** *<cmdfile>*]
[*<X-Toolkit-options>*] [**-e** *<executable>* [*<arguments ...>*]]

Parameter	Meaning
<executable>	Specify the name of the executable you wish to profile. The default is a.out.
-e *<executable>* [*<arguments ...>*]	Supply command line arguments to the program you are profiling. This must be the last option on the command line. This option is useful for batch execution; for example: **cxpa -x** *<cmdfile>* **-e a.out** *<args-list>*
-f	C Series only. Specify fixed scheduling for the program you are profiling to reserve all processors for its use. Fixed scheduling is useful when profiling programs that have sections that execute in parallel. Without fixed scheduling, the number of threads created depends on the system load.
-help	Display the CXpa usage message, which lists and describes command line options.
-nw	Invoke CXpa in line mode. Do not bring up the X window version.
-nx	Do not execute the CXpa commands in .cxpainit, the CXpa start-up command file.

-path *<dir>*	Append the specified directory to the end of CXpa's search path. CXpa uses its search path when it looks for a source file. Any number of directories can be specified by repeating the `-path` option several times. The list of directories is initialized to the current working directory and the directory where the executable or PDF is located.
-pdf *<filename>*	Invoke CXpa with the specified performance data file. CXpa uses the PDF to store profiling data and to generate the performance reports. The default PDF is *<executable>*`.pdf`.
-pid	Adds the process identification number (of the process you are profiling) to the name of the PDF that CXpa creates during a profiling session.
-stack *<bytes>*	Specify the size of the profiling stack in bytes. The default is 10240 bytes. If CXpa aborts because the profiling stack is too small, use this option to increase its size.
-w	Suppress warning messages issued by CXpa.
-x *<cmdfile>*	Execute the CXpa commands in the specified file. After CXpa has executed the commands in the file, the profiling session is terminated.
<X_Toolkit_options>	Specifies X Toolkit options. For more information about these options, refer to your X Window System's documentation.

Description

CXpa is an interactive performance analyzer for C and Fortran that enables you to profile optimized programs compiled with one of the profiling options (`-cxpa`, `-cxpar`, `-cxpab`). CXpa can run under X windows or with CRT terminals.

CXpa enables you to collect profiling data for four types of source code regions: routines, loops, parallel regions, and basic blocks. You can select any of these regions for profiling, provided they exist. The data collected for the selected regions can be viewed and analyzed using reports and 2D or 3D profile graphs.

CXpa stores the performance information collected for a specific run of your program in a performance data file, called a PDF. By default, CXpa appends .pdf to the name of the executable file to form the name of the PDF file (for example, a.out.pdf).

PDFs are platform-independent; the data stored in a PDF created on one platform can be viewed an analyzed on a different platform. For example, you can collect performance data for a large program in batch mode running on an SPP Series computer, then move the PDF file to a work station to interactively view and analyze the data.

To compile a program for profiling, use one of the following profiling options:

- `-cxpa`—Used for profiling routines, loops, and parallel regions (this option is recommended for most profiling).
- `-cxpar`—Used for profiling routines only.
- `-cxpab`—Used for profiling basic blocks only.
- `-cxpalib`—Used for linking your program with system libraries that are instrumented for CXpa (assuming they are installed on your system).
- `-cxpamon=`*<dir>*

  Specifies an alternate directory for CXpa data collection routines (cxpamon.o). As an example, you could use this option if more than one version of CXpa is installed on your system and you want to use the data collection routines for a different version of CXpa. The default value for *<dir>* is /usr/lib.

When compiling, you should only use one profiling instrumentation option (that is, `-cxpa`, `-cxpar`, or `-cxpab`) per source file.

Using the X windows version of CXpa

1. Compile your program with one of the CXpa profiling flags: `-cxpa`, `-cxpar`, or `-cxpab`.
2. Set your X Window DISPLAY environment variable. For example:

   % **`setenv DISPLAY`** *<display_name>*`:0.0`
3. Invoke CXpa with the name of your executable. The following command invokes CXpa in X window mode:

   % **`cxpa a.out &`**

   The Executable Manager window appears.
4. Choose the source code regions you want to profile by clicking the Profile Selection button. The Profile Selection dialog appears. Click the toggle buttons on the upper half of the dialog to select/deselect region types.
5. If you wish to change the default set of metrics that CXpa collects, click the toggle buttons in the lower half of the Profile Selection

dialog to select/deselect the metrics you want to collect. These metrics are only collected at the regions of your program selected for profiling in Step 4 above.

6. Run your program by pressing the Start button.
7. Create and view a performance report or graph by selecting one of the options from the lower-right button menu.

Using the line mode version of CXpa

Line mode allows you to use CXpa interactively without the graphical user interface. Line mode is designed primarily for use on windowless terminals (for example, VT-100s), but you can also use it on X terminals. Performance data is displayed in textual reports.

To use CXpa in line mode:

1. Compile your program with one of the CXpa profiling flags: `-cxpa`, `-cxpar`, or `-cxpab`.
2. Invoke CXpa with the name of your executable. The `-nw` option invokes CXpa in line mode.

   % **cxpa -nw a.out**

3. Choose the source code regions you want to profiling using the `select` command. For example, to select all routine regions, enter:

   (CXpa) **select routine all**

4. If you wish to change the default set of metrics that CXpa collects while your program is running, use the `collect` command. The metrics you select are only collected at the regions of your program that you selected for profiling with the `select` command.
5. Run your program:

   (CXpa) **run**

6. Create and view a performance report by using one of the `analyze` commands. For example:

   (CXpa) **analyze**

Using the CXPA environment variable to specify options

You can specify CXpa command line options in the CXpa environment variable (CXPA). This frees you from having to enter frequently used options repeatedly from the command line.

To use this environment variable in C shell (csh), use the following syntax:

`setenv CXPA '-option -option ...'`

To use this option in Bourne shell (sh), use the following syntax:

`CXPA='-option -option ...' ; export CXPA`

For example,

```
setenv CXPA ' -nw -pid -w'
```

forces CXpa to start in line mode (`-nw`). The `-pid` option tells CXpa to add the process ID number of the process you are profiling to the name of the PDF file it creates when you run your program. The `-w` option suppresses warning messages issued by CXpa.

Examples

The following examples show various ways to invoke CXpa.

```
% cxpa -nw
```

The above command invokes CXpa in line mode with the default executable (a.out) and default PDF (a.out.pdf) if they exist. If neither exist, CXpa starts, but only allows you to analyze existing PDFs. You can specify a PDF with the `set pdf` command.

```
% cxpa prog.out
```

The above command invokes CXpa in X window mode, and the Executable Manager window appears. The file name is treated as an executable file. This begins a profiling session using the specified executable file, and the default performance data file (PDF) named prog.out.pdf.

```
% cxpa -pdf my.pdf
```

The above command invokes CXpa in analysis mode, and the Analysis Control window appears. The `-pdf` option (which is optional) specifies the file name (my.pdf) as the performance data file (PDF). CXpa uses the PDF to store the performance data and to generate performance reports.

In X window mode, when you invoke CXpa with a pdf only, and no executable a.out exists in the current directory, the Analysis Control window appears. From this window, you can analyze many PDFs created with the same executable, but you cannot execute your program or gather more performance data without specifying an executable at the CXpa invocation line.

```
% cxpa -nw prog.out
```

The above command invokes CXpa with the executable `prog.out`. Using the `-nw` ("no windows") option invokes CXpa in line mode.

```
% cxpa -nx
```

The above command invokes CXpa but inhibits it from processing any initialization files.

```
% cxpa -x cmdfile my.out >& output_file < input_file
```

The above command invokes CXpa in batch mode (which is not an interactive mode). The `-x` option tells CXpa to execute the CXpa commands in the file cmdfile. Program output and messages are redirected to the file output_file, and input for the executable `my.out` is read from the file input_file.

```
% fc -03 test.c -cxpa -cxpalib
```

The above command line builds an executable file called a.out that is instrumented for loops, routines, parallel regions, and system libraries in a single step. The `-03` compiler option enables parallel optimizations.

```
% fc -02 -cxpa test.c -c
% fc file.f test.o -cxpalib
```

The first command in the above example creates an object file, test.o, with routines and loops instrumented for CXpa. The second command creates an executable, a.out, with instrumented libraries. The regions in test.o are instrumented for CXpa; the regions in file.o are not instrumented.

If you use the `-cxpalib` option, be aware that:

- The program you are profiling may execute much more slowly.
- Any errors in the instrumentation of the libraries may cause your program to core dump.
- The amount of profiling data will be significantly increased.

If you want to filter out the data from instrumented libraries during collection and/or analysis, use the `set visibility` command (in line mode) or turn off library visibility in the Visibility Selection dialog (in X window mode).

Related Commands

`analyze`	`collect`
`continue`	`deselect`
`quit`	`run`
`select`	`set events`
`set visibility`	

Related Windows

2D Profile window	3D Profile window
Analysis Control window	Analysis Report window
Executable Manager window	Profile Selection dialog

Related Topics

Advanced compiling	Analyzing PDFs only
Compiling	Introducing metrics
Introducing source code regions	Learning CXpa quickly

deselect
d

Deselect source code regions for profiling.

Syntax

deselect [*<region-type>*] [**all** | **in** *<routine-list>* | **at** *<line-number-list>*]

Parameter	Meaning

<region-type>	Specifies the type of region to deselect. Valid values are as follows:
	`routine`
	`loop`
	`pregion`
	`block`

all	Deselects the specified *<region-type>* in all routines in your program. If you do not specify a region type, all regions are deselected.
in *<routine-list>*	Specifies the names of one or more routines whose region types you want to deselect. Separate routine names with a space. If two routines have the same name, prefix them with a file name followed by a colon: *<file-name>* : *<routine-name>*.
at *<line-number-list>*	Deselects all region types at *<line-number-list>*. *<line-number-list>* specifies one or more line numbers of the region type that you want to deselect. Separate line numbers with a space. To deselect a region type in another source file, prefix the line number with a file name followed by a colon: *<file-name>* : *<line-number>*.

Description

The `deselect` command deselects source code regions for profiling in one or more routines or at one or more line numbers. The `deselect` command has no effect on regions that are already deselected.

When you invoke CXpa in line mode, all source code regions in your program are deselected for profiling until you select one or more of them with the `select` command. CXpa ignores deselected regions when you profile a program.

CXpa places collection instructions in your program's executable when you compile your program with one of the following CXpa options:

- `-cxpa`—Inserts routine, loop, and parallel region collection instructions into your program.
- `-cxpar`—Inserts routine collection instructions only.
- `-cxpab`—Inserts basic block collection instructions only.

For example, `deselect all` only deselects basic block collection instructions if you compiled your program with `-cxpab`.

Use the `list selectable` command to list source code regions that can be selected or deselected for profiling.

Examples

The following examples show various ways to use the `deselect` command.

```
(CXpa) deselect loop in init matrix_mult sub3
```

The above command deselects all loop regions for profiling in the routines `init`, `matrix_mult`, and `sub3`.

```
(CXpa) deselect at 14
```

The above command deselects all regions at line 14 in the current source file.

```
(CXpa) deselect pregion all
```

The above command deselects parallel regions for profiling in all routines in your program.

```
(CXpa) deselect loop in sub2
Routine sub2 has no instrumented loop entries.
```

In the above example, the loop regions in subroutine `sub2` could not be deselected because the source file containing `sub2` was not compiled with the `-cxpa` option.

Related Commands

`list selectable` `run` `select`

help
h

Display help information.

Syntax

help [*<string>*]

Parameter	Meaning
<string>	A character string used to search for help topics. All topic titles that contain the string are listed. If only one match is found, the help text for that topic is automatically displayed. If you enter the `help` command without specifying a search string, CXpa displays the text for the `help` command. The string can contain white space without being enclosed in quotes.

Description

The `help` command displays help information on specified commands, CXpa error, warning, and informational messages, and other topics. CXpa searches the help topic titles for the string you specify.

The output of the `help` command is sent to the pager specified in your PAGER environment variable. If you have not specified a pager with the PAGER environment variable, CXpa uses the `more` command to page output.

To display an ASCII text version of the release notice for the version of CXpa currently installed on your system, enter `help release`.

Examples

The following examples show how to use the `help` command.

```
(CXpa) help continue
```

The above command displays help information for the `continue` command.

```
(CXpa) help A19
A19
       Message      PDF <filename> is empty or corrupt.
       Type         ERROR
      Explanation Regenerate the PDF or try a different PDF.
```

The above command displays help information for CXpa error message number A19.

```
(CXpa) help pdf
Searching the topic titles found the following matches for
       'PDF':

   set pdf
   Analyzing PDFs only
   Setting the PDF
   Info PDF dialog
   New PDF dialog
   Open PDF dialog

(CXpa) help set pdf

                                                    set pdf
                                                    set p

          Specify the name of the performance data file (PDF).

Syntax      set pdf <filename>

(Skipping lines of output.)
```

In the above example, the command `help pdf` is ambiguous; CXpa displays all help topic titles that match the string `pdf`. The `help set pdf` command displays the help text for the `set pdf` command.

```
(CXpa) help index list
Index

      Enter 'help <topic_name> to view its  help page.

Windows                                  Using CXpa

About_CXpa_dialog                        Overview_of_CXpa

Analysis_Control_window                  CXpa_limitations
```

(...lines of output omitted)

```
(CXpa) help commands

Commands

      Enter 'help <command_name>' to view the help page
      for that command.

add path

    Add the specified list of directories to CXpa's search path
       for source files.

analyze

     Display performance analysis reports for profiled source
       code regions.
```

(...lines of output omitted)

As shown in the above example, you can view lists of online help topics by entering the following commands:

- `help using list`
- `help index list`
- `help commands`
- `help windows`
- `help reports list`

Related Commands

`info`

info

i

Display information about your CXpa session.

Syntax

```
info
```

Description

The `info` command lists the following information in three categories:

Executable information

- **Current executable**—Lists the executable that you are profiling.
- **Current Process State**—Lists the current state of the process you are profiling. The states include `running`, `paused`, `terminated`, `exited`, and `not started`.
- **Process ID**—Lists the executable's process ID (PID).
- **Created on**—Lists the date that the current executable was created.
- **Instrumentation Version**—Lists the version of the CXpa profiling routines linked into your program with a CXpa compiler option (`-cxpa`, `-cxpar`, `-cxpab`).

PDF information

- **Current PDF**—Lists the name of the current performance data file (PDF).
- **Created on**—Lists the date and time that the PDF was created.
- **PDF Format**—Lists the format version number of the PDF.
- **Created by CXpa Version**—Lists the version number of CXpa that created the PDF.
- **Process State**—Lists the process state recorded in the PDF.
- **Executable Profiled**—Lists the name of the executable you are profiling.
- **Executable Created on**—Lists the date that the executable was created.

- **Instrumentation Version**—Lists the version of the CXpa profiling routines linked into your executable when the PDF was created.
- **Visibility Setting**—Lists whether application or library routines are visible to CXpa during profile data collection or analysis.

File and directory information

- **Current List File**—Lists the name of the source file to use when using the `list` or `list selectable` commands.
- **Current Working Directory**—Lists the current directory.
- **Current Search Path(s)**—Lists the search path that CXpa uses to find source files.
- **Available Subcomplex(s)** —(SPP Series only) Lists subcomplexes available on your system. Use the `set subcomplex` command to specify a different subcomplex to run your program on.

Examples

The following example shows the output of the `info` command.

```
(CXpa) info

       Current Executable : /doc/srv/progs/a.out
    Current Process State : Not started
               Process ID : Not started
               Created on : Tue Nov 29 14:14:46 1994
  Instrumentation Version : 3.1

              Current PDF : /doc/srv/progs/a.out.pdf
               Created on : Wed Nov 30 18:00:00 1994
               PDF Format : 474
  Created by CXpa Version : 3.1
            Process State : exited
      Executable Profiled : /doc/srv/progs/a.out
    Executable Created on : Tue Nov 29 14:14:46 1994
  Instrumentation Version : 3.1
       Visibility Setting : Application, Library

         Current List File : generic.f
Current Working Directory : /doc/srv/progs
   Current Search Path(s) : /doc/srv/progs
  Available Subcomplex(s) : System, Chemistry
```

Related Commands

`version`

list
l

List lines of text from a source file.

Syntax

`list` [*<routine>* | [*<filename>*] [`:`] {*<first-line>* [*<last-line>*] | *<routine>*}]

Parameter	Meaning
<routine>	Specifies the name of a routine to display.
<filename>	Specifies the name of a source file to display.
<first-line>	Specifies a source code line number as the first line to display.
<last-line>	Specifies a source code line number as the last line to display.

Description

The `list` command lists lines of text from the source files that were compiled to form the current executable. Lines containing source code regions that can be selected or deselected for profiling are prefaced with one or more of the following letters: `r` for routines, `l` for loops, `p` for parallel regions, and `b` for basic blocks. Lowercase letters indicate regions that are currently deselected, while uppercase letters indicate regions that are currently selected for profiling.

When you execute the `list` command without parameters, CXpa lists the current source file. The current source file is either the last source file specified in a CXpa command or the source file that contains the program's main entry point.

When you use the `list` command, CXpa uses the directories in its search path to find the needed source file. If a source file has been moved after compiling, use the `add path` command to add the new directory to CXpa's search path.

The following list describes each permutation of the `list` command:

- `list`—List the current source file. Press the **SPACEBAR** to page forward.
- `list` *<routine>*—List the routine specified.
- `list` *<filename>*—List the specified source file. This source file must be one of the files compiled to form the current executable.

- `list` *<first-line>* [*<last-line>*]—List parts of the current source file by specifying the first and possibly the last line to display.
- `list` *<filename>*:*<first-line>* [*<last-line>*]—List parts of the specified file by specifying the first and possibly the last line to display.
- `list` *<filename>*:*<routine>*—List the routine specified. The file name allows you to choose between routines with the same names that are in different files.

Examples

The following examples show various ways to use the `list` command.

```
(CXpa) list
 R 1 PROGRAM GENERIC
   2 INTEGER VAR, VAR3, VAR4, COUNT
   3
   4 VAR = 126
   5 VAR4 = VAR * 16
     .
     .
     .
```

The above command lists the current source file. The uppercase `R` at line 1 indicates the routine region starting at line 1 is selected for profiling.

```
(CXpa) list sub2
 r    1 SUBROUTINE SUB2(MATRIX, VAR3)
      2 INTEGER MATRIX(5,5), VAR3, VAR5
      3
      4 PRINT *,'ENTERING SUB2'
 l p  5 DO I=1,5
 l    6 DO J=1,5
      7 MATRIX(I,J) = (I + J) - VAR3
      8 ENDDO
      9 ENDDO
     10 VAR5 = 38
     11 PRINT *,'LEAVING SUB2'
     12 PRINT *
     13 VAR5 = 12
     14 END
```

The above command lists the routine named `sub2` in the current source file. The lowercase letters `r`, `l`, and `p` indicate routine, loop, and parallel regions that are deselected for profiling.

```
(CXpa) list subs.f:8 30
   8
 L 9 DO 88 COUNT = 1,10,1
  10
  11 IF (COUNT .LT. 5) THEN
  12 VAR = VAR + 3
  13 CALL SUB1(VAR, VAR3)
  14 ELSE
  15 VAR = VAR + 32
  16 CALL SUB1(VAR, VAR3)
  17 ENDIF
  18
  19 VAR3 = VAR3 - 1
  20
  21 88 CONTINUE
  22
  23 CALL SUB4
  24
  25 IF (VAR .EQ. 2000) THEN
  26 CALL SUB5
  27 END IF
  28
  29 END
  30
```

The above command lists lines 8 through 30 in the file named subs.f. The uppercase `L` indicates that the loop region beginning at line 9 is selected for profiling.

```
(CXpa) list matrix_mult.c
```

The above command lists the source file named matrix_mult.c.

```
(CXpa) list 20 55
```

The above command lists lines 20 through 55 of the current source file.

```
(CXpa) list 85
```

The above command lists the current source file starting at line 85.

Related Commands

`add path`	`info`
`list selectable`	`path`

list selectable

ls

List source code regions available for profiling in a source file.

Syntax

`list selectable` [*<routine>* | [*<filename>*] [**`:`**] {*<first-line>* [*<last-line>*] | *<routine>*}]

Parameter	Meaning
<routine>	Specifies the name of a routine to display.
<filename>	Specifies the name of a file to display.
<first-line>	Specifies a source code line number as the first line to display.
<last-line>	Specifies a source code line number as the last line to display.

Description

The `list selectable` command lists only lines of source code that contain source code regions that can be selected for profiling in the source files that were compiled to form the current executable.

Source code regions that can be selected or deselected for profiling are prefaced with one or more of the following letters: `r` for routines, `l` for loops, `p` for parallel regions, and `b` for basic blocks. Lowercase letters indicate regions that are currently deselected, while uppercase letters indicate regions that are currently selected for profiling (refer to the "Selecting regions in line mode" online help topic or section in this book for more information).

When you use the `list selectable` command without parameters, CXpa lists the lines of source code in the current source file that contain selectable source code regions. The current source file is either the last source file specified or the source file that corresponds to the first object file given to the linker to form the current executable.

When you execute the `list selectable` command, CXpa uses the directories in its search path to find the needed source file. If a source file has been moved after compiling, use the `add path` command to add the needed directory to CXpa's search path.

The following list describes each permutation of the `list selectable` command:

- `list selectable`—List lines selectable regions in the current source file.
- `list selectable` *<routine>*—List the lines containing selectable regions in the routine specified.
- `list selectable` *<filename>*—List lines of source code containing selectable regions in the file specified.
- `list selectable` *<first-line>* [*<last-line>*]—List the lines of source code containing selectable regions in the range specified.
- `list selectable` *<filename>*`:`*<first-line>* [*<last-line>*]—List the lines containing selectable regions in the specified source file in the range specified.
- `list selectable` *<filename>*`:`*<routine>*—List selectable regions in the routine specified. Specifying the file name allows you to distinguish between two routines with the same name that are in different files.

If you enter the `list selectable` command and get no output, it is because there were no selectable source code regions in the range you specified.

Examples

The following examples show various ways to use the `list selectable` command.

```
(CXpa) list selectable
  r    1 PROGRAM GENERIC
 L     9 DO 88 COUNT = 1,10,1
```

The above command lists the regions that can be selected for profiling in the current source file. The annotations show that the routine region at line 1 is currently selected for profiling and the loop region at line 9 is deselected.

```
(CXpa) list selectable sub2
  r   1 SUBROUTINE SUB2(MATRIX, VAR3)
 L p  6 DO I=1,5
 L    7 DO J=1,5
 L    9 DO K=1,30
```

The above command lists the regions that can be selected for profiling in the routine named `sub2`. The annotations show that the loop regions at lines 6, 7, and 9 in `sub2` are currently selected for profiling and the routine region at line 1 is deselected.

```
(CXpa) list selectable prog.f:68 99
  r  68 SUBROUTINE SUB4
 L   76 DO I = 1,100,2
 L   79 DO WHILE (K .LT. 10)
 L p 82 DO J = 1,20,1
  r  99 SUBROUTINE SUB5
```

The above command lists the regions available for profiling in lines 68 through 99 in the file named prog.f. The annotations indicate that the loop regions are currently selected for profiling, and the routine regions are currently deselected.

```
(CXpa) list selectable subs.f
```

The above command lists the regions that can be selected for profiling in the file named subs.f. In this case, no regions are available for profiling.

```
(CXpa) list selectable 20 55
 R 33 SUBROUTINE SUB1(VAR, VAR3)
 R 51 SUBROUTINE SUB3(MATRIX)
```

The above command lists the regions that can be profiled on lines 20 through 55 of the current source file.

```
(CXpa) list selectable 51
 R 51 SUBROUTINE SUB3(MATRIX)
 L 56 DO I=1,5
```

The above command lists the regions that can be profiled in the current source file starting at line 51. Both of these regions are currently selected for profiling, as indicated by the uppercase `R` and `L` annotations.

Related Commands

`add path` `info` `list` `path`

path
p

Set CXpa's search path for source files.

Syntax

path *<directory-list>*

Parameter	Meaning
<directory-list>	Specifies one or more directories, separated by a space, as CXpa's search path.

Description

The `path` command replaces CXpa's current search path with the one specified in *<directory-list>*.

By setting CXpa's search path, you tell CXpa where to look for source files. When you compile your source code with CXpa options, CXpa embeds the location of the source files in the executable so that CXpa can find these files. If they are moved after compiling, then CXpa cannot find them.

You can append one or more directories to CXpa's current search path with the `add path` command. You can display CXpa's current search path with the `info` command.

Examples

The following example shows how to use the `path` command.

```
(CXpa) path /usr/data /usr/data/input /usr/data/output
(CXpa) info

   (Skipping lines of command output not relevant to this example.)

Current Search Path(s) : /usr/data
                         /usr/data/input
                         /usr/data/output
```

The `path` command in the above example sets CXpa's search path to the specified directories. The `info` command shows CXpa's search path.

Related Commands

`add path`
`info`
`list`
`list selectable`

quit

q

Exit CXpa.

Syntax

quit

Description

The `quit` command exits you from CXpa. If you issue the `quit` command during an active profiling session, CXpa notifies you and asks if you still wish to quit. If so, the performance data file (PDF) is closed, and the program you were profiling is terminated.

Examples

This section shows examples of the `quit` command.

```
(CXpa) quit
%
```

The `quit` command exits CXpa and returns you to your shell prompt.

```
(CXpa) quit
Program is paused, quit anyway? ([y]/n)?
```

In the above example, CXpa informs you that you have paused a process and asks if you still wish to quit.

Related Commands

`continue` `stop`

rerun

re

Run and profile your program again using the same argument list.

Syntax

rerun [*<i/o_redirection>*]

Parameter	Meaning
<i/o_redirection>	Redirects the program's standard input, output, or error from or to the specified file when you include one of the redirection operators (<, >, >>, >&, >>&).

Description

The `rerun` command executes the current executable with the arguments that you specified in the last `run` command.

NOTE: If you use a preexisting PDF, the performance data in the PDF will be overwritten with new data when you execute the `run` or `rerun` command. Use the `set pdf` command to specify a new name for the PDF to avoid overwriting an existing PDF.

NOTE: The `rerun` command does not retain file redirection specified with the `run` command.

While your program is running, you can type **CTRL-c** to pause profiling. Once paused, you can use the `analyze`, `stop`, or `continue` commands.

Examples

The following examples show how the `rerun` command works.

```
(CXpa) run 8 5
The arguments are 8 and 5.
(CXpa) rerun
The arguments are 8 and 5.
```

The above example shows that the `rerun` command uses the arguments specified in the last `run` command.

```
(CXpa) rerun < /usr/data/input
```

The above command runs your program using the arguments from the last `run` command and redirects standard input from the file /usr/data/input.

```
(CXpa) rerun > /usr/data/output
```

The above command runs your program, using the arguments from the last `run` command and redirects standard output to the file /usr/data/output.

```
(CXpa) rerun >& /usr/data/output
```

The `>&` in the above command tells CXpa to redirect standard output and standard error to the file /usr/data/output.

Related Commands

`continue` `run` `stop`

run

ru

Run and profile your program with the specified arguments.

Syntax

run [*<argument>* . . .] [*<i/o_redirection>*]

Parameter	Meaning
<argument>	Specifies any number of command line arguments to the program you are profiling.
<i/o_redirection>	Redirects the program's standard input, output, or error from or to the specified file when you include one of the redirection operators (<, >, >>, >&, >>&).

Description

The `run` command starts profile data collection and executes the current executable. You specify this executable when you invoke CXpa.

As your program runs, CXpa collects metrics at the regions you selected for profiling with the `select` command. CXpa writes the data it collects to the performance data file (PDF). The default PDF name is *<executable>*.pdf. Use the `set pdf` command to specify another PDF name.

NOTE: If you use a preexisting PDF, the performance data in the PDF will be overwritten with new data when you execute the `run` or `rerun` command. Use the `set pdf` command to specify a new name for the PDF to avoid overwriting an existing PDF.

While your program is running, you can type **CTRL-c** to pause profiling. Once paused, you can use the `analyze`, `stop`, or `continue` commands.

Examples

The examples in this section show various ways to use the `run` command.

```
1.(CXpa) select loop pregion all
2.(CXpa) collect wall_clock cpu
3.(CXpa) run
4.(CXpa) CTRL-c
   (Program execution is paused.)
5.(CXpa) stop
Process 24144 terminated with signal: SIGKILL.
```

The above scenario shows how the `run` command is typically used. The line numbers are for reference only.

1. The `select loop pregion all` command selects all loops and parallel regions in your program for profiling.
2. The `collect wall_clock cpu` command tells CXpa to collect wall clock time and CPU time metrics.
3. The `run` command begins program execution and data collection.
4. Program execution is paused by pressing **CTRL-c**.
5. The `stop` command terminates program execution.

```
(CXpa) run
(Program runs to completion, and any output is displayed.)
```

The command in the above example executes the current executable. No arguments are passed to the program. The program's output follows the `run` command.

```
(CXpa) run 18 364
```

The above command executes the program you are profiling and supplies 18 and 364 as arguments to the program.

```
(CXpa) run < /usr/data/input
```

The above command executes the program you are profiling and redirects standard input from the file /usr/data/input.

```
(CXpa) run > /usr/data/output
```

The above command executes the program you are profiling and redirects standard output to the file /usr/data/output.

```
(CXpa) run >& /usr/data/output
```

The `>&` in the above command tells CXpa to redirect standard output and standard error to the file /usr/data/output.

```
(CXpa) run > /usr/data/output >& /usr/data/errors
```

The above example redirects standard output and standard error to different files.

Related Commands

`continue` `rerun` `stop`

select

sel

Select source code regions in your program for profiling.

Syntax

select [*<region-type>*] [**all** | **in** *<routine-list>* | **at** *<line-number-list>*]

Parameter	Meaning

<region-type>	Specifies the type of region to select. Valid values are as follows:
	`routine`
	`loop`
	`pregion`
	`block`

all	Selects the specified *<region-type>* in all routines in your program. If you do not specify a region type, all available regions are selected.
in *<routine-list>*	Specifies the names of one or more routines whose regions you want to select. Separate routine names with a space. If two routines have the same name, prefix them with a file name followed by a colon: *<file-name>*:*<routine-name>*.
at *<line-number-list>*	Selects regions at *<line-number-list>*. *<line-number-list>* specifies one or more line numbers that contain regions you want to select. Separate line numbers with a space. To select a region that is not in the current source file, prefix the line number with a file name followed by a colon: *<file-name>*:*<line-number>*.

Description

The `select` command to select source code regions for profiling in all routines, in specific routines, or at specific lines in source files.

You can select all available source code regions in your program or a subset. You do not have to recompile your program to select or deselect regions for profiling. Typically, you will want to select all routine regions the first time you profile your program with the `select routine all` command. This provides a complete picture of your program's performance.

After you have identified the routines whose performance you want to improve, you may want to select only those specific routines for profiling. With fewer source code regions selected, less time is spent in the timing routines CXpa uses to collect performance data. Selecting code regions for profiling does increase wall-clock time. Profiling time increases with the number of regions selected for profiling. In general selecting loop regions for profiling increases execution time more than selecting routine regions.

When you invoke CXpa in line mode, all regions in your program are initially deselected for profiling, so you must select one or more of them with the `select` command. When you run your program CXpa collects metrics only at the regions selected for profiling.

Depending on the option(s) with which you compiled your source code, four types of source code regions can be selected for profiling:

- **Routine**—Routine regions are only available for profiling if your source code is compiled with the `-cxpa` or `-cxpar` option.
- **Loop**—Loop regions are only available for profiling if your source code contains loops and is compiled with the `-cxpa` option at optimization level `-O1` or higher.
- **Parallel region**—Parallel regions (parallel loops) are only available for profiling if your source code is compiled with the `-cxpa` option at optimization level `-O3` (at optimization levels below `-O3`, the compiler does not parallelize loops).
- **Basic block**—Basic block regions are only available for profiling if your source code is compiled with the `-cxpab` option.

To list source code regions that can be selected for profiling, use the `list selectable` command.

Examples

The following examples show various ways to use the `select` command.

```
(CXpa) select routine all
```

The above command selects all routine regions in your program for profiling.

```
(CXpa) select pregion in sub2
ERROR A34: Routine sub2 has no parallel regions instrumentation.
```

In the above example, parallel regions in routine `sub2` cannot be selected because the source file containing the routine `sub2` was not compiled with the `-cxpa` and `-O3` options.

```
(CXpa) select pregion at 9
```

The above command selects all parallel regions at line 9 in the current source file.

```
(CXpa) select loop in sub2 init display
```

The above command selects all loop regions in routines `sub2`, `init`, and `display` for profiling.

```
(CXpa) select at 82
(CXpa) list selectable sub4
       r      68   SUBROUTINE SUB4
     l        76   DO I = 1,100,2
     l        79      DO WHILE (K .LT. 10)
     L P      82         DO J = 1,20,1
```

In the above example, the `select at 82` command selects all regions at line 82 of the current source file for profiling.

The `list selectable sub4` command lists the regions that can be selected for profiling in the routine named `sub4`. The letters to the left of the line numbers in the `list selectable` output represent available regions for collection. The uppercase `L` and `P` indicate that line 82 has selected loop and parallel regions.

Related Commands

`deselect`
`list selectable`
`run`
`set pdf`

set events

set e

Specify the type of event to collect at source code regions selected for profiling.

Syntax

set events <*event*> [**read_only**] [**write_only**]

Parameter	Meaning
<*event*>	Specifies the type of event to collect. The type of events that can be collected differ according to machine architecture, and are listed below:

SPP Series events

local_misses—Collects the number of times that data not found in the processor cache was found in local memory (memory allocated to that processor's hypernode). The CTI rings are not used.

remote_misses—Collects the number of times that data not found in the processor cache was found in remote memory (memory allocated to a different hypernode). The CTI rings are used. Multinode configurations only.

local_and_remote_misses—Collects the number of times that data not found in the processor cache was found in local and remote memory, combined.

read_only (SPP Series only)

Specifies that CXpa should collect these events during read operations. A read miss is a data cache miss caused by a load. By default, CXpa collects events during reads and writes (`read_only write_only`).

write_only (SPP Series only)

Specifies that CXpa should collect these events during write operations. A write miss is a data cache miss caused by a store or a data cache miss caused by a load and clear. By default, CXpa collects events during both reads and writes (`read_only write_only`).

C4 Series events

data_accesses—Collects the number of times that a value was accessed in the CPU data cache for the profiled regions of your program.

data_misses—Collects the number of times that a value was not found in the CPU data cache for the profiled regions of your program. This is the default.

instruction_misses—Collects the number of times that an instruction was not found in the CPU instruction cache for the profiled regions of your program.

page_table_misses—Collects the number of page table entry misses that occurred in the profiled regions of your program.

vector_load_stores—Collects the number of loads and stores to and from memory into and out of vector registers for the profiled regions of your program.

Description

The `set events` command specifies the collection of one type of event at the source code regions of your program you have selected for profiling. Each use of this command replaces the values previously set. For more information about events metrics, refer to the "Introducing metrics" and "Selecting metrics in line mode" online help topics or sections in this book.

NOTE: Unless you have selected source code regions to profile with the `select` command and specified event collection with the `events` parameter of the `collect` command, the `set events` command will not cause any events to be collected.

Examples

The following examples show various ways to use the `set events` command.

```
1. (CXpa) select loop all
2. (CXpa) collect cpu wall_clock events
3. (CXpa) set events local_misses
```

The above commands show how to use the `set events` command in combination with the `select` and `collect` commands to select regions for profiling and metrics to collect. The line numbers are for reference only.

1. The `select loop all` command selects all loop routines in your program for profiling.
2. The `collect` command specifies data collection for CPU time, wall clock time, and events. You must define the types of events to collect with the `set events` command.
3. The `local_misses` parameter of the `set events` command is only valid for SPP Series machines. The `set events local_misses` command specifies that you want to collect the number of times that data not found in the processor cache was found in memory allocated on the processor's hypernode.

```
(CXpa) set events vector_load_stores
```

The above command tells CXpa to collect the number of vector memory accesses that occur for the profiled regions of your program. Vector load/store events can only be collected on C4 Series machines.

```
(CXpa) set events remote_misses write_only
```

Remote misses can only be collected on multinode SPP Series machines. The above command tells CXpa to collect the number of times that data not found in the processor cache was found in remote memory (memory allocated to a different hypernode) for the profiled regions of your program. The `write_only` parameter specifies that remote misses will only be collected during write operations.

Related Commands

`collect`
`deselect`
`list selectable`
`run`
`select`

Related Topics

Introducing metrics

Selecting metrics in line mode

set pdf
set p

Specify the name of the performance data file (PDF).

Syntax

`set pdf` *<filename>*

Parameter	Meaning
<filename>	Specifies the name of a PDF. The file name you specify should end in .pdf.

Description

The `set pdf` command sets the name of the performance data file (PDF) to be written to and/or read from during a CXpa session.

The PDF is a binary file that contains the performance data for a single run of your program. The data in a PDF is used to calculate the performance reports that appear when you use one of the `analyze` commands. If you do not set the PDF name, CXpa uses the default PDF name of *<executable>*.pdf.

Using the `set pdf` command, you can change the name of the PDF. You may want to do this for two reasons:

- **To prevent CXpa from overwriting an existing PDF**—If you have invoked CXpa with the name of an executable file, when you use CXpa's `run` or `rerun` command, CXpa overwrites all of the data in the PDF. If you do not want this to occur, you must change the name of the PDF between runs of your program with the `set pdf` command.
- **To analyze a different PDF**—If you have invoked CXpa without an executable or with the name of a PDF only, you can use the `set pdf` command to select a PDF created in a previous CXpa session. You can then display performance reports for that PDF with the `analyze` command.

NOTE: If you invoke CXpa with an executable, you can only analyze a PDF created during the current CXpa session. To analyze a PDF created with a different executable or multiple PDF files, invoke CXpa without specifying an executable or with the name of a PDF file.

Examples

The following examples show how to use the `set pdf` command.

```
(CXpa) set pdf /usr/data/my.pdf
```

The previous command sets the name of the PDF to my.pdf. If a program is now run, performance data is collected in the file /usr/data/my.pdf. If a report is generated, CXpa analyzes the data in /usr/data/my.pdf.

```
(CXpa) set pdf ../profiles/prog1.pdf
```

The previous command sets the name of the PDF to prog1.pdf in the ../profiles directory.

Related Commands

`analyze`
`deselect`
`run`
`select`

Related Topics

Analyzing PDFs only
Setting the PDF

SPP only

set subcomplex

set s

Specify the subcomplex you want to run your program on.

Syntax

`set subcomplex` *<subcomplex_name>*

Parameter	Meaning
<subcomplex_name>	Specifies the name of the subcomplex on which to run your program. The default is the subcomplex from which you invoked CXpa.

Description

Use the `set subcomplex` command to specify the name of the subcomplex you want to run your program on. A *subcomplex* is a collection of processors and memory from one or more nodes of an SPP Series system (which define the boundary within which all threads belonging to a process execute).

The default value is the subcomplex from which you invoked CXpa. You only need to use this command if you want to run your program on a different subcomplex. You must have the appropriate permissions to run the process on the specified subcomplex.

For more information on subcomplexes on SPP Series systems, refer to the scm(1) and scm(4) man pages or the Convex *SPP-UX System Administration Guide* (DSW-853) or contact the system administrator at your site.

To list available subcomplexes on your system, use the CXpa `info` command.

CXpa queries the system for available subcomplexes at start-up, so if a subcomplex is added during a CXpa session, it will not be visible to CXpa during that session.

Examples

The following example shows how to use the `set subcomplex` command.

```
(CXpa) info

      Current Executable : /home/smith/a.out
   Current Process State : Finished
```

(Skipping lines of output not relevant to this example)

```
 Available Subcomplex(s) : Chemistry, System
(CXpa) set subcomplex Chemistry
```

The last line in the output of the `info` command in the above example lists available subcomplexes on your SPP system. The `set subcomplex Chemistry` command specifies that the process will now run on the Chemistry subcomplex (assuming you have the appropriate permissions for that subcomplex).

Related Commands

`info`

set visibility

set v

Filter data displayed in a performance data report.

Syntax

```
set visibility {library | application |
   library application} {process | threads}
```

Parameter	Meaning
`library`	Makes the CXpa-instrumented library routines linked with your program visible to CXpa during profiling or analysis.
`application`	Makes the CXpa-instrumented routines in your program visible during profiling or analysis.
`process`	Enables you to display process-level performance data in reports. This is the default.
`threads`	Enables you to display performance data for individual threads in reports.

Description

The `set visibility` command allows you to set visibility for:

- **Library and application routines**—When a routine is *visible* to CXpa, it means that it can be selected for profiling and that data collected during profiling can be viewed and analyzed in performance graphs and reports.
- **Processes or threads**—By default, performance data in reports (except for parallel region reports) is displayed for processes; by setting visibility for threads you can display performance data for individual threads.

Each time you use the `set visibility` command, it overwrites the previous setting for that type (library/application visibility, process/thread visibility, or both). Refer to the "Examples" section of this reference page or online help topic.

You can make library routines, your application's routines or both visible to a CXpa session. By default, library and application visibility is set for routines. Use the `info` command to view the current setting.

NOTE: Profiling data for system libraries instrumented for CXpa can only be collected and viewed

- **If you compile your source files with the `-cxpalib` option and with CONVEX C V6.0 or greater or CONVEX Fortran V9.0 or greater, and**
- **If the CXpa instrumented libraries are installed on your system.**

Without these conditions, you can set visibility for library regions, but CXpa will only collect or analyze profiling data for application source regions.

You can profile and view performance data at the process level or at the thread level. The default process/thread visibility setting is process.

NOTE: Profiling data for threads can only be collected and viewed if compiler option `-O3` is used, your computer has more than one processor, your program has parallel regions, and it executes in parallel.

Examples

The following examples show how to use the `set visibility` command.

```
(CXpa) set visibility library application thread
```

The above command enables you to display performance data for individual threads and makes both application routines and system libraries instrumented with CXpa visible during profiling and analysis.

```
1.(CXpa) set visibility application
2.(CXpa) set visibility thread
```

The line numbers in the above example are for reference only.

1. The `set visibility application` command sets visibility for application routines only and overrides the previous or default application/library visibility setting. The thread/process visibility setting is not affected.
2. The `set visibility thread` command sets visibility for threads and overrides the previous or default process/thread visibility setting. It does not override the application/library visibility setting specified with the `set visibility application` command.

Related Commands

`analyze`
`deselect`
`info`
`select`
`set events`
`set pdf`

source

so

Execute a CXpa command file.

Syntax

source *<filename>*

Parameter	Meaning
<filename>	Specifies the name of a CXpa command file.

Description

The `source` command executes the CXpa commands in a specified CXpa command file. Sourcing a command file is useful when you find that you are repeating a series of commands during a profiling session.

A CXpa command file is a text file containing a list of CXpa commands. Each command must be on a separate line. Comment lines are denoted with the pound sign (#).

In line mode, CXpa exits and returns you to the shell prompt when it encounters a `quit` command; otherwise, CXpa returns you to the CXpa prompt when reaches the end of the command file.

In batch mode, CXpa exits when it encounters either a `quit` command or the end of the file.

If CXpa encounters an error in a command file, CXpa stops executing the command file and returns the CXpa prompt.

Examples

The following examples show how to use the `source` command and CXpa command files.

```
(CXpa) source my_cmd_file
```

The above command executes the CXpa commands in the file named my_cmd_file.

```
# This is a comment line.
select loop in sub4
run
analyze loop > sub4.report
quit
```

The above example shows the format of a CXpa command file.

Related Commands

`analyze`	`quit`
`run`	`select`

Related Topics

Using batch mode

stop

st

Stop profiling a paused program.

Syntax

```
stop
```

Description

The `stop` command stops data collection, saves the collected data, and terminates the process being profiled. Before you can use the `stop` command, you must pause the program first by pressing **CTRL-c**. Then enter the `stop` command.

When you stop or pause profiling, you can use the `analyze` command to view profile data that was collected up until your program was stopped.

When you display performance reports for a stopped program, the information in the reports is incomplete and only reflects the partial execution of your program.

Examples

The following example shows a likely scenario in which you would use the `stop` command. The line numbers are for reference only.

```
1.(CXpa) run
   (Program output is displayed.)
2.(CXpa) CTRL-c
   (Pressing CTRL-c pauses profiling.)
3.(CXpa) stop
4.(CXpa) analyze
   (Incomplete performance reports are displayed.)
```

The following lines explain the previous example by number:

1. The `run` command runs the program and initiates profile data collection.
2. **CTRL-c** pauses profiling.
3. The `stop` command stops the profiling of your program.
4. The `analyze` command displays the collected profile data.

Related Commands

`analyze` `continue` `run`

version

V

Display version and release information for CXpa.

Syntax

version

Description

The `version` command lists:

- CXpa's version number
- The release date of this version of CXpa
- The product number
- Copyright information

Examples

The output of the `version` command is shown in the following example.

```
(CXpa) version
Convex Performance Analyzer

Version : 3.1
Product # : 710-018415-006
Release Date : 12/21/94

Copyright (c) 1989-1994 Convex Computer Corporation
All rights reserved.

Report problems by running the "contact" program
which will send electronic mail to the
Convex Technical Assistance Center.
```

Related Commands

`info`

Messages 7

This chapter contains reference pages that explain CXpa messages. The messages are listed in order by identification number (ID). Each explanation contains the following sections:

- **Message** — The exact text of the message. Variable parameters are enclosed in angle brackets (<>) and are shown in italic type. For example, *<collection type>* is a variable that represents a type of source code region.
- **Type** — The message type, which can be one of the following:
  - `INFO` — A condition that is normal or expected processing under the given circumstances.
  - `WARNING` — A condition that might not be normal or expected, but is not severe enough to cause an error. CXpa continues to process your command after issuing the warning.
  - `ERROR`—A condition that prevents completion of the current CXpa command.
  - `FATAL`—A condition that prevents further execution of the process being profiled. Both the process and the Cxpa session are terminated.
- **Explanation**—A more detailed explanation of the message, including possible actions to correct the situation.

To display online help for CXpa messages:

- In X window mode, enter the message number in the Search field in the CXpa Help window.
- In line mode, enter the command `help` *<message_number>* at the CXpa prompt.

ID		Description
A1	Message	<*filename*> has been stripped of all symbolic information, unusable.
	Type	ERROR
	Explanation	Quit CXpa, recompile executable source code with one of the CXpa options, and then reinvoke CXpa with the new executable file.
A2	Message	Missing or unmatched <*token*>.
	Type	ERROR
	Explanation	Correct the quotation marks and re-execute the command.
A3	Message	Command syntax error - <*expecting or missing*> <*token*>.
	Type	ERROR
	Explanation	For more information about this command, refer to the online help system or the reference manual.
A4	Message	Command '<*string*>' is ambiguous. Could refer to: <*keyword list*>
	Type	ERROR
	Explanation	Complete your command further so that it can be uniquely recognized.
A5	Message	Command line too complex.
	Type	ERROR
	Explanation	Try to simplify and re-execute the command.
A6	Message	Cannot open input file <*filename*>.
	Type	ERROR
	Explanation	Check to make sure the file exists and the permissions are set so that you can access it.
A7	Message	Cannot open output file <*filename*>.
	Type	ERROR
	Explanation	Check directory and file permissions to see if you are allowed write access.
A8	Message	Cannot have more than one redirection to/from <*filename*>.
	Type	ERROR
	Explanation	Re-execute the command (in line/batch mode) or GUI action (in X window mode) and redirect it to/from another file.
A9	Message	Cannot redirect to <*filename*>.
	Type	ERROR
	Explanation	Re-execute the command (in line/batch mode) or GUI action (in X window mode) and redirect it to/from another file.

ID		Description
A10	Message	Cannot start a new process while another process is still running.
	Type	ERROR
	Explanation	Wait until the current process completes or use the "stop" command (in line/batch mode) or the Abort button (in X window mode) to stop it.
A11	Message	Cannot access PDF *<filename>*.
	Type	ERROR
	Explanation	Check directory and file permissions to see if you have read/write access.
A12	Message	Cannot open file *<filename>*. *<errno description>*
	Type	ERROR
	Explanation	Check directory and file permissions to see if you have write access.
A13	Message	Cannot read from PDF. *<errno description>*.
	Type	ERROR
	Explanation	Check directory and file permissions to see if you have read access.
A14	Message	Cannot read from PDF.
	Type	ERROR
	Explanation	PDF may be corrupt.
A15	Message	Cannot write to PDF. *<errno description>*.
	Type	ERROR
	Explanation	Check directory and file permissions to see if you have write access.
A16	Message	Cannot write to PDF.
	Type	ERROR
	Explanation	PDF may be corrupt.
A17	Message	Cannot find file *<filename>*.
	Type	ERROR
	Explanation	Check that the file exists and that you specified its name correctly. If it is a source file, try using the "add path" command (in line/batch mode) or the Source Search Path dialog (in X window mode) to help CXpa locate the file.
A18	Message	Executable required before using this command.
	Type	ERROR
	Explanation	Quit CXpa and invoke it with the name of an executable file. Then re-execute this command (in line/batch mode) or GUI action (in X window mode).

ID		Description
A19	Message	PDF <*filename*> is empty or corrupt.
	Type	ERROR
	Explanation	Regenerate the PDF again or try a different PDF.
A20	Message	Cannot create PDF <*filename*>.
	Type	ERROR
	Explanation	Check directory and file permissions to see if you have write access.
A21	Message	PDF <*filename*> cannot be analyzed with this executable.
	Type	ERROR
	Explanation	PDF and this executable are incompatible. Invoke CXpa in analysis mode with the PDF file only or regenerate PDF with this executable.
A22	Message	PDF incompatible with this version of CXpa; regenerate PDF.
	Type	ERROR
	Explanation	PDF can no longer be analyzed; regenerate PDF.
A23	Message	Cannot read name of executable file from PDF.
	Type	ERROR
	Explanation	PDF may be corrupt.
A24	Message	No selectable regions for profiling at line <*count*> in file <*filename*>.
	Type	ERROR
	Explanation	Re-execute the command with a different line number. Use the "list selectable" command (in line/batch mode) or the Profile Selection dialog (in X window mode) to view the available source regions.
A25	Message	No selectable regions for profiling at line <*count*>.
	Type	ERROR
	Explanation	Re-execute the command with a different line number. Use the "list selectable" command (in line/batch mode) or the Profile Selection dialog (in X window mode) to view the available source regions.
A26	Message	No selectable regions for profiling at line <*count*> in file <*filename*> of <*collection type*> regions.
	Type	ERROR
	Explanation	Re-execute the command with a different line number. Use the "list selectable" command (in line/batch mode) or the Profile Selection dialog (in X window mode) to view the available source regions.

ID		Description
A27	Message	Source file must be specified with this command.
	Type	ERROR
	Explanation	Re-execute this command with an existing source file (in line/batch mode) or use the Windows menu option, Source Code, to select a source file (in X window mode).
A28	Message	File *<filename>* is not an executable file.
	Type	ERROR
	Explanation	Check the directory and file permissions to see if you are allowed to execute it.
A29	Message	Error in command file *<filename>*.
	Type	ERROR
	Explanation	Edit the command file to correct any errors, then re-execute it.
A30	Message	PDF contains incomplete data.
	Type	ERROR
	Explanation	Regenerate the PDF or select a different PDF.
A31	Message	Obsolete.
	Type	ERROR
	Explanation	None.
A32	Message	PDF, *<filename>*, has the same name as the executable file.
	Type	ERROR
	Explanation	PDF and executable files must be unique. Choose a different filename for the PDF using the "set pdf" command (in line/batch mode) or the File menu option, New PDF (in X windows mode).
A33	Message	Routine *<routine identifier>* is not selectable for profiling.
	Type	ERROR
	Explanation	Recompile the source file containing the routine with one of the CXpa compiler options.
A34	Message	Routine *<routine identifier>* is not selectable for profiling of *<collection type>* regions.
	Type	ERROR
	Explanation	Recompile the source file containing the routine with one of the CXpa compiler options.
A35	Message	Cannot list alternate entries to routine *<routine identifier>*.
	Type	ERROR
	Explanation	Alternate entries cannot be listed, selected, or analyzed directly.

ID		Description
A36	Message	Cannot select alternate entries to routine *<routine identifier>*.
	Type	ERROR
	Explanation	Alternate entries cannot be listed, selected, or analyzed directly.
A37	Message	Routine *<routine identifier>* is not unique.
	Type	ERROR
	Explanation	Re-execute the command (in line/batch mode) or GUI action (in X window mode) qualifying the routine with a filename. Use the format *<filename>*:*<routine>*.
A38	Message	Cannot read source file *<filename>*.
	Type	ERROR
	Explanation	Source file is empty.
A39	Message	PDF required before using this command.
	Type	ERROR
	Explanation	Choose a PDF with the "set pdf" command and re-execute the command (in line/batch mode) or choose a PDF with the File menu option, New PDF, and re-execute the GUI action (in X window mode).
A40	Message	Obsolete.
	Type	ERROR
	Explanation	None.
A41	Message	Obsolete.
	Type	ERROR
	Explanation	None.
A42	Message	Obsolete.
	Type	ERROR
	Explanation	None.
A43	Message	File *<filename>* is not part of source code for current executable.
	Type	ERROR
	Explanation	Re-execute the command (in line/batch mode) with a different source file (in line/batch mode) or use the Windows menu option, Source Code, to select a different source file and then re-execute the GUI action (in X window mode).
A44	Message	Obsolete.
	Type	ERROR
	Explanation	None.

ID		Description
A45	Message	Directory <*filename*> does not exist.
	Type	ERROR
	Explanation	Re-execute the command (in line/batch mode) or GUI action (in X window mode) with a different directory name.
A46	Message	Obsolete.
	Type	ERROR
	Explanation	None.
A47	Message	Terminating with signal <*signal identifier*>. <*errno description*>
	Type	FATAL
	Explanation	CXpa has received a fatal signal and is terminating. There is no recovery from this condition.
A48	Message	No regions selected for profiling. No region-level analysis will be possible.
	Type	INFO
	Explanation	To obtain region-level metrics use the "select" and "collect" commands (in line/batch mode) or the Region Selection dialog (in X window mode).
A49	Message	Obsolete.
	Type	ERROR
	Explanation	None.
A50	Message	Obsolete
	Type	ERROR
	Explanation	None.
A51	Message	Obsolete
	Type	FATAL
	Explanation	None.
A52	Message	Cannot find routine <*routine identifier*>.
	Type	ERROR
	Explanation	Check to see if the source file containing this routine is within the search path(s) or re-execute the command (in line/batch mode) or GUI action (in X window mode) with a different routine name.
A53	Message	Routine <*routine identifier*> was not compiled for or rejected from profiling.
	Type	ERROR
	Explanation	Recompile the source file containing the routine with one of the CXpa compiler options. Refer to the CXpa limitations section online help topic or the CXpa Reference.

ID		Description
A54	Message	Cannot analyze alternate entries to routine *<routine identifier>*.
	Type	ERROR
	Explanation	Alternate entries cannot be listed, selected, or analyzed directly.
A55	Message	Routine *<routine identifier>* is not unique.
	Type	ERROR
	Explanation	Re-execute the command (in line/batch mode) or GUI action (in X window mode) qualifying the routine with a filename. Use the format *<filename>*:*<routine>*.
A56	Message	Routine *<routine identifier>* is not selectable for analysis of *<collection type>* regions.
	Type	ERROR
	Explanation	Recompile the source file containing the routine with one of the CXpa compiler options then select and profile this routine.
A57	Message	No profilable source code regions could be found. Recompile with current compiler with a CXpa option.
	Type	ERROR
	Explanation	Your executable contains no source files compiled for profiling with Cxpa Recompile source files(s) and relink with one of the CXpa options to generate a new executable file.
A58	Message	Profiling routines incompatible or missing. Relink with current compiler or linker with a CXpa option.
	Type	ERROR
	Explanation	The profiling routines contained in your executable file are no longer compatible with the current version of CXpa or it was not compiled and linked for profiling with CXpa. Relink with one of the CXpa options to generate a new executable file.
A59	Message	No active PDF selected.
	Type	ERROR
	Explanation	Cannot create a new window until you select an active PDF. Select a PDF from the Active PDF list.
A60	Message	No loop, parallel, or block source code regions are available in *<filename>*.
	Type	INFO
	Explanation	The indicated PDF does not contain any profiling data for loop, parallel, or block source code regions. Only routine-level analysis is available for this PDF.

ID		Description
A61	Message	No <*metric level*> data to graph for <*metric region*> source code regions.
	Type	INFO
	Explanation	There was no data to graph for the source code region you specified. Try graphing a different metric or a different region.
A62	Message	PDF <*filename*> is already in the Analysis Control list; cannot add again.
	Type	ERROR
	Explanation	You can select this PDF from the list to create multiple windows.
A63	Message	Obsolete.
	Type	ERROR
	Explanation	None.
A64	Message	Obsolete.
	Type	ERROR
	Explanation	None.
A65	Message	Cannot save <*graph type*> to <*type identifier*> file. <*errno description*>.
	Type	ERROR
	Explanation	CXpa could not save the graph in the format you requested. Try saving it in a different format.
A66	Message	Exceeded maximum nesting level for CXpa command files.
	Type	ERROR
	Explanation	The command files you were executing contained more than 20 nesting levels. Redefine some of the command files to reduce the level of nesting.
A67	Message	No subcomplexes are available at this time.
	Type	ERROR
	Explanation	If needed, contact your system administrator for assistance in creating a new subcomplex.
A68	Message	Subcomplexes are not available on this architecture.
	Type	ERROR
	Explanation	Subcomplexes are only available on SPP Series machines.
A69	Message	Could not access subcomplex <*filename*>.
	Type	ERROR
	Explanation	Verify that the specified subcomplex exists by using the "info" command (in line/batch mode) or Info Session dialog (in X window mode). If you are still unable to set the subcomplex, please contact your system administrator for assistance.

ID		Description
C1	Message	Cannot open executable file <*filename*>.
	Type	ERROR
	Explanation	The indicated executable file could not be opened. Check to make sure the file exists and that you have permission to access it. If the file does exist, it might contain data that has been corrupted. In that case, try recompiling your program to create a new executable file.
C2	Message	Cannot open location range table for <*filename*>.
	Type	ERROR
	Explanation	Could not open the location range table (.lrt file) for your executable file. The .lrt file should reside in your local .CTI directory. If you have moved this file, use the "add path" command to specify the new location of the file. If the .lrt file does not exist (or if it exists but contains corrupted data), recompile your program with the appropriate option (-cxdb for CXdb and -cxpa, -cxpar, or -cxpab for CXpa) so that the compiler can generate the necessary CTI data files.
C3	Message	Cannot open source unit table for <*filename*>.
	Type	ERROR
	Explanation	:Could not open the source unit table (.sut file) for your executable file. The .sut file should reside in your local .CTI directory. If you have moved this file, use the "add path" command to specify the new location of the file. If the .sut file does not exist (or if it exists but contains corrupted data), recompile your program with the appropriate option (-cxdb for CXdb and -cxpa, -cxpar, or -cxpab for CXpa) so that the compiler can generate the necessary CTI data files.
C4	Message	Cannot open type and scope information table for <*filename*>.
	Type	ERROR
	Explanation	:Could not open the type and scope information table (.tsi file) for your executable file. The .tsi file should reside in your local .CTI directory. If you have moved this file, use the "add path" command to specify the new location of the file. If the .tsi file does not exist (or if it exists but contains corrupted data), recompile your program with the appropriate option (-cxdb for CXdb and -cxpa, -cxpar, or -cxpab for CXpa) so that the compiler can generate the necessary CTI data files.
C5	Message	Pathname <*filename*> is not a valid directory.
	Type	ERROR
	Explanation	The pathname you specified is not a directory. Try a different pathname.

ID		Description
C6	Message	Cannot open executable file for source file <*filename*>.
	Type	ERROR
	Explanation	Could not open the executable file associated with the indicated source file. Check to make sure the executable file exists and that you have permission to access it.
C7	Message	Cannot find source file for <*filename*> in search path.
	Type	ERROR
	Explanation	Could not find the source file associated with your executable file. If you have moved the source file since compiling it, use the "add path" command to specify the new location of this file.
C8	Message	Cannot find CTI data files for <*filename*> in search path.
	Type	ERROR
	Explanation	Cannot find the CTI (Compiler-Tools Interface) data files associated with your program. If you have moved these files, use the "add path" command to specify the new location of the files. If these files do not exist, recompile your program with the appropriate option (-cxdb for CXdb and -cxpa, -cxpar, or -cxpab for CXpa) so that the compiler can generate the necessary CTI data files.
C9	Message	Cannot open variable table for <*filename*>.
	Type	ERROR
	Explanation	Could not open the variable table (.vt file) for your executable file. The .vt file should reside in your local .CTI directory. If you have moved this file, use the "add path" command to specify the new location of the file. If the .vt file does not exist (or if it exists but contains corrupted data), recompile your program with the appropriate option (-cxdb for CXdb and -cxpa, -cxpar, or -cxpab for CXpa) so that the compiler can generate the necessary CTI data files.
C10	Message	Ambiguous file name <*filename*>. Use a qualified pathname.
	Type	ERROR
	Explanation	The specified source file name is ambiguous. Use a path name to qualify the source file name.
C11	Message	Source file <*filename*> is out of date. Last modified <*date*>; compiled <*date*>.
	Type	WARNING
	Explanation	The specified source file is out of date with respect to the version used for compilation. This is only a warning; the specified version of your source file will be used.

ID		Description
C12	Message	Object file <*filename*> compiled with early version of compiler.
	Type	WARNING
	Explanation	The indicated object file was compiled with a version of the compiler that did not provide all the features currently available in CONVEX products. This might limit some of your results. For maximum capability, recompile the indicated file with the CONVEX Fortran V9.0 (or later) compiler or the CONVEX C V6.0 (or later) compiler.
C13	Message	Cannot find file <*filename*> within current search path.
	Type	ERROR
	Explanation	Could not find the specified file in your current search path. Use the "add path" command to add directories to the search path.
C14	Message	Cannot open name space table for <*filename*>.
	Type	ERROR
	Explanation	Could not open the name space table (.ns file) for your executable file. The .ns file should reside in your local .CTI directory. If you have moved this file, use the "add path" command to specify the new location of the file. If the .ns file does not exist (or if it exists but contains corrupted data), recompile your program with the appropriate option (-cxdb for CXdb and -cxpa, -cxpar, or -cxpab for CXpa) so that the compiler can generate the necessary CTI data files.
C15	Message	Data file <*filename*> is out of date. Last compiled <*date*>; created <*date*>.
	Type	ERROR
	Explanation	The indicated data file is out of date with respect to the executable file. Try relinking the executable file to include the correct version of the data file. Use the "add path" command to add directories to the search path.
C16	Message	File <*filename*> compiled with prerelease compiler.
	Type	WARNING
	Explanation	The indicated source file was compiled with a prerelease version of the Fortran compiler. Recompile this file to obtain full symbolic debugging information.
C17	Message	Cannot read line <*count*> from file <*filename*>.
	Type	ERROR
	Explanation	Could not read from the indicated source file. If you have modified this file since its last compilation, try compiling the file again.

ID		Description
C18	Message	Cannot find CTI expression table information for *<filename>*.
	Type	ERROR
	Explanation	Could not open the expression table (.xpt file) for your executable file. The .xpt file should reside in your local .CTI directory. If you have moved this file, use the "add path" command to specify the new location of the file. If the .xpt file does not exist (or if it exists but contains corrupted data), recompile your program with the appropriate option (-cxdb for CXdb and -cxpa, -cxpar, or -cxpab for CXpa) so that the compiler can generate the necessary CTI data files.
C19	Message	File *<filename>* is not a valid object file. *<errno description>*
	Type	ERROR
	Explanation	The specified file is not a valid object file, for the reason indicated. Try a different object file name.
C20	Message	Cannot find CTI call relation table information for *<filename>*.
	Type	ERROR
	Explanation	Could not find the call relation table information for your executable file. This information should reside in the CTI data files in your local .CTI directory. If you have moved the CTI data files, use the "add path" command to specify the new location of these files. If the CTI data files do not exist (or if they exist but contain corrupted data), recompile your program with the appropriate option (-cxdb for CXdb and -cxpa, -cxpar, or -cxpab for CXpa) so that the compiler can generate the necessary CTI data files.
C21	Message	Cannot find CTI backend information for *<filename>*.
	Type	ERROR
	Explanation	Could not find the CTI backend information (.be file) for your executable file. The .be file should reside in your local .CTI directory. If you have moved this file, use the "add path" command to specify the new location of the file. If the .be file does not exist (or if it exists but contains corrupted data), recompile your program with the appropriate option (-cxdb for CXdb and -cxpa, -cxpar, or -cxpab for CXpa) so that the compiler can generate the necessary CTI data files.

ID		Description
C22	Message	Cannot find CTI monitor descriptor table information for *<filename>*.
	Type	ERROR
	Explanation	Could not find the monitor descriptor table information for your executable file. This information should reside in the CTI data files in your local .CTI directory. If you have moved the CTI data files, use the "add path" command to specify the new location of these files. If the CTI data files do not exist (or if they exist but contain corrupted data), recompile your program with the appropriate option (-cxdb for CXdb and -cxpa, -cxpar, or -cxpab for CXpa) so that the compiler can generate the necessary CTI data files.
D1	Message	Stub table not found in current executable.
	Type	ERROR
	Explanation	A critical data structure could not be found in your current program. This prevents the backtrace command from working. You may either continue debugging cautiously or try recompiling and relinking your application. If this persists, please submit a problem report.
D2	Message	The unwind table not found in current executable.
	Type	ERROR
	Explanation	A critical data structure could not be found in your current program. This prevents the backtrace command from working. You may either continue cautiously or try recompiling and relinking your application. If this persists, please submit a problem report.
D3	Message	Current executable is unreadable.
	Type	ERROR
	Explanation	The current executable cannot be read. No further work can continue on this executable. Try recompiling and relinking your application. If this persists, please submit a problem report.
D4	Message	File *<filename>* is not a core file.
	Type	ERROR
	Explanation	The given file is not recognized as a core file. A core file is a file created when your program aborts unexpectedly. No other file can be a core file. If this file was created by an unexpected abort, then it may have been corrupted. Recreate a new core file by running your program again and be sure that you have enough disk space. If this problem persists, please submit a problem report.

ID		Description
D5	Message	Cannot find symbolic support in current executable.
	Type	ERROR
	Explanation	The current executable does not have any symbolic support. Both debugging and performance analysis may precede; however, many important features will not work such as loop instrumentation, source code to program counter correlation, and printing user variables. If you want these features, recompile your program with the appropriate option (-cxdb for CXdb and -cxpa, -cxpar, or -cxpab for CXpa).
S1	Message	Cannot seek data in process memory. Errno: *<errno>*; address: *<address>*
	Type	ERROR
	Explanation	Operation could not be performed. There is no recovery.
S2	Message	Cannot read data from process memory. Errno *<errno>*; address *<address>*.
	Type	ERROR
	Explanation	Operation could not be performed. There is no recovery.
S3	Message	Partially read data from process memory
	Type	WARNING
	Explanation	Operation could not be performed. There is no recovery.
S4	Message	Cannot write data to process memory. Errno *<errno>*; address *<address>*.
	Type	ERROR
	Explanation	Operation could not be performed. There is no recovery.
S5	Message	Partially written data to process memory
	Type	WARNING
	Explanation	Operation could not be performed. There is no recovery.
S6	Message	Process is executing.
	Type	ERROR
	Explanation	Operation could not be performed. There is no recovery.
S7	Message	Process is not executing.
	Type	ERROR
	Explanation	Operation could not be performed. There is no recovery.
S8	Message	Process is not paused.
	Type	ERROR
	Explanation	Operation could not be performed. There is no recovery.

ID		Description
S9	Message	Process not detached.
	Type	ERROR
	Explanation	Operation could not be performed. There is no recovery.
S10	Message	Process is detached.
	Type	ERROR
	Explanation	Operation could not be performed. There is no recovery.
S11	Message	Process no longer exists.
	Type	ERROR
	Explanation	Operation could not be performed. There is no recovery.
S12	Message	Breakpoint already exists at addr *<address>*.
	Type	ERROR
	Explanation	Operation could not be performed. There is no recovery.
S13	Message	Cannot access executable *<filename>*. *<errno description>*.
	Type	ERROR
	Explanation	Check the permissions on this file.
S14	Message	fork() system call failed. *<errno description>*.
	Type	ERROR
	Explanation	Operation could not be performed. There is no recovery.
S15	Message	exec() system call failed.
	Type	ERROR
	Explanation	Operation could not be performed. There is no recovery.
S16	Message	Process cannot be terminated. *<errno description>*.
	Type	ERROR
	Explanation	Operation could not be performed. There is no recovery.
S17	Message	Process cannot be attached to. *<errno description>*.
	Type	ERROR
	Explanation	Operation could not be performed. There is no recovery.
S18	Message	Process cannot be detached. *<errno description>*.
	Type	ERROR
	Explanation	Operation could not be performed. There is no recovery.

ID		Description
S19	Message	Process cannot be continued. *<errno description>*.
	Type	ERROR
	Explanation	Operation could not be performed. There is no recovery.
S20	Message	Process cannot be closed. *<errno description>*.
	Type	ERROR
	Explanation	Operation could not be performed. There is no recovery.

Index

Symbols

% CPU field in events reports 131
  discussed, for parallel regions 129
.cxpainit start-up file 247
=> symbol in Source Code window 19, 224
2D Profile button 18, 173
2D profile graphs
  customizing 197
  displaying associated source code 146
  saving to a file 205
  scaling options 213
  sorting options 219
2D Profile window 145
3D Profile button 18
3D profile graphs
  customizing 197
  displaying associated source code 152
  rotating 152
  saving to a file 205
  sorting options 219
3D Profile window 151

A

abbreviations, command 233
Abort button 173
About CXpa dialog 157
active PDF 30
active PDF, selecting 160
`add path` command 235
Adder functional unit 118
adding a directory to CXpa's search path 227
All metrics button (Analysis Report window) 165
All regions button (Analysis Report window) 164
All/None buttons (Profile Selection dialog) 203
Analysis Control window 29, 159
Analysis Report window 163
`analyze` command 237
analyzing PDFs only
  in line mode 30
  in X window mode 29
annotations
  in Source Code window 224
  source code region 54
application/library visibility
  setting in line mode 297
  setting in X window mode 231
arguments, program
  specifying in X window mode 170
  specifying on command line 32
assistance, technical xiv
associated documents xiv
audience xi

B

`b` annotation for deselected basic block region 224
`B` annotation for selected basic block region 224
Basic Block report 95
  fields in, defined 97
batch mode
  instructions for using 31
  overview 9
  sample shell script for batch execution 33
blocks, basic
  as a region type 44
  Basic Block performance report 95
  defined 44
Browse buttons (Help window) 36, 180
buttons
  2D Profile 173
  Abort 173
  All/None (Profile Selection dialog) 203
  Change File (in Source Code window) 224
  Continue 173
  Create 2D Profile 161
  Create 3D Profile 161
  Create Report 161
  Customize (2D profile) 147
  Customize (3D profile) 152
  Customize (report) 165
  Pause 172
  Profile Selection 172
  Reset Zoom 155
  Start 172
  SubComplex Selection (SPP Series only) 172
  Zoom In 155
  Zoom Out 155

C

`-c` compiler option 45
cache miss events
  for C4 Series architectures 76
  for SPP Series architectures 76
Change File button (in Source Code window) 224

changing CXpa's search path
add path command 235
in X window mode 227
path command 273
Chime Count 105, 108
chime counts 112
chimes 121
calculating 121
chime counts metrics 74
chore, defined 131
collect command 241
collecting metrics
for events, in line mode (set events command) 289
in line mode (collect command) 241
in X window mode (Profile Selection dialog) 199
command files
example in batch mode 31
executing at CXpa start-up 248
executing with the source command 299
format 31
input using the -x option 31
command line editing 22
command syntax xii
commands
abbreviations 233
add path 235
analyze 237
collect 241
continue 245
cxpa 247
deselect 255
help 259
info 263
introduction 233
list 265
list selectable 269
path 273
quit 275
rerun 277
run 281
select 285
set events 289
set pdf 293
set subcomplex 295
set visibility 297
source 299
stop 301
version 303
compiler options
-c 45
-cxpa 256
-cxpab 95
-cxpalib, using 50
-cxpamon, using 50
-cxpar 256
for profiling 249
listed 43
-O1 44
-O3 44
optimization, related to profiling 43
-no 43
-O0 43
-O1 43
-O2 43
-O3 43
-p 44
-pb 44
-pg 44
profiler, described 44
compiling
advanced topics 49
and linking in one step 45
and linking separately 45
options. *see* compiler options
syntax 43
concurrency
factor for parallel regions (CPU/Wall) 127
for parallel regions 131
contact utility xv
Contents button (Help window) 36, 180
Continue button 173
continue command 245
controlling execution
continue command 245
rerun command 277
run command 281
stop command 301
using buttons on Executable Manager window 172
CPU time 73
CPU/Wall clock time 74
CPU/Wall metric
discussed, for parallel regions 128, 131
Create 2D Profile button 161
Create 3D Profile button 161
Create Report button 161
current list file name, displaying
in X window mode 187
using the info command 264
Current PDF field 160
current PDF name, displaying
in X window mode 185
using the info command 263
current PDF, selecting 160
Current Process State 183
listed in output of info command 263
current working directory, listed 187
Customize (2D profile) button 147
Customize (3D profile) button 152
Customize (report) button 165
Customize Report dialog 167
customizing
2D and 3D profiles 197
reports 164, 167

cxpa command 247
-cxpa CXpa compiler option 43
CXPA environment variable 250
-cxpab CXpa compiler option 43
.cxpainit start-up file 247
-cxpalib CXpa compiler option 43, 50
-cxpamon CXpa compiler option 43, 50
cxpamon.o
  specifying alternate location with -cxpamon 50
-cxpar CXpa compiler option 43

D

data cache accesses, defined 76
data cache misses, defined 76
data_accesses 87
  parameter of the set events command 290
data_misses 87
  parameter of the set events command 290
deselect command 63, 255
dialogs
  About CXpa 157
  Customize Report 167
  Filter Profile 175
  Filter Report 177
  Info Executable 183
  Info PDF 185
  Info Session 187
  New PDF 189
  Open PDF 193
  Profile Customization 197
  Profile Selection 58, 199
  Save Profile 205
  Save Report 209
  Scale 213
  Scheduling Selection 217
  Sort 219
  Source Code Selection 221
  Source Search Path 227
  Subcomplex Selection 229
  Visibility Selection 231
Directories field in New PDF dialog 190
Directories field in Save Report dialog 210
directory
  adding to CXpa's search path 227, 235
  current working directory, listed 187
  removing from CXpa's search path 228
  replacing CXpa's search path with the path command 273
displaying reports in X window mode 163
Divider functional unit 118
documentation
  associated Convex documentation xiv
  ordering information xiv

E

-e CXpa start-up option 32, 247
e profiling status 131
editing the command line 22
Elapsed Wall Time field 172
enabling event collection
  in line mode (collect command) 241
  in X window mode (Profile Selection dialog) 200
environment variable, CXPA 250
ERROR message type 305
error messages
  displaying explanations for (in line mode) 260
  listed and described by number 307
Event Counter 1 options menu 202
Event Counts metrics 74
Event Latency Time metrics 74
Event Latency/Counts metrics 75
Event Latency/CPU metrics 75
events
  enabling event collection
    in line mode using the collect command 241
    in X window mode (Profile Selection dialog) 199
  selecting
    in line mode using set events command 289
    in X window mode 79
    using Profile Selection dialog 79
    using the collect and set events commands 85
Events metrics
  for C4 Series architectures 76
  for routines 138
  for SPP Series architectures 75
Exclude Child/Inner Metrics button 175
Executable field
  on the Analysis Control window 161
  on the Executable Manager window 172
executable file
  creation date, displaying in line mode 263
  creation date, displaying in X window mode 183
  current, displaying in X window mode 183
  displaying information about
    in line mode (info command) 263
    in X window mode 183
Executable Manager window 169
executable. *see* files, executable
execution, controlling
  continue 245
  rerun command 277
  run 281
  stop command 301
Execution/iteration counts metrics 73
exiting CXpa
  in X 20
  quit command 275

F

-f CXpa start-up option 247
FATAL message type 305
fields
  current PDF 160
  Elapsed Wall Time 172
  Executable 161, 172
  Filter 190, 206, 210
  Finished State 161
  Open PDF 160
  PDF 172
  Process State 172
  Program Arguments 172
  Scheduling 172
  SubComplex 172
  Window Count 161
files
  .cxpainit (start-up file) 247
  .pdf 25, 293
  executable
    compiling 44
    creation data, displaying 183
    displaying information about (X window mode) 183
    displaying information in line mode 263
  object 45
  PDF
    setting in line mode 26, 293
    setting in X window mode 25
  search path for, setting with add path command 235
  source code
    list command 265
    listing (in line mode) 265
    listing (in X window mode) 223
    selecting files to display (X window mode) 221, 224
    Source Code window 223
Filter button 190, 206, 210
Filter Profile dialog 175
Filter Report dialog 177
Finished State field 161
fixed scheduling 247
  -f option of cxpa command 247
  setting from the command line 247
  setting in X window mode 217
format of PDF listed
  in line mode 263
  in X window mode 185
functional units
  C2 and C3 Series 118
  C4 Series 119
  defined 113
  described 117

G

g profiling status 131
getting help online (in line mode) 259
Go Back button (Help window) 36, 180
graphs
  2D Profile 146
  3D Profile 151
  customizing 197
  rotation in 3D Profile window 152
guidelines for using CXpa 5

H

help command 259
help using CXpa's window mode 35
Help window 179
  buttons 180
  creating 181
help, online
  contents, displaying 180
  displaying
    in line mode (help command) 259
    in X window mode 180
  instructions (accessing) 180
  printing online help text 180
  searching 181

I

Include Child/Inner Metrics button 175
info command 263
Info Executable dialog 183
INFO message type 305
Info PDF dialog 185
Info Session dialog 187
instruction cache misses, defined 76
instruction_misses 87
  parameter of the set events command 290
Instrumentation Version
  displaying in line mode (info command) 263
  of executable 183
  of PDF 186
Instrumenting Executable dialog 17
interfaces to CXpa 8
introduction
  learning CXpa quickly 15
  to CXpa and profilers 3
invoking CXpa
  in line mode 20
  in window mode 15
  man page 247
  with a PDF only 159
    in line mode 30
    in X window mode 29

K

key bindings, for command line editing (line mode) 22

L

l annotation for deselected loop regions 224
L annotation for selected loop regions 224
latency field in events reports 131
latency metrics 74
learning CXpa quickly 15
libraries, system
  and -cxpalib compiler option 50
  instrumented for CXpa 50
  setting visibility (in line mode) 297
  setting visibility (in X window mode) 231
library/application visibility
  setting in line mode 297
  setting in X window mode 231
limitations, of CXpa 11
line mode
  changing the PDF name (set pdf command) 293
  creating reports with analyze command 237
  description in man page 250
  deselecting regions for profiling 255
  displaying
    information for a CXpa session 263
    online help (help command) 259
    source code (list command) 265
  invoking CXpa 247
  key bindings (command line editing) 22
  learning CXpa quickly 20
  -nw (no windows) option 247
  overview 9
  quitting CXpa 275
  selecting regions for profiling (select command) 285
  specifying metrics to collect (collect command) 241
  specifying type of event to collect 289
linking. *see* compiling, linking
list command 265
list file name, displaying
  in X window mode 187
  using the info command 264
list selectable command 269
listing
  regions available for profiling
    in line mode using list selectable command 269
  source files 265
Loader functional unit 118
local and remote memory misses
  defined 76
local memory 75
local memory misses
  defined 76
local_and_remote_misses 87
  parameter of set events command 289
local_misses 86
  parameter of set events command 289
Loop reports
  discussed 99
  displaying in line mode (analyze command) 238
  displaying in X window mode 164
loops
  as a region type 44

M

m profiling status 132
messages, CXpa
  displaying online help for 260, 305
  introduction 305
  listed and described by number 307
metrics
  available on all architectures 73
  available on C Series machines only 74
  available on C4 and SPP Series machines only 74
  described 73
  discussed 73
  events
    selecting in X window mode 79
  hardware events (SPP and C4 Series only) 74
  latency (SPP Series only) 74
  selecting
    in line mode 85
    in X window mode 200
  selecting with the collect command 241
  sorting in 2D and 3D profile graphs 219
Mflops 105, 108
  estimated 74
  vector 74
Multiplier functional unit 118

N

New PDF dialog 189
New PDF menu option 25
-no compiler optimization option 43
notational conventions xii
-nw (no windows) CXpa start-up option 247

O

-O1 compiler optimization option 43, 44
-O2 compiler optimization option 43
-O3 compiler optimization option 43
object file. *see* files, object

online help, using
in line mode 259
in X window mode 35
see also help, online 35
Open PDF dialog 193
Open PDF field 160
opening PDFs 160
optimization
associated compiler documentation xiv
Optimized Loops section (reports) 127
options
compiler
-cxpa 256
-cxpab 95
-cxpalib 50
-cxpamon 50
-cxpar 256
optimization levels and profiling 43
CXpa start-up
-e 247
-f 247
-help 247
listed 247
-nw 247
-nx 247
-path 248
-pdf 248
-pid 248
-stack bytes 248
-w 248
-x 248
specifying with CXPA environment variable 250
ordering documentation xiv
organization xi
Original Loops section (reports) 127
overview
available metrics 7
batch mode interface 9
CXpa 7
interfaces to CXpa 8
line mode interface 9
performance reports 10
profiles and graphs 9
X window interface 8

P

p annotation for deselected parallel region 224
P annotation for selected parallel regions 224
-p compiler option 44
p profiling status 132
page table cache misses, defined 76
page_table_misses 87
parameter of the set events command 290
parallel performance graph 151
Parallel Region performance reports 127
creating 132
displaying in line mode (analyze command) 238
displaying in X window mode 164
fields in, defined 130
parallel regions
as a region type 44
path command 273
-path CXpa start-up option 247
Pause button 172
pausing a program
in line mode 245
-pb compiler option 44
PC Value 97
PDF (performance data file)
active 160
active PDF, selecting (X window mode) 30
analyzing multiple PDFs (in X window mode) 29
changing name to prevent overwriting of existing PDF 25, 293
controlling from the Analysis Control window 159
creating with the New PDF dialog 189
CXpa version created with listed
in line mode 263
in X window mode 185
default name (*<executable>*.pdf) 25
described 25
displaying information about
in line mode (info command) 263
in X window mode 185
executable creation date 185
executable listed 185
format version listed
in line mode 263
in X window mode 185
Info PDF dialog 185
instrumentation version listed 186
invoking CXpa with PDF only 159
listing creation date
in line mode 263
in X window mode 185
listing the current PDF
in X window mode 185
using the info command 263
opening 160
Open PDF dialog 193
other PDFs (line mode) 30
other PDFs (X window mode) 30
process state listed (in X window mode) 185
setting
at CXpa start-up (-pdf option) 248
in line mode 26
in X window mode 25
with the set pdf command 293
visibility setting listed 186
-pdf CXpa start-up option 29, 30, 247
PDF field (Executable Manager window) 172

performance reports
saving to a file (X window mode) 209
-pg compiler option 44
-pid CXpa start-up option 248
preface xi
Process button 177
Process ID 183
Process State field 172
process state of PDF 185
process, current state
displaying in line mode 263
displaying in X window mode 183
product number
displaying in X window mode 157
displaying with the version command 303
Profile Customization dialog 197
Profile Selection button 172
Profile Selection dialog 199
selecting metrics to collect 80, 200, 201
selecting regions to profile 200
specifying an event metric to collect 202
Profile selection dialog
selecting regions to profile 201
profilers
defined 3
prof 4
profiling
benefits 4
by statistical sampling 4
learning CXpa quickly 15
overview 3
regions 44
profiling a program 170
profiling status (PS) field in reports 131
program
executing with the run command 281
reexecuting, with the rerun command 277
stopping execution in line mode 301
program arguments
specifying on command line with -e option 32
Program Arguments field 172
PS (profiling status)
for loops 106
PS (profiling status) field in performance reports 131

Q

quit command 275
quitting CXpa
in X 20
with the quit command 275

R

r annotation for deselected routine regions 224
R annotation for selected routine region 224
read misses, defined 75
read_only 87
parameter of set events command 289
redirection
of program I/O 281
reports 240
redirection operators
using with analyze command 237
using with rerun command 277
using with run command 281
regions, source code
annotations 54
described 53
deselecting with deselect command 255
selecting
in X window mode 200
selecting for profiling 55
in line mode 63
in X window mode 57
using the select command 285
types of 44
Related Commands, defined 233
release notice, CXpa
displaying in line mode 259
remote memory 75
remote memory misses
defined 76
remote_misses 86
parameter of set events command 289
removing a directory from CXpa's search path 228
reporting problems xv
reports
Analysis Report window 163
Basic Block 95
fields in, defined 97
creating or viewing 93
customizing, in X window mode 167
displaying
in line mode with analyze command 238
in X window mode 164
filtering data in
line mode 92
X window mode 92
header information 93
library/application visibility setting 92
Loop 99
Computation 100
creating 108
Events 103
fields in 106
Optimized Loops section 99
Original Loops section 99
Time to Solution 101
Vector Register Utilization 105
Optimized Loops section 127
Original Loops section 127
overview 91

Parallel Region 127
displaying in X window mode 164
Events (C4 and SPP Series only) 129
fields in, defined 130
Time to Solution 127
redirecting to a file (in line mode) 240
Routine 135
Computation 136
Events (C4 and SPP Series only) 137
fields in, described 139
Time to Solution 136
saving to a file (in X window mode) 209
thread/process visibility setting 92
`rerun` command 277
Reset Zoom button 155
resuming
data collection 245
execution 245
Routine reports
Computation 136
described 135
displaying in line mode (`analyze` command) 238
displaying in X window mode 164
Events (C4 and SPP Series only) 137
fields in, defined 139
Time to Solution 136
routines
as a region type 44
calling uninstrumented routines 49
setting library/application visibility (in X) 231
`run` command 281
running a program under CXpa
in line mode 281

S

Save Profile dialog 205
Save Report dialog 209
Scale dialog 213
X-axis scaling 213
Y-axis scaling 214
Scheduling field 172
Scheduling Selection dialog 217
scheduling, fixed 247
`-f` option of the `cxpa` command 247
setting from the command line 247
setting in X window mode 217
script, batch mode execution example 33
Search button (Help window) 36, 181
Search Field (Help window) 36
search path
adding to with the `add path` command 235
changing (in X window mode) 227
current, listed 187
`-path` CXpa start-up option 248
setting with the `path` command 273
searches
online help searches (titles or all text) 181
`select` command 63, 285
`select pregion` example 287
selecting
events
in X window mode 81
events to collect
in line mode 289
metrics to collect
in X window mode 79, 199
regions to profile
in line mode 63
in X window mode 58, 59, 199
selecting metrics to collect
in line mode (`collect` command) 241
session information, displaying 187
`set events` command 289
parameters (for C4 Series) 87
parameters (for SPP Series) 86
`set pdf` command 293
`set subcomplex` command 295
`set visibility` command 297
using to filter report data 92
setting the PDF 25
Sort dialog 219
source code
annotations 54, 224
correlation in 2D profile graph 146
correlation in 3D profile graph 152
displaying (in line mode) 265
displaying associated code from profile windows 19
displaying in X window mode 54
`list` command 265
`list selectable` command 269
search path, modifying (X window mode) 227
selecting source files to display (X window mode) 221, 224
source code regions
annotations for 54
described 53
deselecting in line mode 63
deselecting with the `deselect` command 255
listing regions that can be profiled (X window mode) 224
selecting for profiling 55
in line mode 63
in X window mode 57
Profile Selection dialog 200
Source Code Selection dialog 221
Source Code window 223
`source` command 299
source files
listing in line mode 265
replacing search path with the `path` command 273
Source Search Path dialog 227
`-stack bytes` CXpa start-up option 247

stack size needed for CXpa 248
Start button 172
starting CXpa 247
steps for learning CXpa 5
stop command 301
stopping a program
  using the stop command 301
subcomplex
  listing available, in line mode 264
  selecting in line mode 295
  selecting, in X window mode 229
SubComplex field 172
SubComplex Selection button (SPP Series only) 172
Subcomplex Selection dialog 229
syntax
  conventions xii
  defined 233
system libraries, instrumented
  linking with -cxpalib compiler option 50
  setting visibility 50

T

t profiling status 132
Technical Assistance Center (TAC) xiv
Thread button 177
thread visibility
  setting in line mode 298
  setting in X window mode 177
Titles/All Text button (Help window) 36
Titles/All Text searches in Help window 36, 181

U

u profiling status 132
using CXpa
  in line mode 20
  in X window mode 15
using this book xi

V

vector chaining 113
Vector Flops 105, 108
vector functional units
  C2 and C3 Series 118
  C4 Series 119
vector load/store events, defined 76
vector Mflops 123
  calculating estimated 123
  estimated Mflops metric 74
  events metrics (C Series) 74
Vector Spills 105
vector spills 74
vector_load_stores 87
  parameter of the set events command 290
version command 303
version number of CXpa
  displaying in X window mode 157
  displaying with the version command 303
visibility
  application/library
    and -cxpalib compiler option 231
    setting in line mode 297
    setting in X window mode 231
  process/thread
    setting in line mode 298
    setting in X window mode 177
  settings
    displaying in X window mode 186
Visibility Selection dialog 231

W

wall clock time 73
WARNING message type 305
Window Count field 161
window mode overview 8
windows
  2D Profile 145
  3D Profile 151
  About CXpa dialog 157
  Analysis Control window 159
  Analysis Report 163
  Customize Report dialog 167
  Executable Manager 169
  Filter Profile dialog 175
  Filter Report dialog 177
  Help 179
  Info Executable dialog 183
  Info PDF dialog 185
  Info Session dialog 187
  introduction 143
  New PDF dialog 189
  Open PDF dialog 193
  Profile Customization dialog 197
  Profile Selection dialog 199
  Save Profile dialog 205
  Save Report dialog 209
  Scheduling Selection dialog 217
  Sort dialog 219
  Source Code 223
  Source Code Selection dialog 221
  Source Search Path dialog 227
  Subcomplex Selection dialog 229
  Visibility Selection dialog 231
working directory, listed 187
write misses, defined 75
write_only 87
  parameter of the set events command 289

X

-x CXpa start-up option 31, 247
x profiling status 132
X resource, controlling automatic window creation 160
X Toolkit options 248
X window mode
  description in man page 249
  overview 8
  using 170

Y

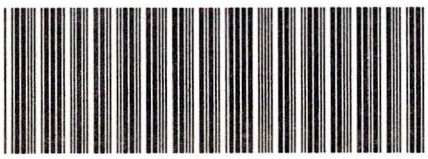
y profiling status 132

Z

z profiling status 132
Zoom In button 155
Zoom Out button 155

ORDER NUMBER
DSW-253

DOCUMENT NUMBER
710-004730-005



CONVEX PRESS