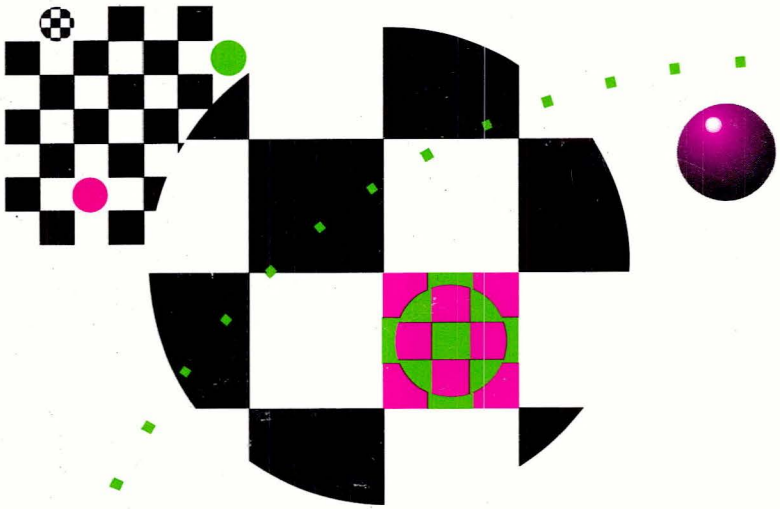


CONVEX

CXpa Reference for Exemplar Systems

First Edition

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

CXpa Reference for Exemplar Systems

Order No. DSW-605

First Edition
June 1995

CONVEX Press
Richardson, Texas
United States of America

CXpa Reference for Exemplar Systems

Order No. DSW-605

Printed in the United States of America

Revision information for

CXpa Reference for Exemplar Systems

Edition	Document No.	Description
First	710-004730-008	Initial release, June 1995. Released with V3.3 of Convex CXpa software. For Exemplar SPP Series systems, this book replaces the *CXpa Reference*, Second Edition (DSW-253).

Contents

Part 2 Reference pages

Using this book

Purpose and audience

The *CXpa Reference for Exemplar Systems* is both a user's guide and a reference. This book introduces new users to CXpa and describes the metrics, graphs, reports, commands, and interfaces that are available in the Convex CXpa software. This book 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.
- **Glossary**
- **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 1200 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 the following Convex books:

- ***Exemplar Programming Guide*** (Order No. 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 SPP 1000/1200 Architecture*** (Order No. DHW-014) — Describes the Convex implementation of scalable parallel processing on Exemplar SPP 1000 and SPP 1200 machines.
- ***CXdb Reference*** (Order No. DSW-474) — Comprehensive reference for the Convex Visual Debugger, CXdb.
- ***Fortran Language Reference*** (Order No. DSW-037)
- ***Fortran User's Guide*** (Order No. DSW-037)
- ***C User's Guide*** (Order No. DSW-086)

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 (DSW or DHW) 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 the Technical 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
- 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 minutes 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 wall clock time for your program.

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

Step 5 To get further insight into the performance of that routine, try profiling the loops in that routine and identify the loop that took the longest to execute.

Step 6 Try rewriting the loop (or routine) in another way. Refer to the *Exemplar Programming Guide* (DSW-067) for more information about optimizing your code.

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

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

Overview of CXpa

CXpa is an interactive runtime performance analyzer for programs compiled by the Convex C and FORTRAN compilers. 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. Performance information can be viewed for individual threads of execution or across the entire process.

Available metrics

By default, CXpa measures CPU time, wall clock time, and iteration/execution counts for selected regions of your program. The following additional metrics are provided:

- Memory access event counts
- Latency time for memory access events (latency is the time it takes to satisfy a memory access request)

Available memory access event counts and latency metrics differ according to machine type, as described in the following sections.

SPP 1000 off-processor events and latency metrics

On SPP 1000 machines, performance counters located on the CPU agent chip (off-processor) can be configured to collect cache miss event counts and latency metrics 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). This means that the memory reference request was sent across the hypernode memory crossbar.
- 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). This means that the memory reference request was sent across the CTI (Convex Coherent Toroidal Interconnect).
- Local and remote cache misses combined
- Event latency time, event latency/CPU time, and event latency/counts metrics for the above events

You can also specify whether the above off-processor events are collected during read accesses (loads) only or write accesses (stores) only. By default, events are collected during reads and writes.

SPP 1200 on-processor events and latency metrics

On SPP 1200 machines, event counters on the HP PA-RISC 7200 Series processors (on-processor event counters) can be configured to collect the following events:

- Data cache miss counts and latency, data cache access counts, average data cache miss latency, and data cache hit rates
- Instruction cache miss counts and latency, and average instruction cache miss latency
- Completed instruction counts, CPU clock cycles, average number of CPU cycles per instruction, and the MIPs rate

Refer to the "Introducing metrics" online help topic or section of this book for more information about metrics available on each architecture.

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:

- 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 visualize 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 access 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 that are graphed. The 2D Profile graphs the data per process, while the 3D Profile graphs the data per thread (for parallel codes).

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 zooming 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 and 3D profile graphs allow you to view or specify the range or number of data values displayed at one time in the 2D or 3D 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 for printing or export**—You can save profile graphs in PostScript or xwd formats for printing or to ASCII format for export to other graphic packages.
- **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
- Serial loops
- 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:

- **Counts**—Includes loop iteration/execution counts or routine call counts.
- **Computation**—Includes CPU time and % total CPU time metrics.
- **Time to Solution**—Includes wall clock time and CPU/wall clock time metrics.
- **Events**—Available event reports and metrics differ according to machine architecture:
  - Off-processor event reports (SPP 1000 only)—Locally resolved cache misses, remotely resolved cache misses, or locally and remotely resolved cache misses.
  - On-processor event reports (SPP 1200 only)—Data cache misses and latency; data cache accesses; instruction cache misses and latency; and instructions completed and clock cycles.

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.
- 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 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
```

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. For example:

   ```
   setenv DISPLAY mydisplay:0.0
   ```

2. Compile and link your program with one of the following options:

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

   For example:

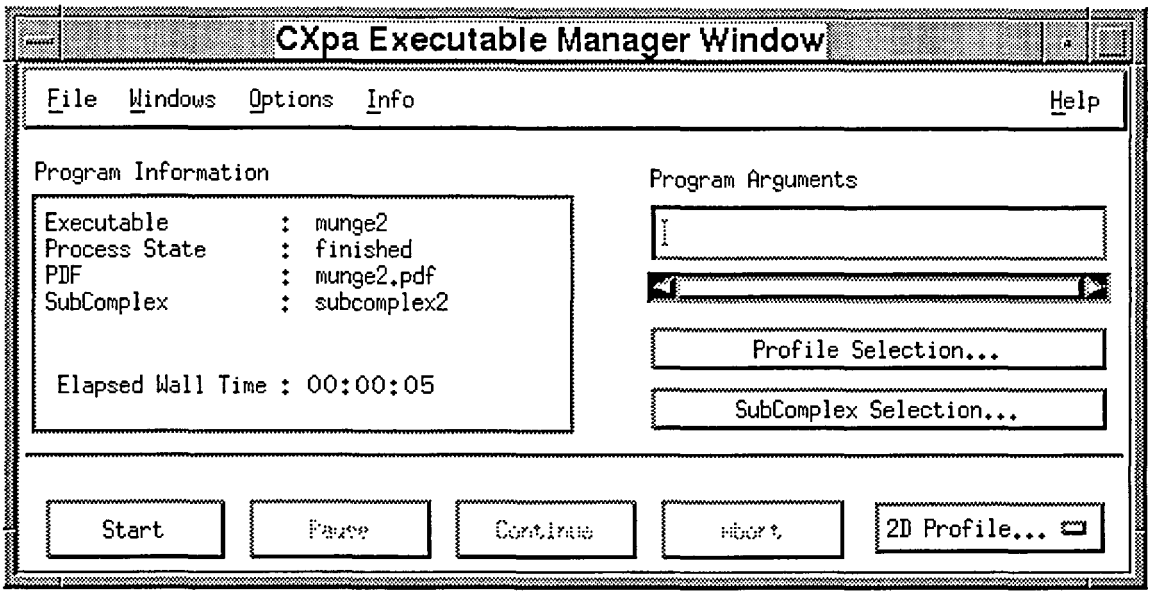
   ```
   fc -cxpa prog.f
   ```

3. Invoke CXpa with an executable:

   ```
   % cxpa a.out &
   ```

CXpa is a licensed product. If you do not acquire a license after starting CXpa, contact the system administrator at your site for more information.

The Executable Manager window appears.

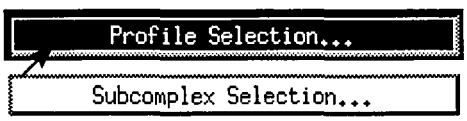


By default, CXpa collects CPU time, wall clock time, and iteration/execution counts for all routines in your program. Use these defaults the first time you profile a program using CXpa. To use the defaults, skip to step 5.

NOTE: You will obtain the most accurate profiling results if you run the program in a standalone environment (that is, on a "quiet" system or a dedicated system).

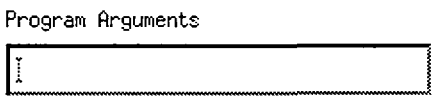
To select different source code regions (such as serial or parallel loops) for profiling and/or collect different metrics (such as events), continue with step 4.

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.

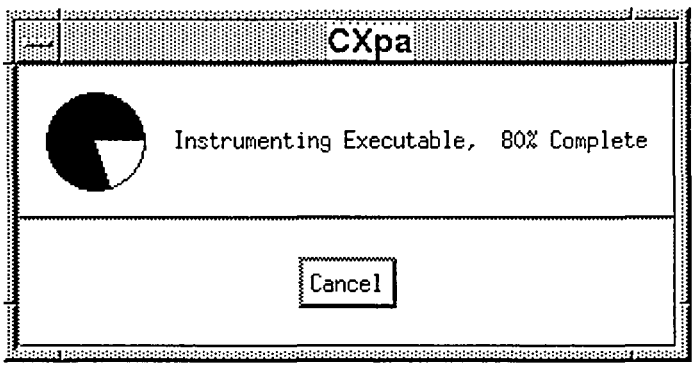


NOTE: Profiling time increases as you select more regions and metrics.

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



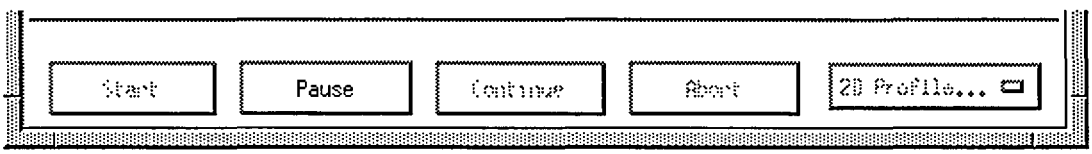
6. Press the Start button. An xterm window appears to display your program's input and output. A dialog also appears while CXpa is instrumenting your executable.



Large programs may take a while to instrument. Once your program is instrumented, it starts executing, and the dialog disappears.

- To stop instrumentation and cancel the run, press the Cancel button.
- To suspend program execution during profiling, press the Pause button at the bottom of the Executable Manager window.

NOTE: To obtain the most accurate profiling data, do not pause your program during profiling. For best results, run the program to completion and then analyze the results.

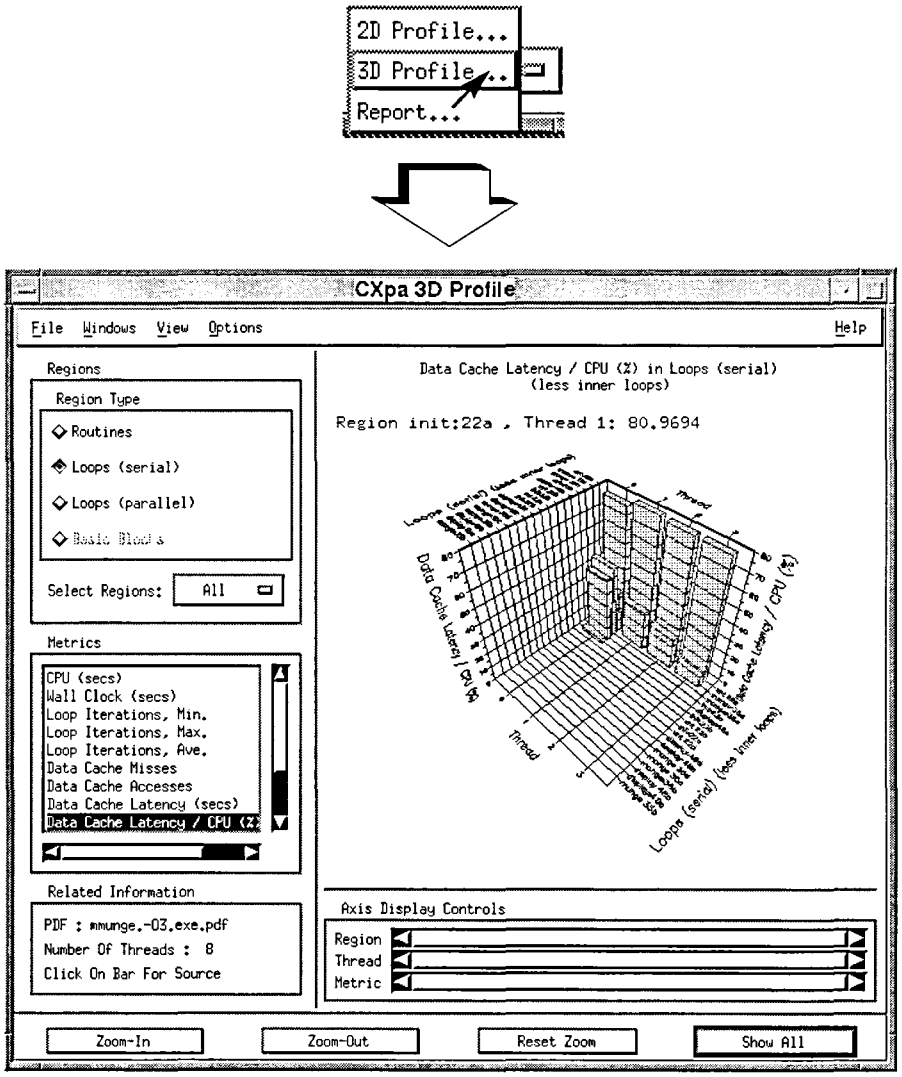


Once your program is paused, you can:

- 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

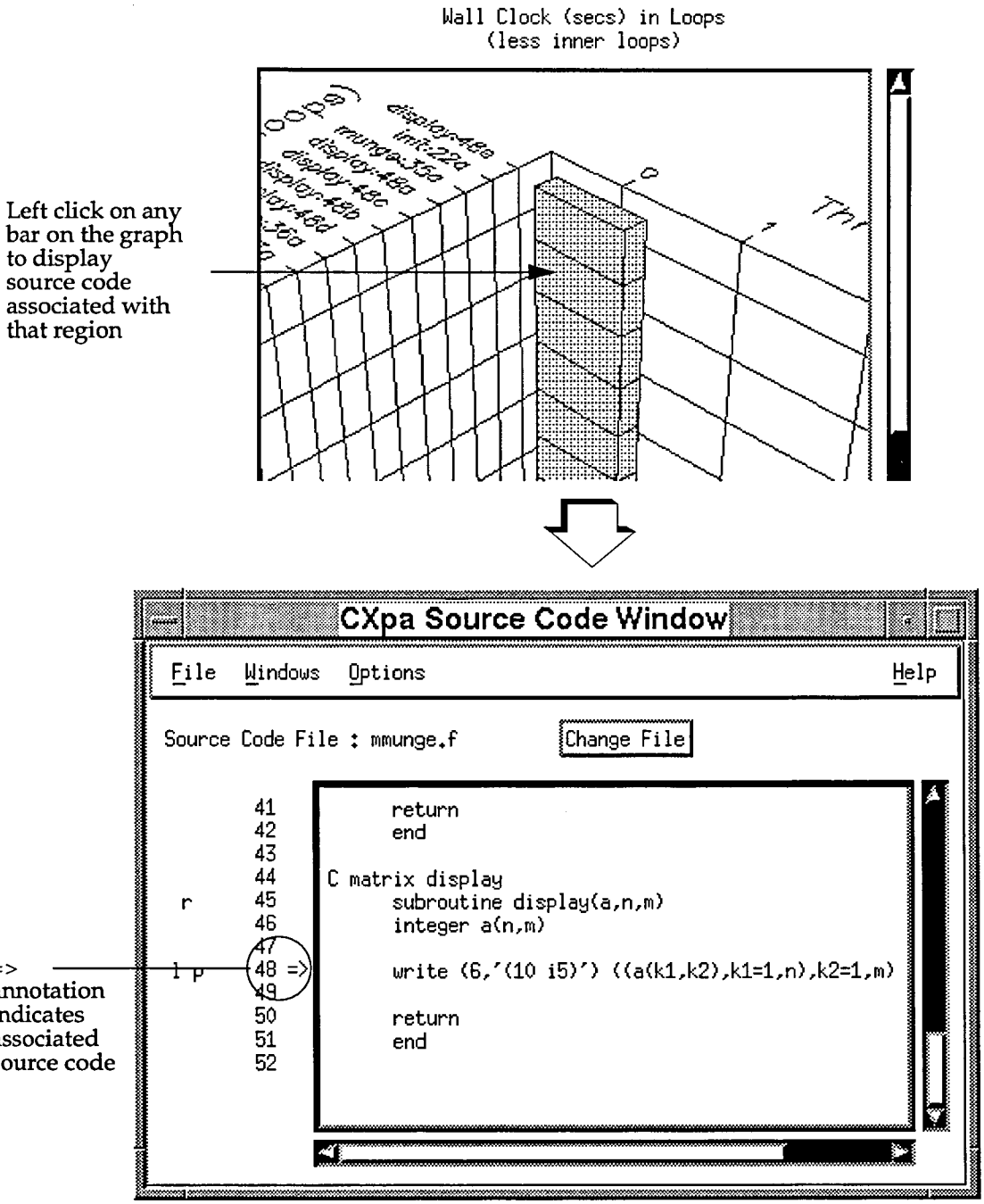
7. Display profile results by selecting one of the options from the button at the bottom-right corner or from the Windows menu:
   - **2D Profile**—Displays a 2D graph of profiling data accumulated across all threads of the process. You can dynamically select regions and metrics to graph.
   - **3D Profile**—Displays a 3D graph of profiling data for individual threads. You can dynamically select regions and metrics to graph.
   - **Report**—Displays an Analysis Report window that contains textual performance reports.

   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.

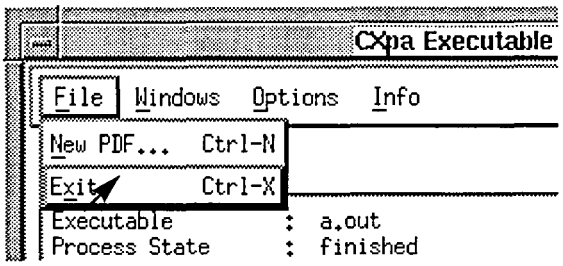


The above example shows a 3D graph of the ratio of data cache miss latency to CPU time (expressed as a percentage) for serial loop regions in a program running on an SPP 1200 system.

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.


Wall Clock (secs) in Loops
(less inner loops)
Left click on any bar on the graph to display source code associated with that region
CXpa Source Code Window
File Windows Options Help
Source Code File : mmunge.f
Change File
41 return
42 end
43
44 C matrix display
r 45 subroutine display(a,n,m)
46 integer a(n,m)
47
l p 48 => write (6,'(10 i5)') ((a(k1,k2),k1=1,n),k2=1,m)
49
50 return
51 end
52
=> annotation indicates associated source code

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 one of the following options:

   `-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)
   ```

   CXpa is a licensed product. If you do not acquire a license after starting CXpa, contact the system administrator at your site for more information.

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

   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.

   For example:

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

   The previous command selects all routines in your program for profiling.

NOTE: You will obtain the most accurate profiling results if you run the program in a standalone environment (that is, on a "quiet" or dedicated system).

4. Choose the type of metrics that you want to collect with the `collect` command.

   There is no default setting for metric collection in line mode. Unless you specify the type of metric(s) you want to collect with the `collect` command, CXpa will not collect any metrics. You can select one or more of the following options:

   - `cpu`—Collects CPU time.
   - `wall_clock`—Collects wall clock time.
   - `events`—Collects memory access events and latency time. If you enable event collection with this parameter, you must then use the `set events` command to specify the type of events collected.
   - `counts`—By default, iteration/execution counts are always collected. You do not need to specify this parameter unless you want CXpa to collect only iteration/execution counts. To do this, specify the `counts` parameter only (for example, `collect counts`).

   For example:

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

   You should collect CPU time and wall clock time for all routines the first time you profile a program using CXpa.

5. If you enabled event collection with the `events` parameter of 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.

The number and type of events you can collect are architecture-specific. Refer to the online help topic or reference page for the `set events` command for more information.

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

The previous 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 data cache miss. The `local_misses` parameter is specific to SPP 1000 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, do not pause your program during profiling. For best results, run the program to completion and then analyze the results.

   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
   ```

   When you enter the `analyze` command without specifying any parameters, CXpa generates and displays all available performance reports. To display reports for a specific region type, 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**
Execute a shell command	**!**<*command*>
Forward character	**CTRL-f**
Forward word	**ESC-f**
Kill to end of line	**CTRL-k**
Lower case word	**ESC-l**
Next command	**CTRL-n**
Previous command	**CTRL-p**
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`
`help`
`set events`
`collect`
`select`

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 1000 system can be viewed (in reports or graphs) on an SPP 1200 system, 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.

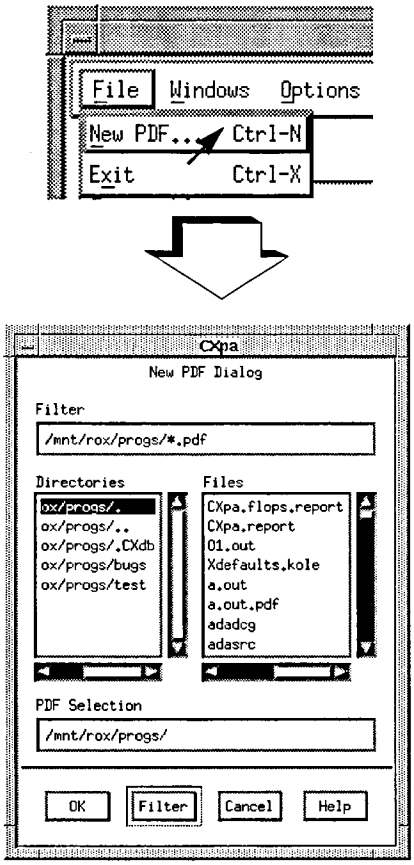
NOTE: If you invoke CXpa with an executable, you can only analyze a PDF created during the current CXpa session. To analyze a PDF created with a different executable or multiple PDF files, invoke CXpa without specifying an executable or with the name of a PDF file.

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

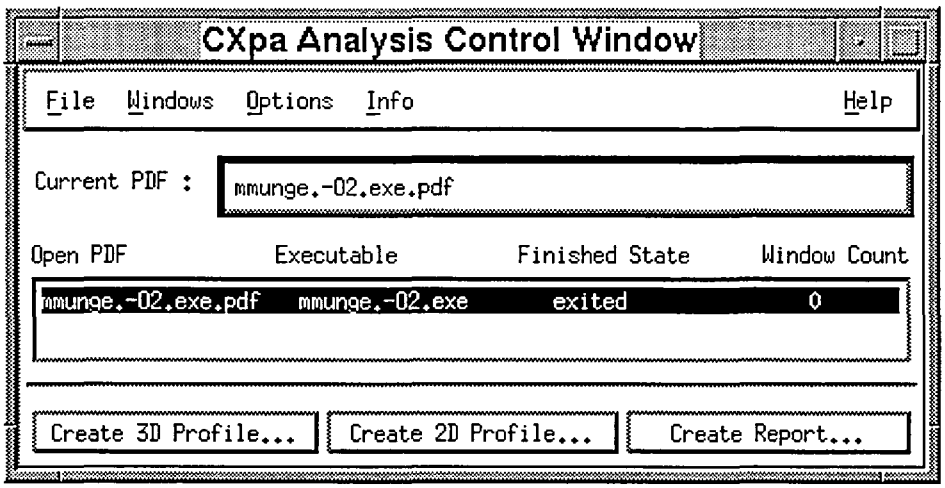
In both X window and line modes, you can view only performance data in PDFs that were created in previous CXpa sessions when you invoke CXpa with the name of a PDF file only (without specifying an executable).

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 profile or report windows when you invoke CXpa with the name of a PDF file (without specifying an executable). The resource is `Cxpa*defaultWindow`, and it can accept the following values: `All`, `None`, `2DProfile`, `3DProfile`, `Report`, or `Source`. The default is `None`.

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 routine 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 cmd_file -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.

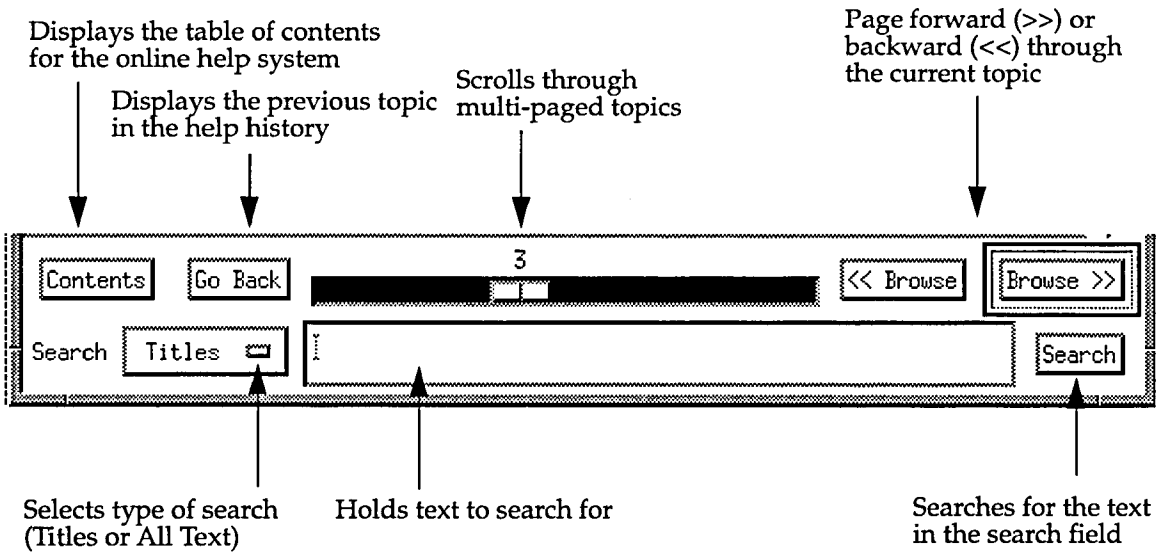
This section explains how to use the following features of the Help window:

- 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

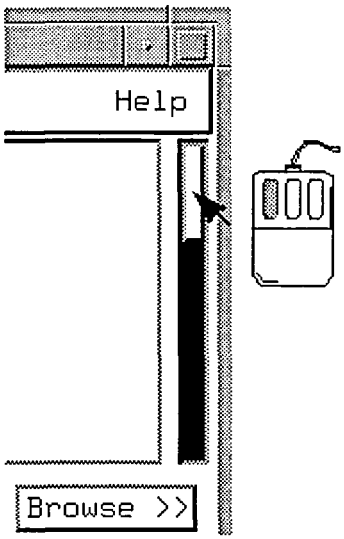
The buttons in this window are for navigating the online help system, as shown in the following figure.



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.)



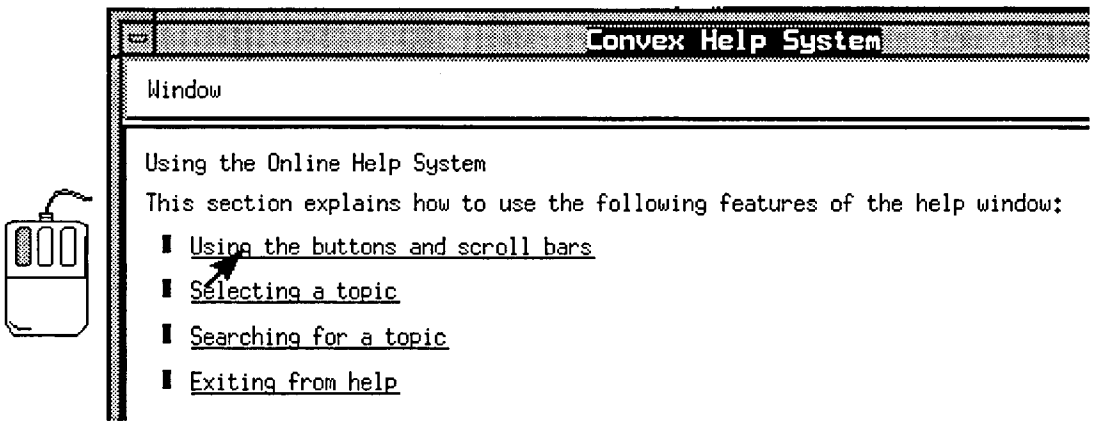
Using the menus

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

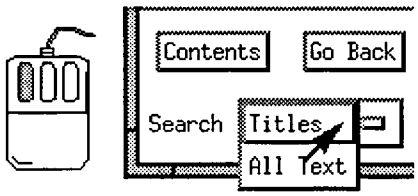
In this Help window, any underlined text is a help topic name. Click on the underlined text with the left mouse button to view the help page for that topic.



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

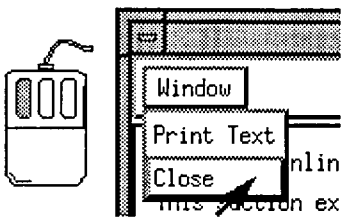
4. Click on the Search button, or press **RETURN**.

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 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`	Routines (if compiled `-cxpa` or `-cxpar`); loops (if compiled `-cxpa`); basic blocks (if compiled `-cxpab`)
`-O3`	Routines (if compiled `-cxpa` or `-cxpar`); parallel regions and loops (if compiled `-cxpa`); 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 (parallel 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, serial loops, and parallel loop 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:

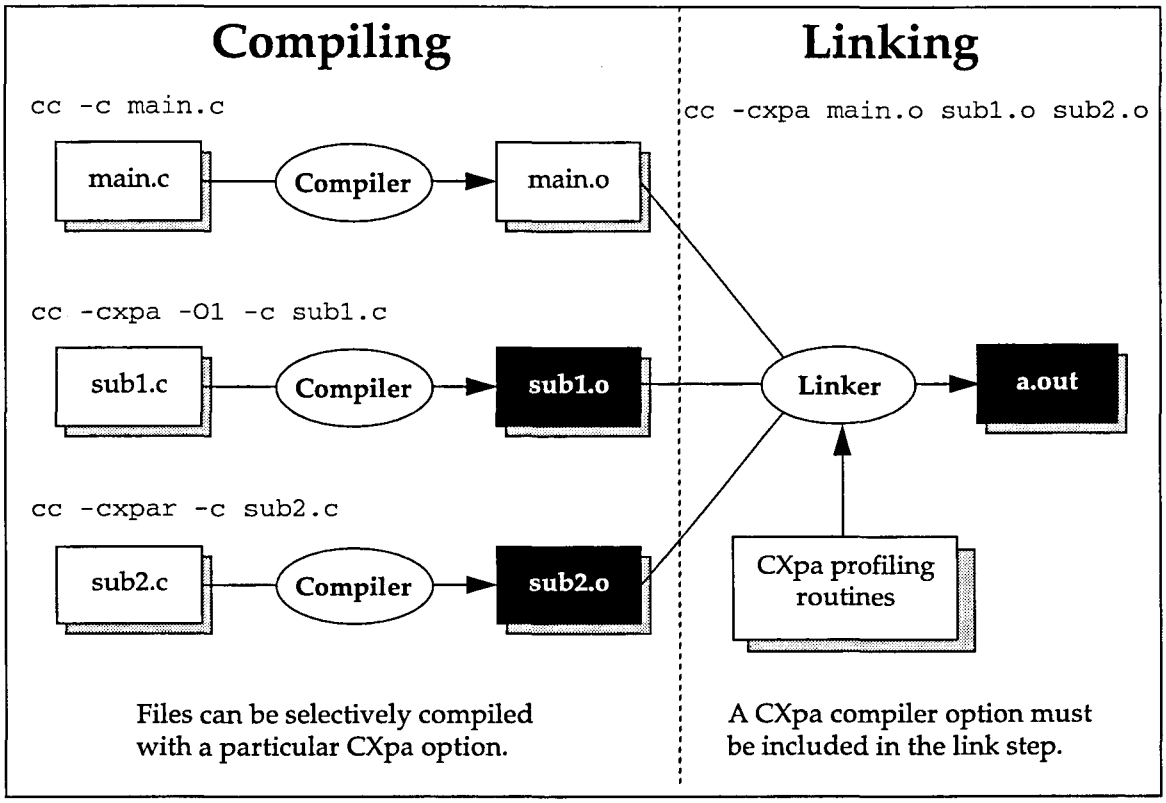
```
cc -cxpa -O1 main.c
```

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
- Compiling routines with both `-cxpa` and `-cxpab`
- Including CXpa compiler options (`-cxpamon`, `-cxpalib`, `-cxpa`, `cxpar`, and `-cxpab`) within FORTRAN `OPTIONS` statements.

Related Topics Advanced compiling

Related Commands `cxpa`

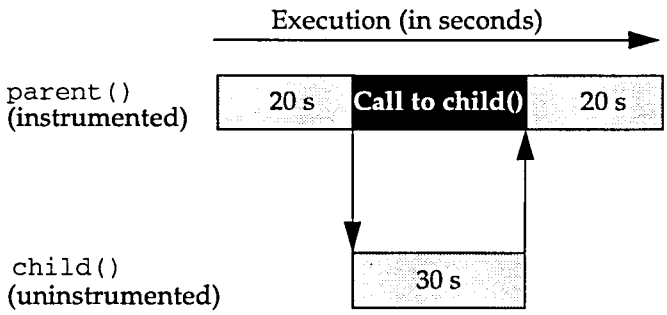
Advanced compiling

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
- 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 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).

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 Profile 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.

Use the following guidelines for selecting regions to profile:

NOTE: Selecting all source code regions in all routines for profiling will result in increased profiling time, and significant profiling intrusion may be incurred.

- The first time you profile your program under CXpa, select all routines in your program for profiling at the routine region level (this is the default). 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 that consume the most wall clock time or CPU time, then select only those routines for profiling at the loop or parallel region level and rerun your program under CXpa. With fewer source code regions selected, less time is spent in the timing routines CXpa uses to collect performance data.
- 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)

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.

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."

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).

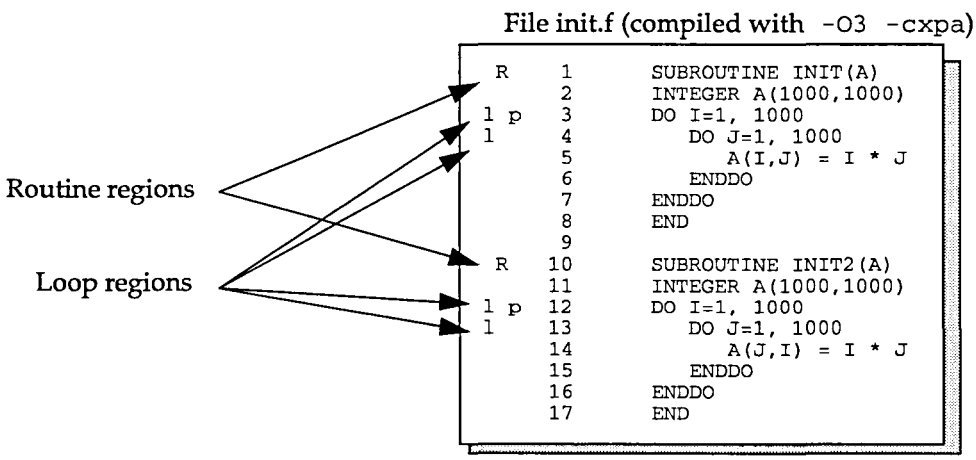
Source code region type	Selected	Deselected
Routine	R	r
Loop	L	l
Parallel loops	P	p
Basic block	B	b

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.

Related Topics

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

Related Windows

Executable Manager window
Profile Selection dialog

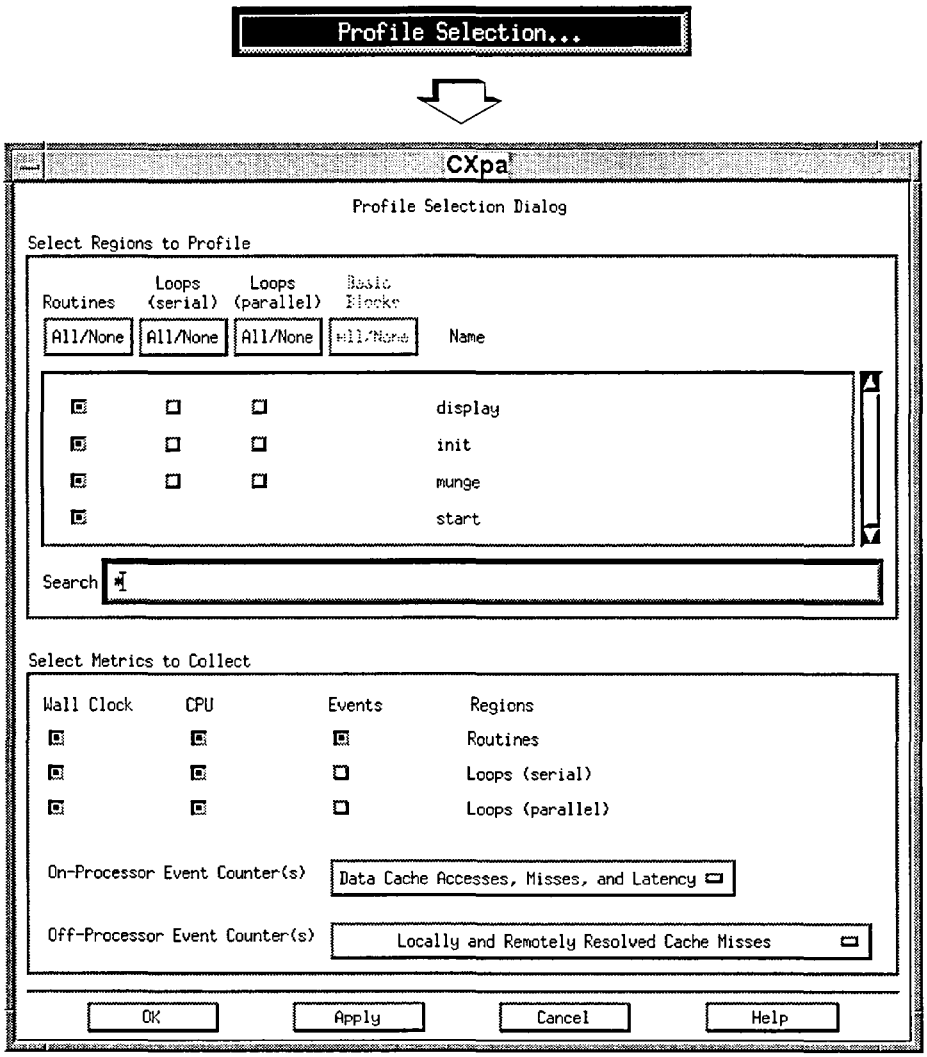
Related Commands

`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:

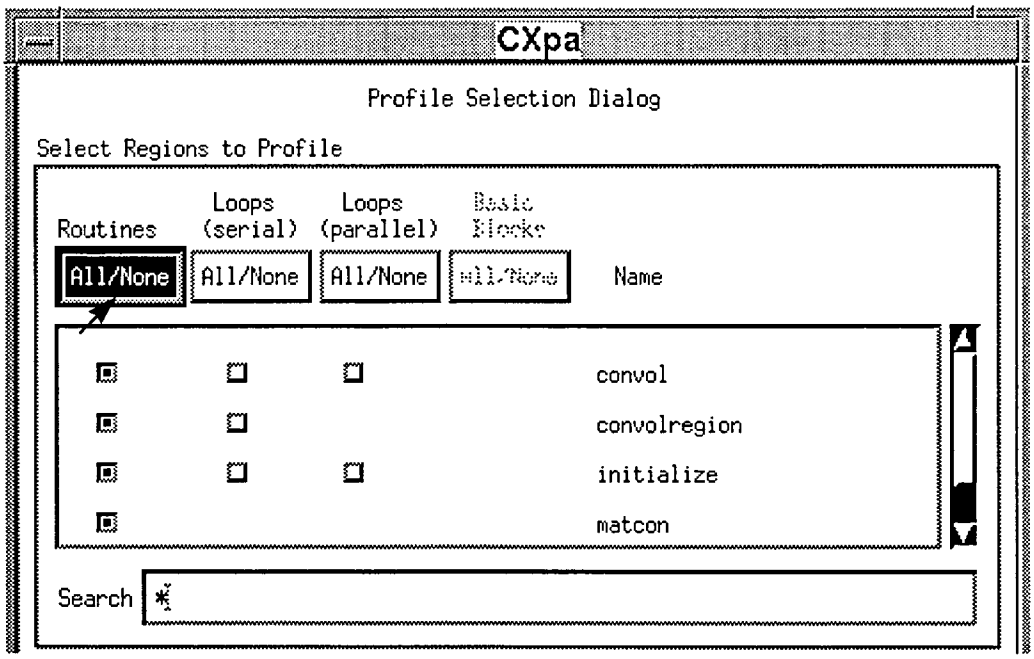
1. Make sure that all routines are selected (by default, all routines are selected when you first bring up the Profile Selection dialog). You can toggle the Routines All/None button for Routine regions to make sure that all routines are selected.
2. Click the All/None toggle buttons for the types of 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.

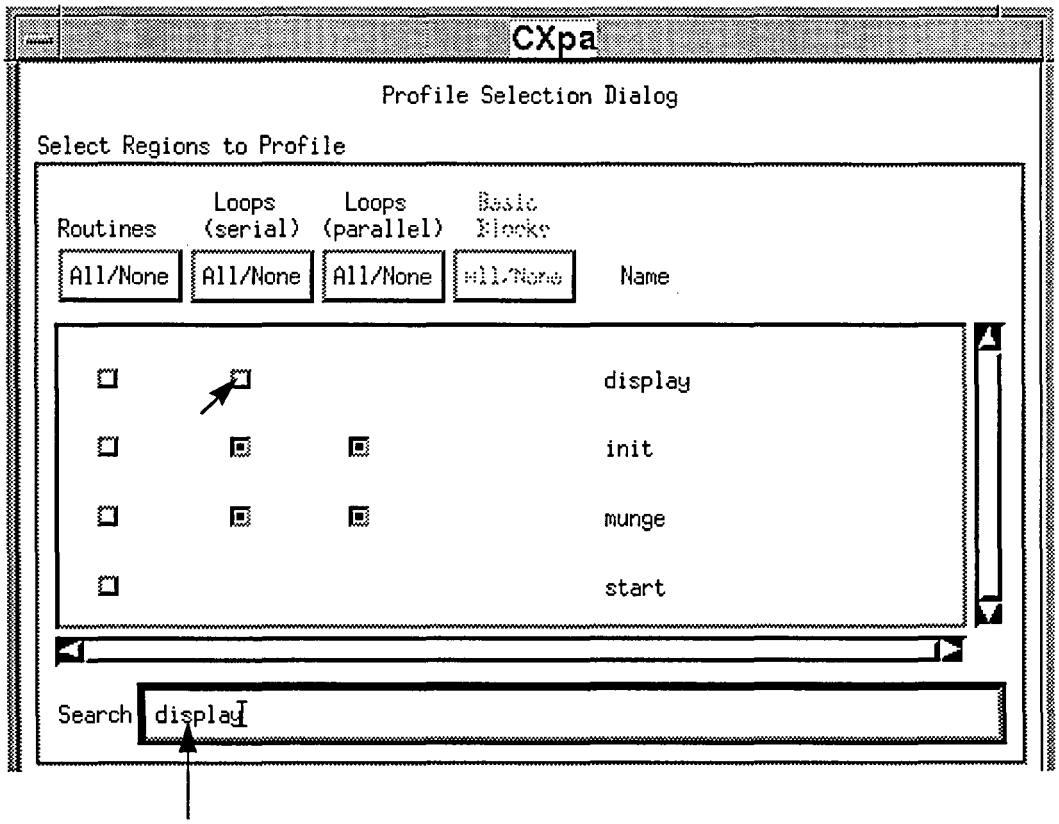
Selecting types of regions to profile in specific routines

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. Select all routine regions first to find out which routines contain performance bottlenecks. Then, select specific region types in these routines for profiling.

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.

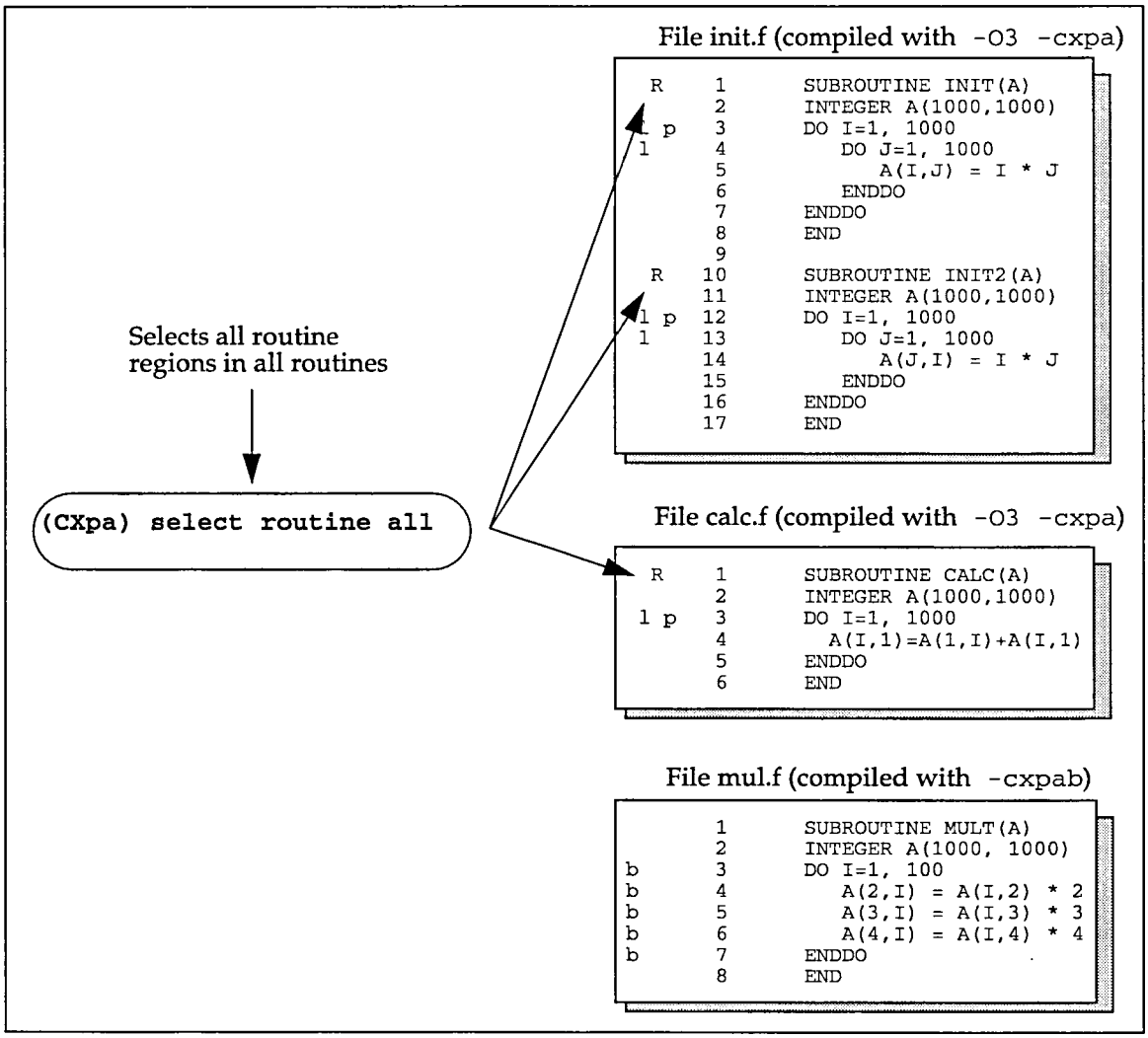
In the examples, uppercase annotations indicate selected source code regions, while lowercase annotations indicate deselected source code regions. In line mode, annotated source code is displayed using the `list` or `list selectable` commands.

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

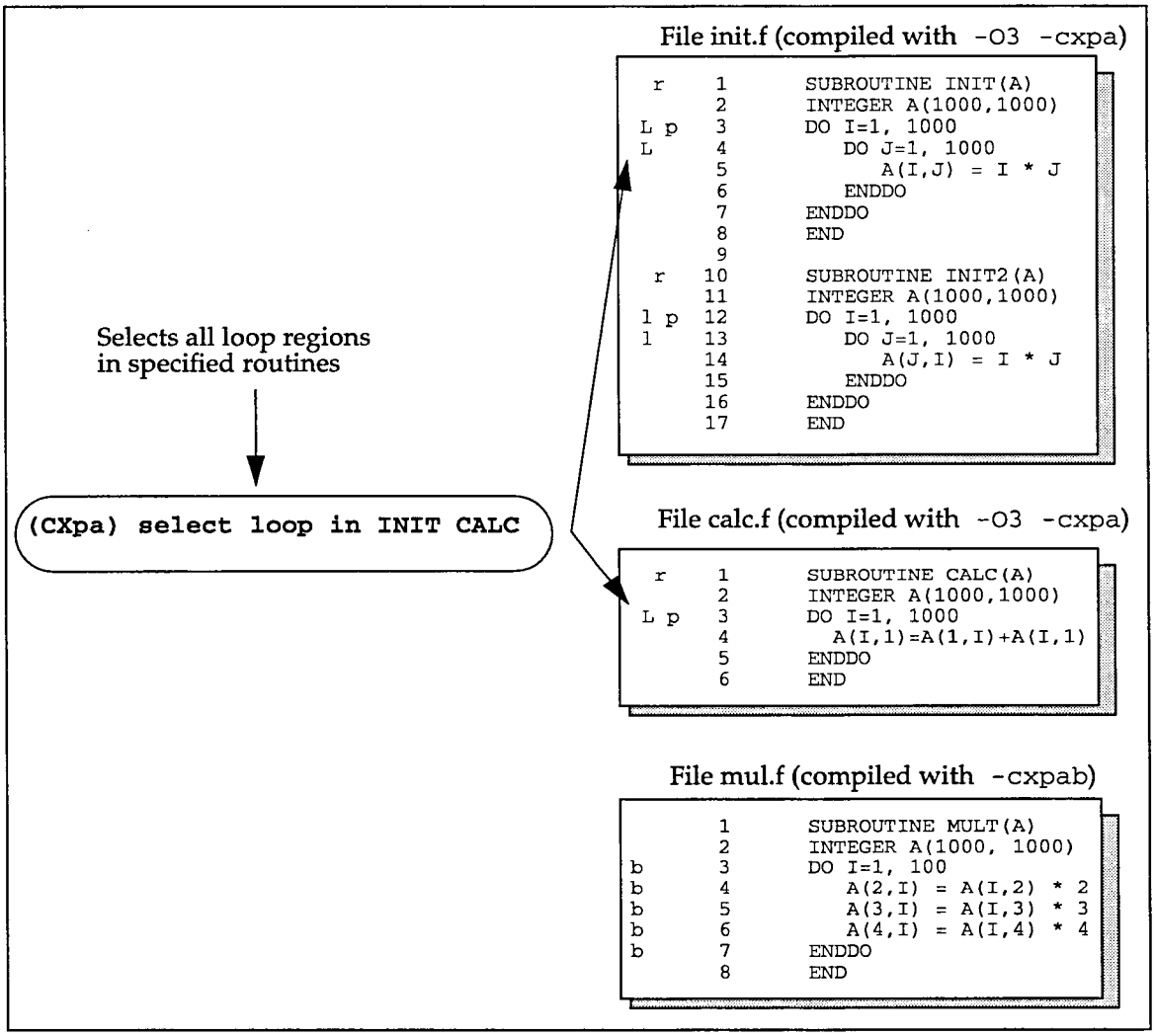
You can select or deselect one type of source code region in specific routines using one or more of the following commands:

- `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

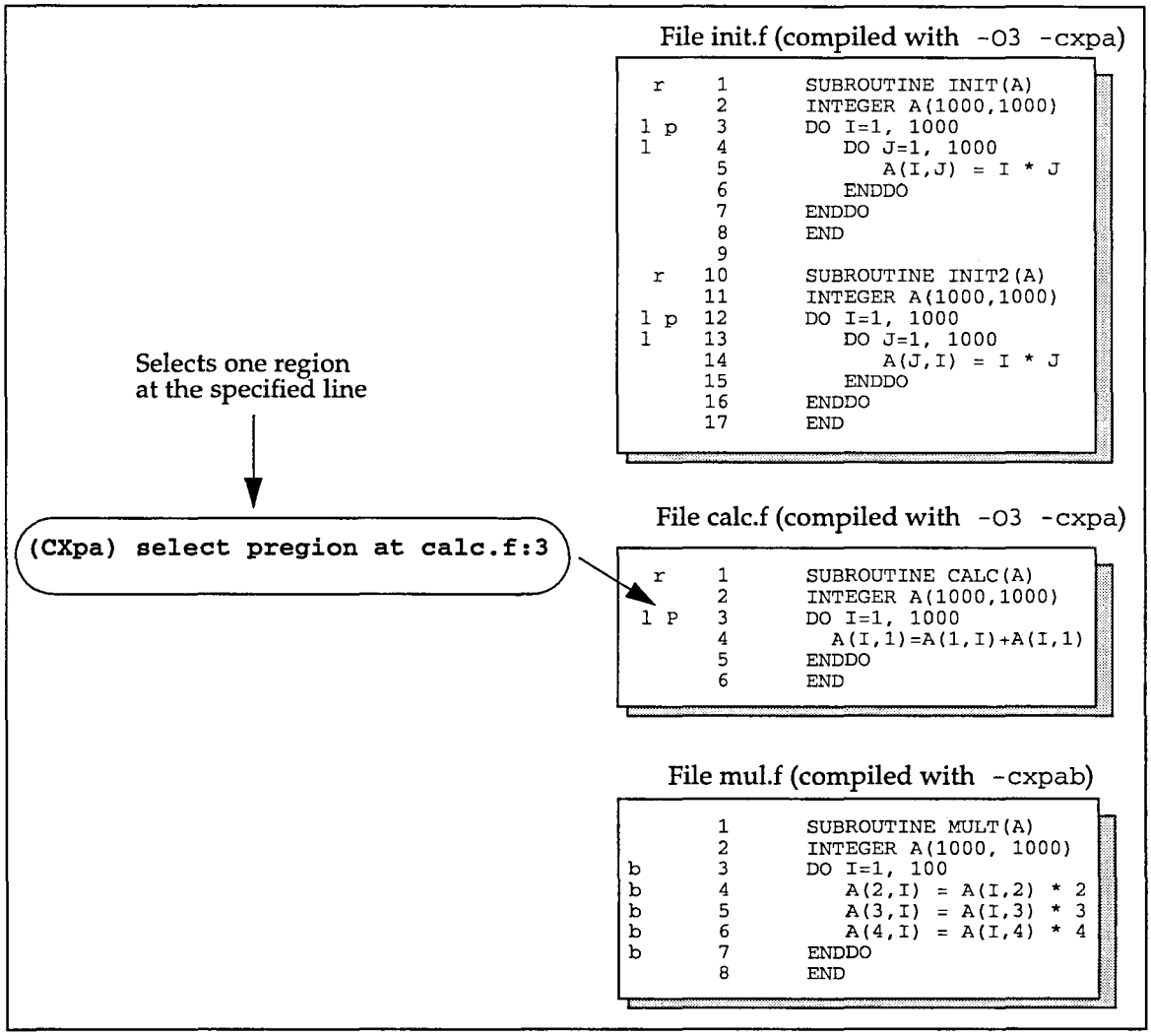
If you do not want to select or deselect all regions at a specific line, you can select or deselect a single region type at a line. This makes it possible to profile a parallel region without profiling the loop around it.

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.

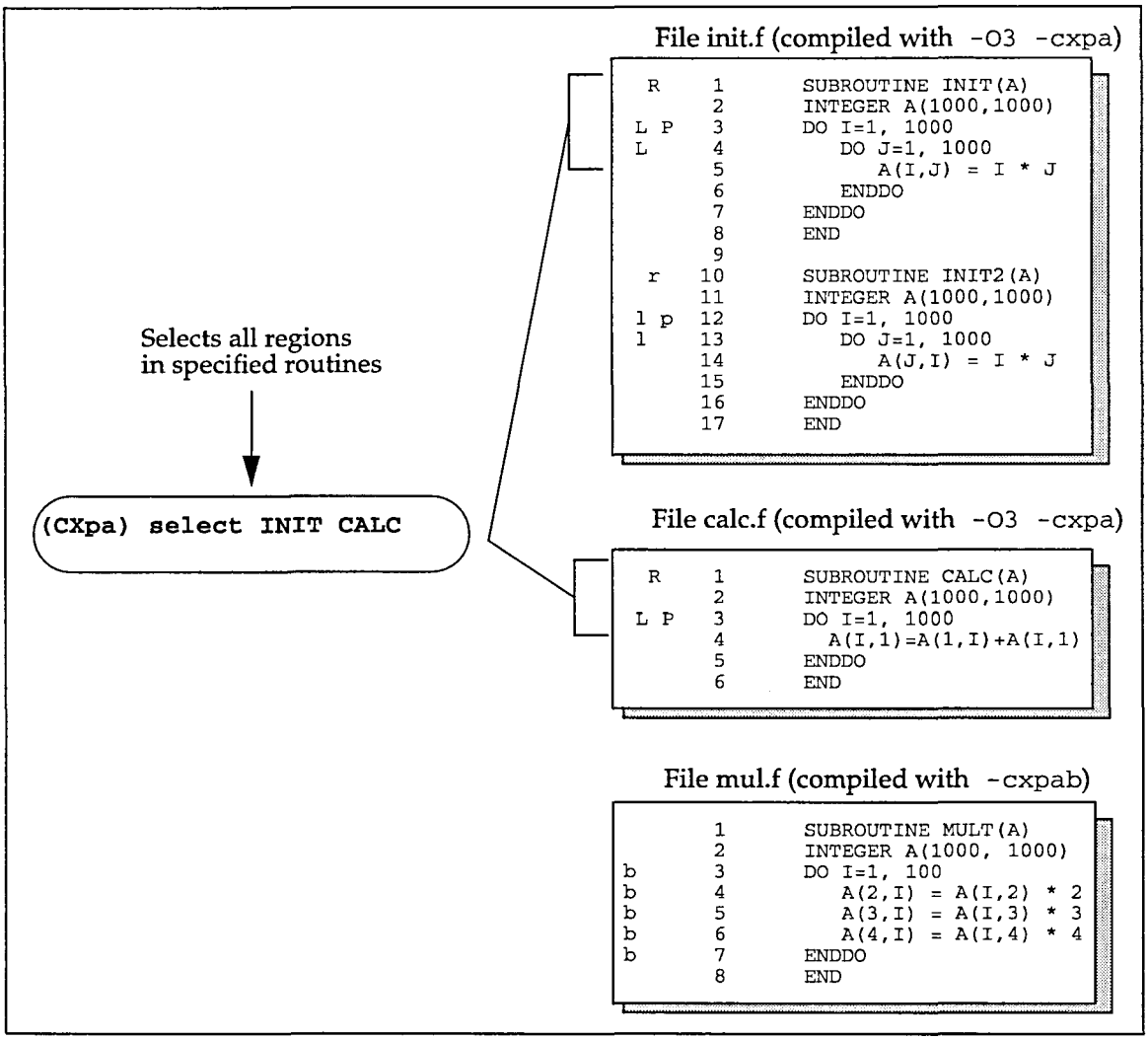
The `select pregion at` command in the following figure selects the parallel region at line 3 in the `calc.f` source file. The loop region on the same line remains deselected.



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.

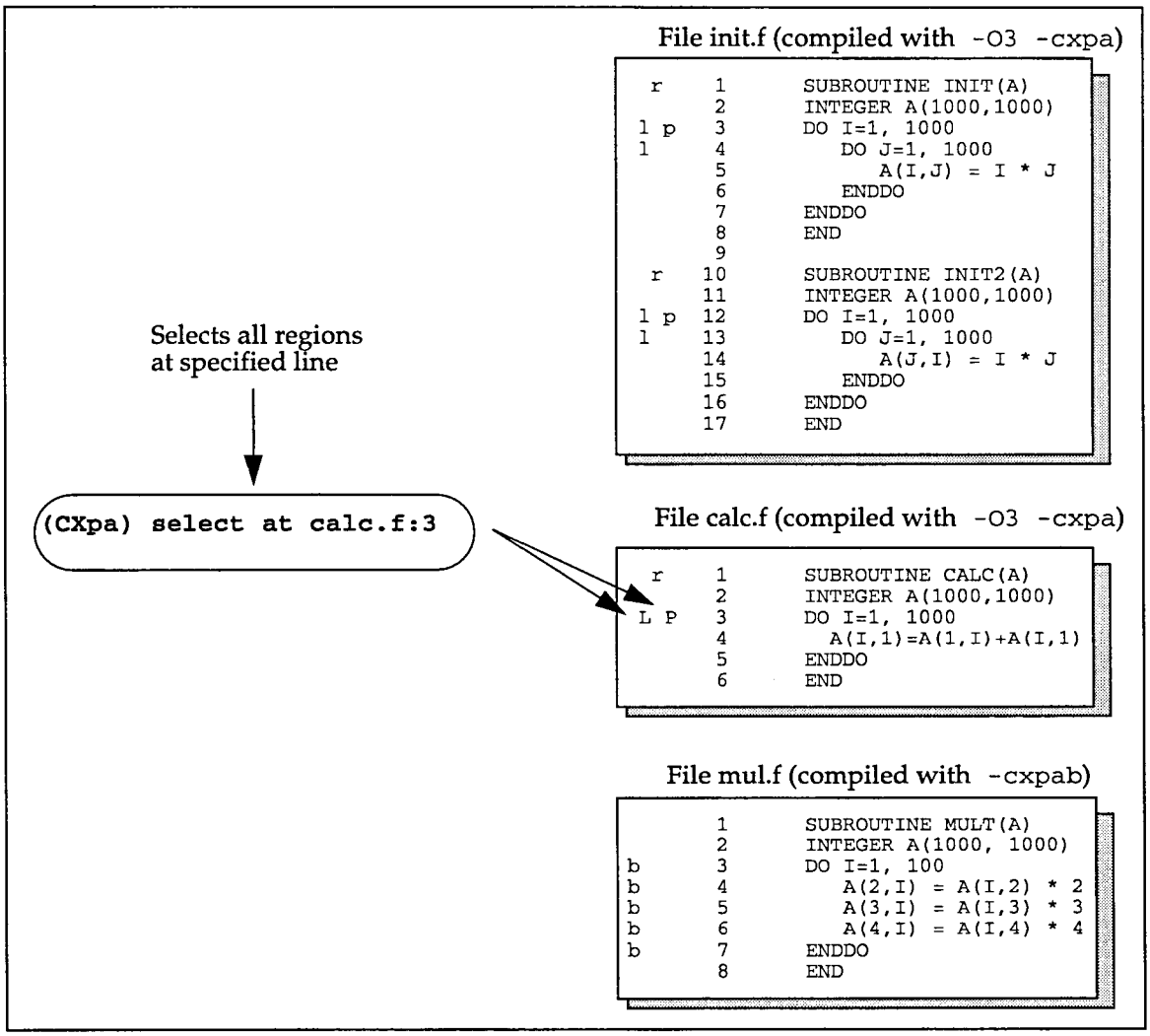
The following figure illustrates how to select all regions in specific routines.



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.

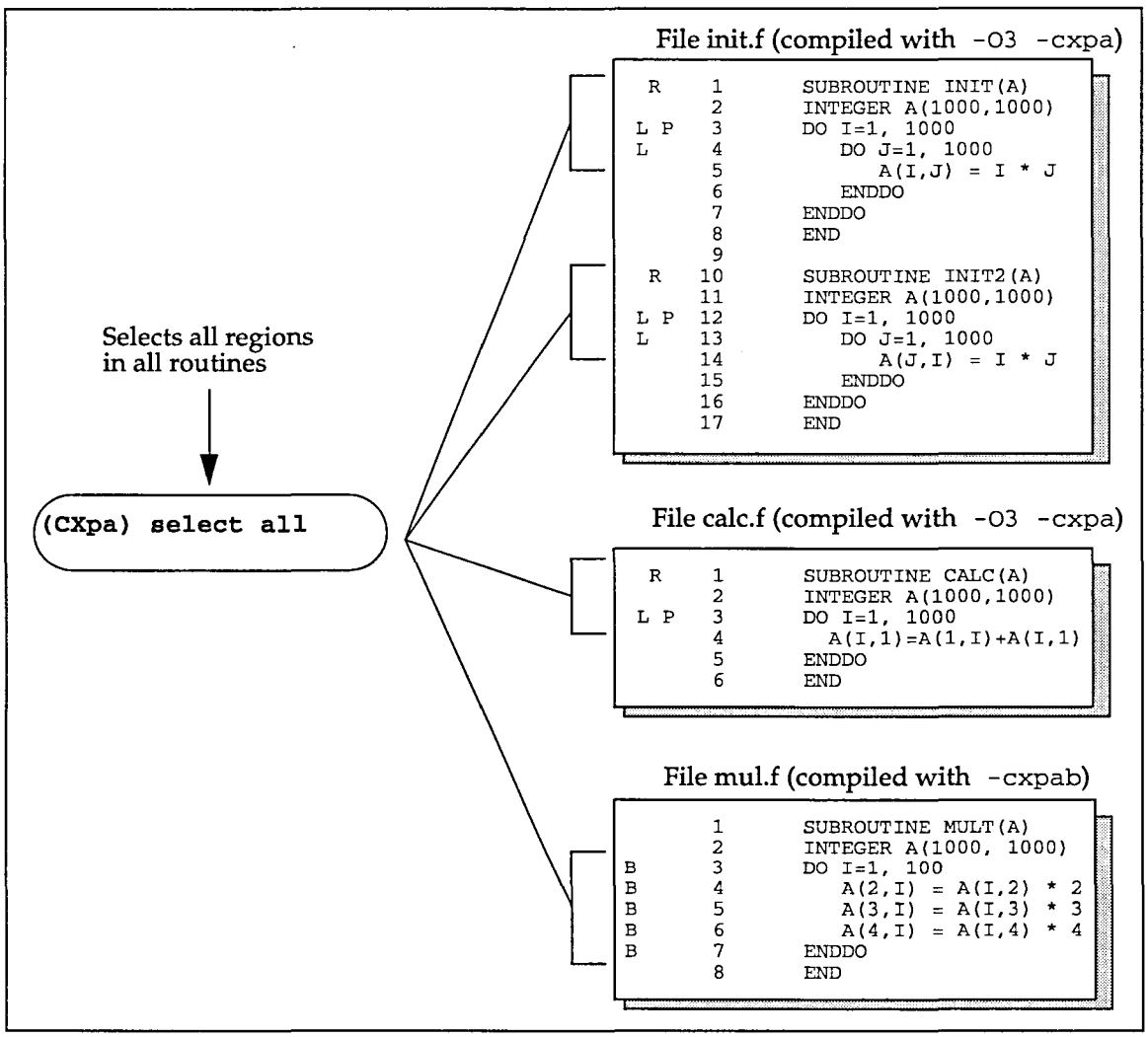
To select or deselect all regions at a line in a specific routine, use the `select at` or `deselect at` commands, followed by the file name and line number you want to enable. The file name and line number are separated by a colon. The example below shows how to select all regions at a specific line.



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 the types of performance metrics you want to collect for these regions. Collecting and comparing different metrics can help you discover performance bottlenecks such as:

- Routines and loops that consume the most wall clock or CPU time
- Loops that generate excessive cache misses
- Regions of code that spend a significant amount of their CPU time waiting on memory
- Lack of any effective parallelism in a particular loop or routine
- Memory bank-contention and/or cache thrashing among threads in parallel regions
- Uneven distribution of work across threads in parallel regions

The first time you profile your program under CXpa, collect CPU time and wall clock time for all routines in your program. This will enable you to determine which routines take the longest to execute and/or consume the most CPU resources.

The topics discussed in this section include:

- Metrics available on all architectures
- SPP Series event metrics:
  - SPP Series event terminology
  - SPP 1000 off-processor events
  - SPP 1200 on-processor events
- Exclusive routine timing

Metrics available on all architectures

The following metrics are available on all architectures:

- **CPU Time**—Time the CPUs work on the process.
- **Wall Clock Time**—Time to solution, including process idle time.

- **CPU/Wall clock time**—The ratio of CPU time to wall clock time. This metric is computed during analysis if CPU and wall clock time metrics are collected.
  - For serial regions, if the CPU/Wall clock ratio is high, the region is compute-bound.
  - For parallel regions, this metric measures the *concurrency factor*, or the speed-up achieved through parallelization. Values that approach n, where n is the number of processors, indicate good parallel concurrency.
  - For both parallel and serial regions, if the CPU/Wall clock ratio is low, this could indicate a performance bottleneck caused by one or more of the following:

    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 regions to find out if the bottleneck is due to memory accesses.

- **Execution counts**—Number of times a routine or basic block was executed or, for loops, the number of loop invocations.
- **Iterations**—Minimum, maximum, and average number of iterations for serial loops.
- **Chunks**—For parallel loops, this indicates the number of chunks (or packets of loop iterations) that are assigned to execute on a particular thread.

  Chunk counts can be used to examine relative load balancing across threads in parallel loops. CXpa does not currently report the number of iterations in a chunk (chunk size).

Event metrics

For SPP Series computers, event counts and latency metrics can be collected for memory access events. Memory access events occur when required data or instructions must be retrieved from memory rather than from the processor cache.

Monitoring events such as cache miss counts and latency for "hot" routines on subsequent runs of your program under CXpa can be useful for identifying second-order effects that contribute to poor performance such as ineffective cache utilization or contended access to data among processors on the same node.

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.

The type of event metrics that can be collected vary according to machine architecture.

SPP Series event metrics terminology

The following definitions will help you understand and interpret Exemplar SPP Series events.

Average clock cycles per instruction

This metric, available on SPP 1200 systems only, measures the efficiency of instruction scheduling for the region being profiled. It is computed during analysis if instruction counts and clock cycles are collected. On SPP 1200 systems, the maximum number of instructions that can be executed in a single clock cycle is 2. This means that the theoretical peak value for the average clock cycles metric on this architecture is 0.5.

If the value for average clock cycles is high, and it is associated with a frequently called routine that performs only a small amount of work (for example, a routine that simply assigns a new value to a variable and returns), inlining the routine could improve the instruction scheduling by eliminating the routine call overhead.

Code sections that contain many patterns of consecutive loads and stores followed by immediate use of requested operands can also result in high average clock cycles.

In some cases, the instruction scheduling can be improved by compiling the routine at a higher optimization level or, if programming at the assembly level, changing the order of instructions. Refer to the *Hewlett-Packard PA-RISC 1.1 Architecture and Instruction Set Reference Manual* (Manual Part No. 09740-90039) for information on how to achieve optimal instruction scheduling.

Average MIPS rate

The average MIPS (millions of instructions per second) rate is calculated during analysis if instruction counts, clock cycles, and wall clock time are collected. This metric is only available on SPP 1200 systems.

The formula CXpa uses to calculate the average MIPS rate is as follows:

$$Average\_MIPS\_rate = \frac{Number\_of\_instructions\_completed}{Wall\_clock\_time\_in\_sec}$$

The theoretical peak MIPS rate for Hewlett-Packard PA-RISC 7100/7200 processors with a 10-ns clock cycle at a clock rate of 100 MHz is 200 MIPS. There is a direct correlation between average clock cycles and average MIPS rates; decreasing average clock cycles will result in increased average MIPS rates.

CPU Agent chip

The CPU Agent chip is the gate array on Convex SPP systems that provides a high-speed interface between the pairs of PA-RISC processors in a functional block on a hypernode and the hypernode crossbar.

Performance counters (referred to as *off-processor* counters) located on the CPU agent used in SPP 1000 systems provide information about locally and remotely resolved data cache misses and latency.

CTI (Coherent Toroidal Interconnect) rings

The ring interconnect that connects all the hypernodes of a multihypernode Convex SPP Series system together in a ring topology. While the CTI ring is derived from the IEEE SCI (Scalable Coherent Interface) standard, complete compatibility is sacrified to provide lower latencies.

Data cache miss

A data cache miss occurs if data to be loaded does not reside in the processor's data cache.

Data cache hit

A data cache hit occurs if data to be loaded resides in the processor's data cache.

Data cache hit rate

This metric, available on SPP 1200 systems only, measures the percentage of total data cache accesses that were data cache hits. If the data cache hit rate is low, this indicates cache thrashing.

The data cache hit rate is calculated as follows:

$$data\_cache\_hit\_rate = \frac{total\_data\_cache\_accesses - data\_cache\_misses}{data\_cache\_accesses} \times 100$$

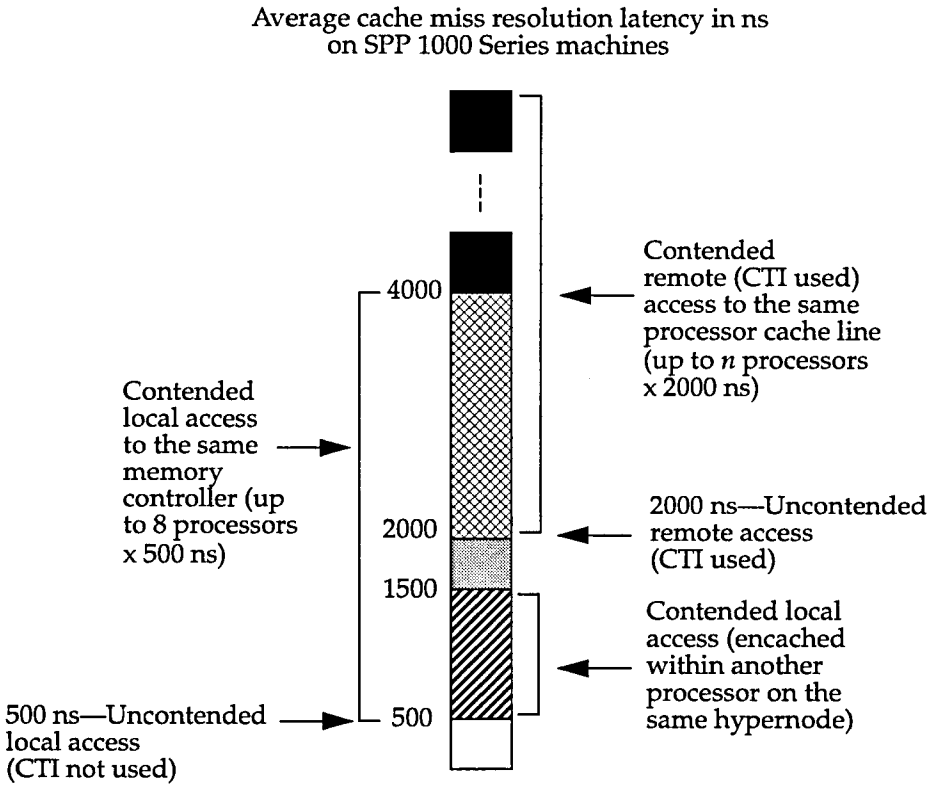
Event counts

The number of times an event, such as a data or instruction cache miss, occurred.

Event latency/counts

Average latency time per event for a region. For example, if total data cache miss counts are collected, then the event latency/counts metric represents the average amount of time it took to resolve a data cache miss.

Average cache miss resolution latency metrics can be very useful when examining regions with large numbers of cache misses and/or latency, and can be characterized on SPP 1000 Series machines as shown in the following figure.



You can use this information to interpret 2D and 3D profile graphs of average cache miss resolution.

Event latency/CPU

Ratio of event latency to CPU time for a region, expressed as a percentage. When this percentage is high, it means that the program spent the majority of its CPU time in the region reaching for data or instructions that were not encached rather than computing with encached data or instructions.

Hypernode

In Exemplar SPP Series systems, a set of up to eight processors and physical memory organized as a symmetric multiprocessor (SMP) running a single image of the operating system microkernel. An SPP system consists of one or more hypernodes, with a high speed CTI ring connecting the hypernodes.

Instruction cache miss

An instruction cache miss occurs if data to be loaded does not reside in the processor's instruction cache.

Latency

The amount of time spent accessing memory to locate data or instructions not found in the processor's data or instruction cache.

Locally resolved cache misses

Cache misses that are resolved by using the hypernode crossbar to access memory found on the same hypernode as the processor; the CTI rings are not used.

Read miss

An instruction or data cache miss caused by a load.

On-processor and off-processor events

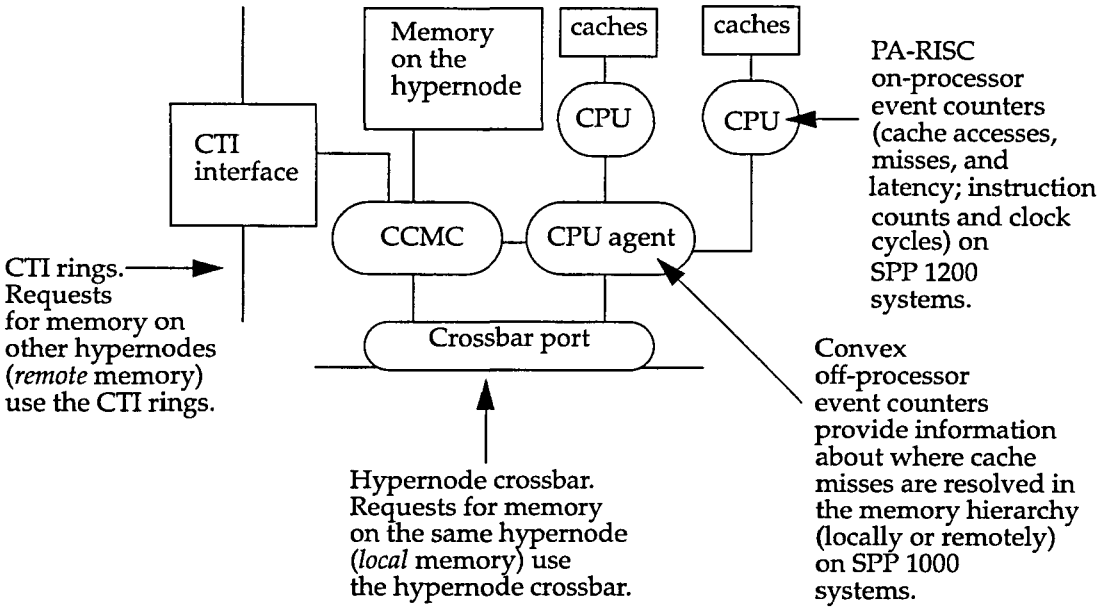
On-processor events are monitored by event counters located on the HP PA-RISC processors used in SPP 1200 systems. These event counters monitor events from the processor's point of view only.

Off-processor events are monitored by event counters on the SPP 1000 Convex CPU agent chip. These event counters can be configured to monitor the number of data cache miss events that are resolved locally and/or remotely (remote miss events imply use of the CTI rings; local miss events do not use the CTI rings) and whether those events occurred due to reads (loads) or writes (stores).

The following diagram shows the location of event counters on Exemplar SPP Series systems. Refer to *Exemplar SPP 1000/1200 Architecture* (DHW-014) or the *Exemplar Programming Guide* (DSW-067) for more information.

NOTE: SPP 1200 systems use HP PA-RISC 7200 processors, which have on-processor performance counters for collecting total data cache accesses, misses, and latency; instruction cache misses and latency; and instruction counts and clock cycles.

SPP 1000 systems use HP PA-RISC 7100 processors, which have no on-processor event counters. For SPP 1000 systems, a latency timer was added as an off-processor event counter to provide this information.



Remotely resolved cache misses

Cache misses that were resolved by using the CTI rings to access memory on another hypernode. Accessing memory on other hypernodes using the CTI rings takes significantly longer than accessing memory on the same hypernode via the hypernode crossbar.

Write miss

An instruction or data cache miss caused by a store or by a load and clear.

SPP 1000 off-processor events

On SPP 1000 systems, the following off-processor events can be collected. In addition, you can specify whether these events are collected during read accesses (loads), write accesses (stores), or both. The default is both.

NOTE: Collecting remote miss events on single-node SPP 1000 systems yields zeros.

Locally resolved data cache misses and latency

> Configures off-processor event counters to collect the number of times that data not found in the processor cache was found in local memory (memory allocated to that processor's hypernode). Data cache miss latency is also collected.

Remotely resolved data cache misses and latency

Configures off-processor event to collect the number of times that data not found in the processor cache was found in remote memory (memory allocated to another hypernode). Data cache miss latency is also collected.

Locally and remotely resolved data cache misses and latency

Collects the number of local and remote memory misses and latency, combined.

SPP 1200 on-processor events

On SPP 1200 systems, the following on-processor events can be collected:

Data cache access counts, miss counts, and latency

Configures on-processor event counters to collect the total number of data cache access, data cache misses, and cache miss latency. Average latency and data cache hit rates are computed during analysis.

Instruction cache miss counts and latency

Configures on-processor event counters to collect the total number of instruction cache misses and latency. Average instruction cache miss latency is computed during analysis.

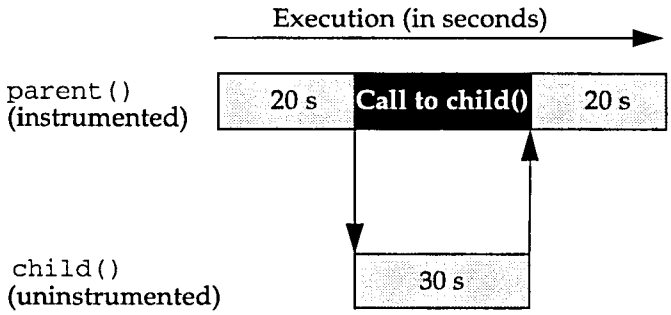
Instruction counts and clock cycles

Configures the on-processor event counters to collect the number of completed instructions and clock cycles. The average number of clock cycles per instruction is computed during analysis. The average MIPS rate is computed during analysis if wall clock time is also collected.

Exclusive routine timing

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.

Related Topics

Glossary
Loop reports
Parallel Region reports
Reports
Routine reports
Selecting metrics in line mode
Selecting metrics in X window mode

Related Windows

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

Related Commands

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

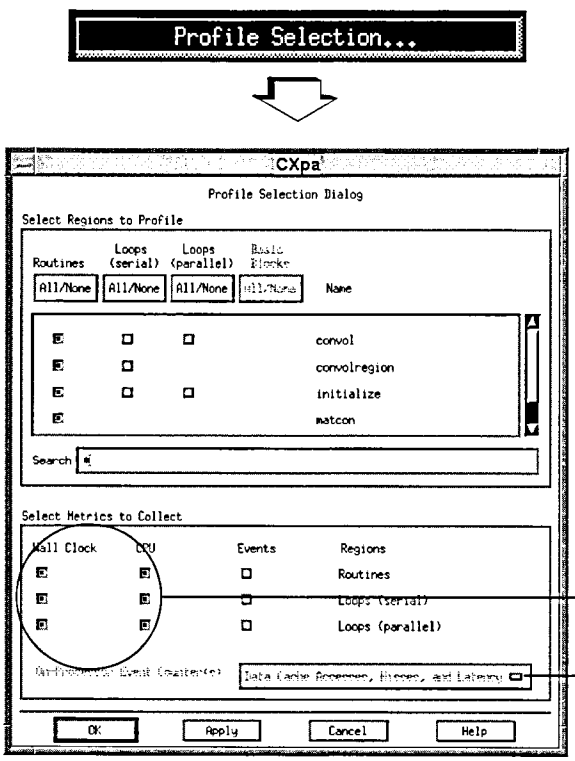
Selecting metrics in X window mode

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, wall clock 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 on each architecture. 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.

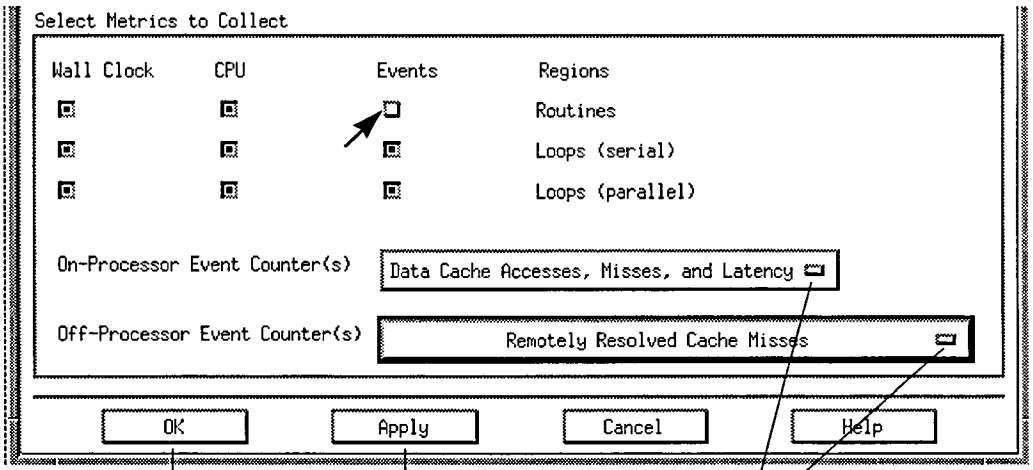


By default, CXpa collects CPU time and wall clock time for all routines.

On/Off-processor event selection menus vary according to architecture and are desensitized until event collection is enabled.

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.

Click toggle buttons to select/deselect metrics for specified regions.



Apply settings and close the dialog.

Apply settings without closing the dialog.

Event Counter(s) option menus show current selection(s) for on/off-processor events when event collection is enabled.

If you wish to collect events metrics, you must first choose Events as one of the metrics in the Select Metrics to Collect section of this dialog.

When event collection is enabled, the Event Counter options menu shows the type of event currently selected.

Refer to the next section, "Selecting events," 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 event metrics SPP Series systems.

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

Selecting events

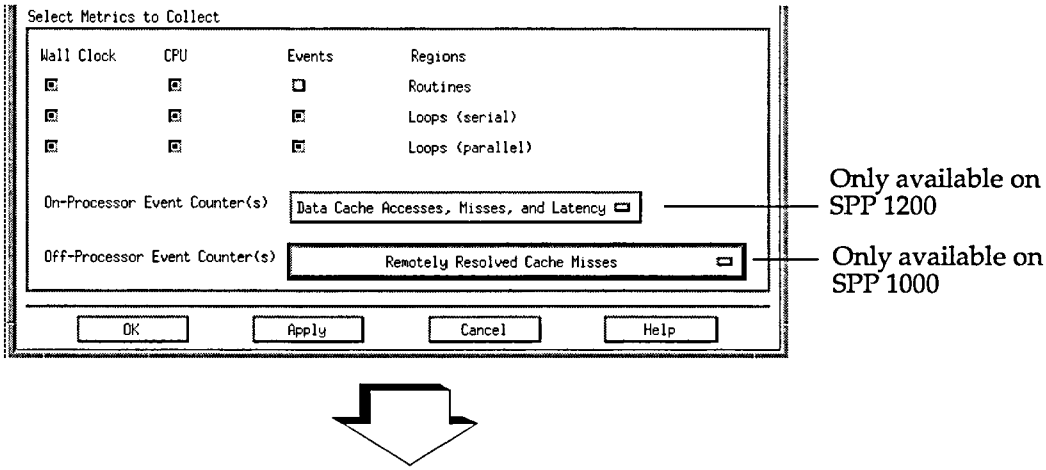
To specify the type of event you want to collect:

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.

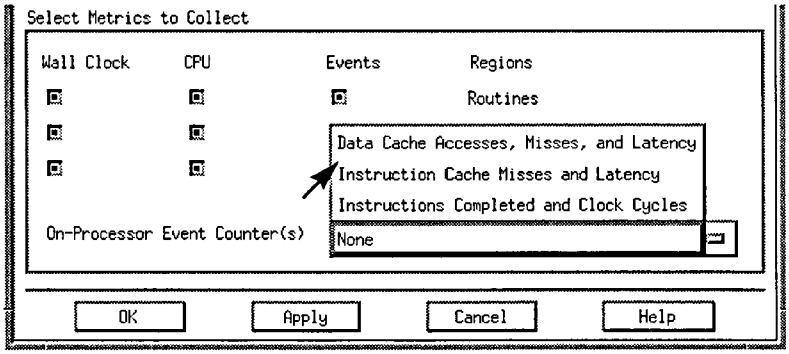
   The type and number of events that can be collected per program run differ according to machine architecture:

   - On SPP 1200 systems, you can monitor one set of on-processor events. The default on-processor event set is Data Cache Accesses, Misses, and Latency.
   - On SPP 1000 systems, you can monitor one type of off-processor event per program run. The default is Locally and Remotely Resolved Cache Misses.

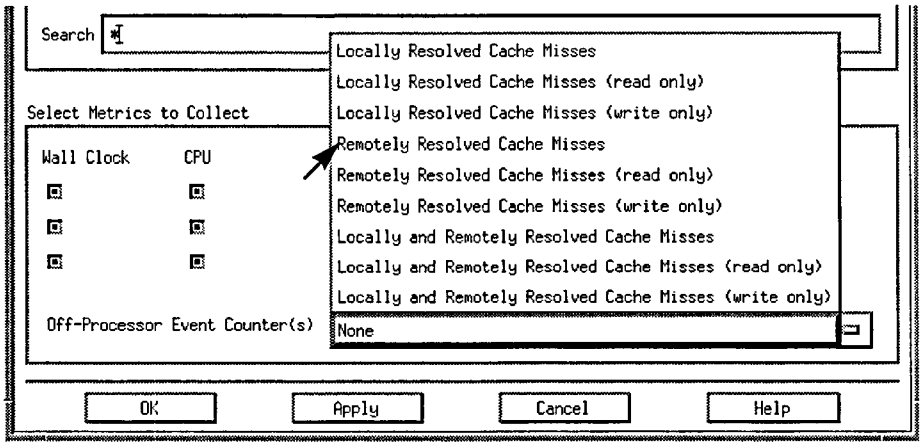
2. Click the left mouse button on an Event Counter(s) options menu to display a list of available event types. The types of events available vary between SPP 1200 and SPP 1000 systems, as shown in the following figure.



On-processor events (SPP 1200 only)



Off-processor events (SPP 1000 only)



NOTE: Because each architecture has a different set of events that can be collected, the Event Counter options menus available vary according to whether you are running on an SPP 1000 or an SPP 1200 system.

3. Holding down the left mouse button, position the mouse cursor over the type of event or events you want to select, then release the mouse button.

   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 settings and close the Profile Selection dialog or press Apply to apply the settings without closing the 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 this book for more information:

- "Introducing metrics"—List of available metrics on each architecture, metric descriptions, and event terminology.
- Reference page for the `collect` command—Complete syntax and examples.
- Reference page for the `set events` command—Complete syntax and examples.

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. Refer to the reference page for the `collect` command for more information and a list of valid parameters.

If you chose to collect events, you must use the `set events` command to specify the type of events you want to collect.

3. If you have specified event collection with the `events` parameter of 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 type and number of events you can collect differ according to machine architecture. Refer to the next section for a list of valid parameters for specifying events for SPP 1000 and SPP 1200 architectures.

   For example:

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

   The above command executed on an SPP 1000 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 1000 machines.

NOTE: Each use of the `set events` command overwrites its previous setting.

4. 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 type and number of events that can be collected per program run differ according to machine architecture:

- On SPP 1200 systems, you can monitor one set of on-processor events per program run.
- On SPP 1000 systems, you can monitor one type of off-processor event per program run.

Valid parameters to the `set events` command for specifying events on each architecture are listed in the following sections.

SPP 1000 off-processor events

On SPP 1000 systems, 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 data cache miss.
- **`remote_misses`**—Number of times the profiled regions of your program had to access remote memory (memory on another hypernode) because of a processor data cache miss. This implies use of the CTI (Convex Coherent Toroidal Interconnect) rings. This event is valid for multihypernode configurations only.
- **`local_and_remote_misses`**—Number of locally and remotely resolved data cache misses combined for the profiled regions of your program.

You can use one or both of the following parameters of the `set events` command to specify whether the type of off-processor 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`.

SPP 1200 on-processor events

SPP 1200 systems use HP PA-RISC 7200 processors which have on-processor event counters. This enables you to monitor one set of events per program run. Valid on-processor event parameters are listed below:

- **`data_cache`**—Configures SPP 1200 on-processor event counters to collect data cache access counts, miss counts, and latency.
- **`instruction_cache`**—Configures SPP 1200 on-processor event counters to collect instruction cache miss counts and latency.
- **`instruction_counts`**—Configures SPP 1200 on-processor event counters to collect completed instruction counts and clock cycles.

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, as well as instructions for creating and viewing reports, report filtering options, and a description of the information contained in the CXpa report header.

Report types

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

- Routines
- Serial loops
- 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. Performance reports that are available for profiled regions are listed below:

- **Counts report**—Iteration/execution counts.
- **Computation report**—CPU time and % total CPU time.
- **Time to solution report**—Wall clock time and CPU/wall clock time.
- **Events**—Available reports and metrics differ according to machine architecture and the type of event collected during the run of the program that generated the PDF file:
  - Off-processor event reports (SPP 1000 only)—Locally resolved cache misses, remotely resolved cache misses, or locally and remotely resolved cache misses.
  - On-processor event reports (SPP 1200 only)—Data cache misses and latency; data cache accesses; instruction cache misses and latency; or instructions completed and clock cycles.

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.

For a list of fields, abbreviations, and annotations that can appear in CXpa reports, refer to the "Report fields" online help topic or section of 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. You can create an Analysis Report window using one of the following methods:

- 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 create customized reports by specifying a subset of routines that contain regions of the currently selected type by selecting the Subset item from the Select Regions option menu in the Analysis Report window. Refer to the "Region Subset Selection dialog" online help topic or section of this book for more information.

Line mode

In line mode, use a form of the `analyze` command to display performance reports (for example, `analyze`, `analyze loop`, or `analyze pregion`). CXpa displays reports in line mode using the pager specified with your PAGER environment variable. 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.
- **Profile 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`.
- **Metric totals**—Totals for metrics collected across all threads of the process (for example, total wall clock time orCPU time).
- **Architecture**—Convex architecture where the PDF was created and related system information.

Related Topics

Basic Block report
Parallel Region reports
Reports
Introducing metrics
Report fields
Routine reports

Related Windows

Analysis Report window
Region Subset Selection dialog
Filter Report dialog
Visibility Selection 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 "Report fields" online help topic or section of this book 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 for test coverage analysis. 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 Basic Blocks as the region type.

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

Related Commands

`analyze`

Related Topics

Introducing metrics
Loop reports
Parallel Region reports
Report fields
Reports
Routine reports

Related Windows

Analysis Report window
Region Subset Selection dialog

Loop reports

Description

Loop performance analysis reports display metrics for profiled serial 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 serial loop regions that were executed:

- **Counts**—Iteration/execution counts.
- **Computation**—CPU time, including and excluding time spent in inner loops.
- **Time to solution**—Wall clock time and CPU/wall clock time.
- **Events**—Available event reports and metrics differ according to machine architecture and the type of event collected during the run of the program that generated the PDF file:
  - Off-processor event reports (SPP 1000 only)—Locally resolved cache misses, remotely resolved cache misses, or locally and remotely resolved cache misses.
  - On-processor event reports (SPP 1200 only)—Data cache misses and latency; data cache accesses; instruction cache misses and latency; and instructions completed and clock cycles.

All reports for serial loops display the following information:

- Line number corresponding to the source code for the loop—Line numbers for optimized loops annotated with a lowercase letter indicate that the loop was split into two or more loops during optimization.
- Number of times the loop was executed
- 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. This provides the most accurate picture of loop performance for your program.

  Refer to the "Report fields" online help topic or section of this book for a listing of the abbreviations shown in the Optimization column and their meaning.

  For a complete discussion of automatic optimizations performed by Convex compilers and manual optimization techniques for Convex SPP Series machines, refer to the *Exemplar Programming Guide* (Order No. DSW-067).

- Inclusive metric values—Includes values for inner loops (plus inner).
- Exclusive metric values—Does not include values for inner loops (less inner).

Refer to the "Report fields" section online help topic or section of this book for information about fields (colums), abbreviations, and annotations that can appear in loop reports.

Refer to the "Reports" or "`set visibility`" online help topics or sections of this book for information about filtering options for reports.

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

Reports

For loop regions, four types of reports are available: Iteration Counts, Computation, Time to Solution, and Events. These are described in the following sections, along with sample reports.

Iteration counts

The metrics in the Iteration Counts report for serial loops can be used to examine the following:

- Number of times the loop was executed
- Number of iterations per invocation (total, minimum, maximum, and average)

An Iteration Counts report for serial loops is shown in the following example. The Optimization column of the report shows that the loop at line 129 was split into two loops, and that one of those loops was partially unrolled, with a loop unrolling factor of 5 (`pU:5`).

```
================================================================================
                        Loop Performance Analysis
                                For: init
================================================================================
Optimized Loops:
                                                        Iteration Counts
                          Times
 Line    NL Optimization  Exec      Min       Max       Avg        Total     PS
-------  -- ----------- -------- --------- --------- --------- ------------ --
   128    0             1000000         5         9       9.0      8980000
   129a   1 pU:5        8980000         1         1       1.0      8980000
   129b   1             8962040         1         4       4.0     35740400
```

Computation

Loop computation reports are displayed for each routine containing profiled loop regions that were executed. The metrics in the Computation report for serial loops can be used to examine the amount of CPU time spent executing each loop:

- Excluding time spent in inner loops
- Including time spent in inner loops

A sample loop Computation report is shown in the following example.

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

Optimized Loops:
                                    Computation
                          Times       (less inner)          (plus inner)
 Line    NL Optimization  Exec      CPU Time              CPU Time              PS
-------  -- ----------- --------  ----------            ----------            --
   128    0             1000000     219.794               356.863
   129a   1 pU:5        8980000      68.728                68.728
   129b   1             8962040      68.340                68.340
```

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
- Ratio of CPU time to wall clock time

For all loops, if the CPU/wall clock ratio is low, it indicates that there may be some type of performance bottleneck caused by one or more of the following:

- 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 serial loops, if the CPU/wall clock ratio is high, the region is compute-bound.

For parallel loops, the CPU/wall clock ratio 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 that approach *n*, where *n* is the number of processors, indicate good parallel concurrency.

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. The high CPU/Wall ratio for the loops in this region (7.76 to 7.64) indicates near peak parallel concurrency This program was run on an SPP 1200 subcomplex with 8 CPUs.

```
================================================================================
                          Loop Performance Analysis
                              For: convolregion
================================================================================
Optimized Loops:
                                    Time to Solution
                          Times          (less inner)          (plus inner)
 Line    NL Optimization  Exec     Wall Clock CPU/Wall  Wall Clock CPU/Wall  PS
------- -- ----------- --------  ---------- --------  ---------- --------  --
   128   0              1000000      16.370     1.00      27.864     7.76
   129a  1 pU:5          8980000       5.717     1.00       5.717     7.76
   129b  1              8962040       5.778     1.00       5.778     7.64
```

Events

If you have chosen to collect events, CXpa displays event reports for serial loops. The types of event reports displayed vary according to machine architecture and the type of event or events collected for the program run that generated the PDF file.

- Refer to the *Exemplar SPP 1000/1200 Architecture* (Order No. DHW-014) reference for more information about these architectures.
- Refer to the "Introducing metrics" online help topic or section of this book for more details on the types of event metrics that can be collected on a specific architecture.

Cache misses and latency

Cache miss and latency reports vary according to the architecture and the exact type of cache miss event collected for the run of the program that generated the PDF file being analyzed.

On SPP 1000 machines, you can configure off-processor performance counters to collect cache miss counts and latency time for total cache misses, cache misses that were resolved locally (on the same hypernode as the processor—CTI not used), or cache misses that were resolved remotely (on a different hypernode—CTI used). The cache miss report heading indicates the type of event collected for that run of your program, and can be one of the following:

- Locally resolved cache misses and latency (read only, write only, or both)

- Remotely resolved cache misses and latency (read only, write only, or both)
- Locally and remotely resolved cache misses and latency (read only, write only, or both)

On SPP 1200 machines, you can configure on-processor performance counters to collect data cache accesses, misses, and latency or instruction cache misses and latency. The cache misses and latency report heading indicates the type of event collected, and can be one of the following (data cache accesses are displayed in a separate report):

- Data cache misses and latency
- Instruction cache misses and latency

Cache misses and latency reports for loops enable you to examine:

- Total number of cache miss events that occurred in a loop
- Latency—The total wall clock time spent accessing memory to locate data or instructions not found in the processor's cache.
- % CPU—Ratio of latency time to CPU time, expressed as a percentage.

  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 milliseconds, then the data is probably not significant.
- Differences in latency and cache miss counts among threads can indicate data access patterns for threads that cause memory bank contention among threads or cache thrashing.

The following sample cache miss report for loops was generated on an SPP 1000 system. For this program run, locally resolved cache misses were collected for both read and write operations. Locally resolved cache miss counts indicate the number of cache misses that are resolved by using the hypernode crossbar to access memory found on the same hypernode as the processor. The CTI (Convex Coherent Toroidal Interconnect) is not used.

```
================================================================================
                          Loop Performance Analysis
                              For: convolregion
================================================================================
Optimized Loops:
            Locally and Remotely Resolved Cache Misses and Latency
                          Times                    (less inner)
 Line     NL Optimization  Exec       Number of        Latency      % CPU     PS
-------  -- -----------  --------   ---------------  ------------- --------  --
   128    0                160000           398597          0.226     0.13%
   129a   1 pU:5          7192000          3746200          2.081     3.16%
   129b   1               7015760           294035          0.174     0.33%

                          Times                    (plus inner)
 Line     NL Optimization  Exec       Number of        Latency      % CPU     PS
-------  -- -----------  --------   ---------------  ------------- --------  --
   128    0                160000          4438832          2.481     0.85%
   129a   1 pU:5          7192000          3746200          2.081     3.16%
   129b   1               7015760           294035          0.174     0.33%

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

Data cache accesses (SPP 1200 only)

On SPP 1200 systems, you can configure on-processor event counters to collect data cache misses, accesses, and latency. A separate Data Cache Accesses report provides the following information:

- Total number of data cache accesses
- Data cache hit rate (`% Hits` column)—The data cache hit rate is calculated as follows:

$$data\_cache\_hit\_rate = \frac{total\_data\_cache\_accesses - data\_cache\_misses}{data\_cache\_accesses} \times 100$$

If the data cache hit rate (`% Hits`) is low, this indicates cache thrashing.

A sample Data Cache Accesses report for loops is shown below.

```
================================================================================
                          Loop Performance Analysis
                             For: convolregion
================================================================================
Optimized Loops:
                                Data Cache Accesses
                            Times                    (less inner)
 Line     NL Optimization   Exec       Number of                    % Hits   PS
-------  -- -----------  --------   ---------------                --------  --
   128    0              1000000        8845854107                   98.9%
   129a   1 pU:5         8980000        1926684484                   97.7%
   129b   1              8962040        1859999803                   97.5%

                            Times                    (plus inner)
 Line     NL Optimization   Exec       Number of                    % Hits   PS
-------  -- -----------  --------   ---------------                --------  --
   128    0              1000000       12632538394                   98.5%
   129a   1 pU:5         8980000        1926684484                   97.7%
   129b   1              8962040        1859999803                   97.5%
```

Instructions completed and clock cycles (SPP 1200 only)

On SPP 1200 systems, you can configure on-processor event counters to collect instructions completed and clock cycles. A separate Instructions Completed and Clock Cycles report provides the following information:

- Number of instructions completed
- Average clock cycles (cycles per instruction)
- Average MIPS (millions of instructions per second) rate, calculated only if wall clock time is also collected

The average clock cycles metric measures the efficiency of instruction scheduling for the region being profiled. It is computed during analysis if instruction counts and clock cycles are collected. On SPP 1200 systems, the maximum number of instructions that can be executed in a single clock cycle is 2. This means that the theoretical peak value for the average clock cycles metric on this architecture is 0.5.

If the value for average clock cycles is high, and it is associated with a frequently called routine that performs only a small amount of work (for example, a routine that simply assigns a new value to a variable and returns), inlining the routine could improve the instruction scheduling by eliminating the routine call overhead.

Code sections that contain many patterns of consecutive loads and stores followed by immediate use of requested operands can also result in high average clock cycles.

In some cases, the instruction scheduling can be improved by compiling the routine at a higher optimization level or, if programming at the assembly level, changing the order of instructions. Refer to the *Hewlett-Packard PA-RISC 1.1 Architecture and Instruction Set Reference Manual* (Manual Part No. 09740-90039) for information on how to achieve optimal instruction scheduling.

The average MIPS (millions of instructions per second) rate is calculated during analysis if wall clock time is also collected. The formula CXpa uses to calculate the average MIPS rate is as follows:

$$Average\_MIPS\_rate = \frac{Number\_of\_instructions\_completed}{Wall\_clock\_time\_in\_sec}$$

The theoretical peak MIPS rate for Hewlett-Packard PA-RISC 7100/7200 processors with a 10-ns clock cycle at a clock rate of 100 MHz is 200 MIPS. There is a direct correlation between average clock cycles and average MIPS rates; decreasing average clock cycles will result in increased average MIPS rates.

Loop reports

A sample Instructions Completed and Clock Cycles report for loops is shown below.

```
================================================================================
                          Loop Performance Analysis
                             For: convolregion
================================================================================
Optimized Loops:
                    Instructions Completed and Clock Cycles
                                                  (less inner)
                                 Times                        Avg Clock   Avg MIPS
 Line     NL Optimization        Exec       Number of           Cycles       Rate     PS
-------  -- ----------- --------  ---------------  ------------- --------  --
   128    0              1000000       14263931087           1.6        63
   129a   1 pU:5         8980000        3838782979           1.9        54
   129b   1              8962040        3624657999           2.0        51

                                                  (plus inner)
                                 Times                        Avg Clock   Avg MIPS
 Line     NL Optimization        Exec       Number of           Cycles       Rate     PS
-------  -- ----------- --------  ---------------  ------------- --------  --
   128    0              1000000       21727372065           1.7        59
   129a   1 pU:5         8980000        3838782979           1.9        54
   129b   1              8962040        3624657999           2.0        51

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

Context

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

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

Related Commands

`analyze`

Related Topics

Basic Block report
Parallel Region reports
Routine reports
Introducing metrics
Reports

Related Windows

Analysis Report window
Region Subset Selection 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 loops:

- **Time to solution**—Includes wall clock time, CPU/wall clock time, and chunk counts.
- **Events**—Available event reports and metrics differ according to machine architecture and the type of event collected during the run of the program that generated the PDF file:
  - Off-processor event reports (SPP 1000 only)—Locally resolved cache misses, remotely resolved cache misses, or locally and remotely resolved cache misses.
  - On-processor event reports (SPP 1200 only)—Data cache misses and latency; data cache accesses; instruction cache misses and latency; and instructions completed and clock cycles.

Each parallel loop region report contains two sections:

- `Optimized Loops (cumulative, including spawn/join overhead)`—These metrics are cumulative across all threads executing in the parallel region and include spawn and join overhead. These values are graphed in the 2D Profile window for parallel loops.
- `Optimized Loops (by thread, excluding spawn/join overhead)`—These metrics are calculated on a per-thread basis for all threads executing in the parallel region and do not include spawn and join overhead. These values are graphed in the 3D Profile window for parallel loops.

Refer to the "Report fields" online help topic or section of this book for information about fields (columns), abbreviations, and annotations that can appear in parallel region reports.

Refer to the "Reports" or "`set visibility`" online help topics or sections of this book for information about filtering options for reports.

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

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 loops can be used to examine the following:

- Total wall clock time spent in each parallel loop
- Distribution of work across threads—By looking at CPU time, wall clock time, and/or chunk counts, you can spot parallel loop 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 *chunk* is a unit of work executed on a single thread in a parallel loop. This unit of work corresponds to a packet of iterations of a parallelized loop assigned to execute on a single thread. CXpa currently does not track the number of iterations in a chunk (chunk size).
- Ratio of CPU time to wall clock time

  If the CPU/wall clock ratio is high, the region is compute-bound.

  If the CPU/wall clock ratio is low, it indicates that there may be some type of performance bottleneck caused by one or more of the following:
  - 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 ratio is the concurrency factor for the parallel loop. (Parallel loops are annotated with a "P" in the Optimization column.) 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.

A sample Time to Solution report for a parallel region is shown below. The cumulative CPU/wall clock ratio for all threads indicates good parallel concurrency, and a comparison of metrics for individual threads indicates an even distribution of work among threads for this loop.

```
================================================================================
                    Parallel Region Performance Analysis
                                For: convol
================================================================================
Optimized Loops (cumulative, including spawn/join overhead):

                                Time to Solution
 Line    Times Exec                    CPU        Wall Clock   CPU/Wall  PS
-------  ----------              -------------  ------------- --------  --
    76a           1                    36.364          6.519     5.58

Optimized Loops (by thread, excluding spawn/join overhead):

                                Time to Solution
 Line    TID  Chunks                    CPU        Wall Clock   CPU/Wall  PS
-------  --- --------              -------------  ------------- --------  --
    76a    0        1                    4.699          6.516     0.72
           1        1                    4.517          6.102     0.74
           2        1                    4.541          5.890     0.77
           3        1                    4.532          5.789     0.78
           4        1                    4.518          6.022     0.75
           5        1                    4.528          5.823     0.78
           6        1                    4.524          5.804     0.78
           7        1                    4.505          5.892     0.76
================================================================================
```

Events

If you have chosen to collect events, CXpa displays event reports for parallel loop regions. The types of event reports displayed vary according to machine architecture and the type of event or events collected for the program run that generated the PDF file.

- Refer to the *Exemplar SPP 1000/1200 Architecture* (Order No. DHW-014) reference for more information about these architectures.
- Refer to the "Introducing metrics" online help topic or section of this book for more details on the types of event metrics that can be collected on a specific architecture.

Cache misses and latency

Cache misses and latency reports vary according to the architecture and the exact type of cache miss event collected for the run of the program that generated the PDF file being analyzed.

On SPP 1000 machines, you can configure off-processor performance counters to collect cache miss counts and latency time for total cache misses, cache misses that were resolved locally (on the same hypernode as the processor—CTI not used), or cache misses that were resolved remotely (on a different hypernode—CTI used). The cache miss report heading indicates the type of event collected for that run of your program, and can be one of the following:

- Locally resolved cache misses and latency (read only, write only, or both)
- Remotely resolved cache misses and latency (read only, write only, or both)
- Locally and remotely resolved cache misses and latency (read only, write only, or both)

On SPP 1200 machines, you can configure on-processor performance counters to collect data cache accesses, misses, and latency or instruction cache misses and latency. The cache misses and latency report heading indicates the type of event collected, and can be one of the following (data cache accesses are displayed in a separate report):

- Data cache misses and latency
- Instruction cache misses and latency

Cache misses and latency reports for loops enable you to examine:

- Total number of cache miss events that occurred in a loop
- Latency—The total wall clock time spent accessing memory to locate data not found in the cache.
- % CPU—Ratio of latency time to CPU time, expressed as a percentage.

  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 milliseconds, then the data is probably not significant.

- Balance of work distributed among threads—Chunk counts directly indicate how well work is distributed across threads.
- Differences in latency and cache miss counts among threads can indicate data access patterns for threads that cause memory bank contention among threads or cache thrashing.

The following sample cache miss report for loops was generated on an SPP 1000 system. For this program run, locally resolved cache misses were collected for both read and write operations. Locally resolved cache miss counts indicate the number of cache misses that are resolved by using the hypernode crossbar to access memory found on the same hypernode as the processor (the CTI is not used).

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

Optimized Loops (cumulative, including spawn/join overhead):
                        Local Cache Miss Events
 Line     Times Exec     Number of      Latency       % CPU                   PS
-------  ----------  -------------  -------------  --------                   --
    36a           1         394634          0.196      14.2%

Optimized Loops (by thread, excluding spawn/join overhead):
                        Local Cache Miss Events
 Line     Thread Id      Number of       Latency        Chunks       % CPU    PS
-------  ----------  -------------  -------------  -------------  --------  --
    36a           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%
================================================================================
```

Data cache accesses (SPP 1200 only)

On SPP 1200 systems, you can configure on-processor event counters to collect data cache misses, accesses, and latency. A separate Data Cache Accesses report provides the following information:

- Total number of data cache accesses
- Data cache hit rate (`% Hits` column)—The data cache hit rate is calculated as follows:

$$data\_cache\_hit\_rate = \frac{total\_data\_cache\_accesses - data\_cache\_misses}{data\_cache\_accesses} \times 100$$

If the data cache hit rate (`% Hits`) is low, this indicates cache thrashing.

A sample Data Cache Accesses report for a parallel loop is shown below.

```
================================================================================
                    Parallel Region Performance Analysis
                                For: convol
================================================================================
Optimized Loops (cumulative, including spawn/join overhead):

                              Data Cache Accesses
 Line     Times Exec                Number of                 % Hits    PS
-------  ----------             ---------------              --------   --
    76a           1                 10451916305                 98.6%

Optimized Loops (by thread, excluding spawn/join overhead):

                              Data Cache Accesses
 Line     TID  Chunks                Number of                 % Hits    PS
-------  --- --------             ---------------              --------   --
    76a    0        1                 1082792895                 98.6%
           1        1                 1187502472                 98.6%
           2        1                 2362213002                 98.6%
           3        1                 1187497307                 98.6%
           4        1                 1187477250                 98.6%
           5        1                 1187498467                 98.6%
           6        1                 1187560947                 98.6%
           7        1                 1069369057                 98.6%
================================================================================
```

Instructions completed and clock cycles (SPP 1200 only)

On SPP 1200 systems, you can configure on-processor event counters to collect instructions completed and clock cycles. A separate Instructions Completed and Clock Cycles report provides the following information:

- Number of instructions completed
- Average clock cycles (cycles per instruction)
- Average MIPS (millions of instructions per second) rate, which is computed only if wall clock time is also collected

On SPP 1000 and SPP 1200 systems, the maximum number of instructions that can be executed in a single clock cycle is 2. This means that the theoretical best value for the average clock cycles metric on these architectures is 0.5.

Code sections that contain many patterns of consecutive loads and stores followed by immediate use of requested operands can also result in high average clock cycles.

In some cases, the instruction scheduling can be improved by compiling the routine at a higher optimization level or, if programming at the assembly level, changing the order of instructions. Refer to the *Hewlett-Packard PA-RISC 1.1 Architecture and Instruction Set Reference Manual* (Manual Part No. 09740-90039) for information on how to achieve optimal instruction scheduling.

The average MIPS (millions of instructions per second) rate is computed during analysis if wall clock time is also collected. The formula CXpa uses to calculate the average MIPS rate is as follows:

$$Average\_MIPS\_rate = \frac{Number\_of\_instructions\_completed}{Wall\_clock\_time\_in\_sec}$$

The theoretical peak MIPS rate for Hewlett-Packard PA-RISC 7100/7200 processors with a 10-ns clock cycle at a clock rate of 100 MHz is 200 MIPS. There is a direct correlation between average clock cycles and average MIPS rates; decreasing average clock cycles will result in increased average MIPS rates.

Parallel Region reports

A sample instructions and clock cycles report for a parallel loop is shown below.

```
================================================================================
                    Parallel Region Performance Analysis
                                For: convol
================================================================================
Optimized Loops (cumulative, including spawn/join overhead):
                    Instruction Completed and Clock Cycles
                                                          Avg Clock    Avg MIPS
 Line     Times Exec                    Number of           Cycles       Rate     PS
-------  ----------                 ---------------  -------------  --------  --
    76a           1                    2814188100            1.3        77

Optimized Loops (by thread, excluding spawn/join overhead):

                    Instructions Completed and Clock Cycles
                                                          Avg Clock    Avg MIPS
 Line     TID  Chunks                    Number of           Cycles       Rate     PS
-------  ---  --------              ---------------  -------------  --------  --
    76a    0         1                  363366280            1.3        55
           1         1                  350258608            1.3        57
           2         1                  350325595            1.3        59
           3         1                  350312174            1.3        60
           4         1                  350255945            1.3        58
           5         1                  350283183            1.3        60
           6         1                  350286464            1.3        60
           7         1                  349091249            1.3        59

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

Context

To view parallel region reports from the Analysis Report window in X window mode, select Parallel Loops as the region type. To view parallel region reports in line mode, use the `analyze pregion` command.

Related Topics

Introducing metrics
Reports
Loop reports
Routine reports

Related Windows

Analysis Report window
Region Subset Selection dialog

Related Commands

`analyze`

Report fields

This section lists and describes (columns), abbreviations, annotations, and other terms that appear in CXpa reports. These are listed in alphabetical order.

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

- `% CPU`—Ratio of cache miss latency to CPU time, expressed as a percentage.
- `% Hits`—Data cache hit rate. SPP 1200 Data Cache Accesses report only.
- `Avg clock cycles`—Average number of clock cycles per instruction. SPP 1200 Instructions Completed and Clock cycles report only.
- `Avg MIPS rate`—Average number of MIPS (millions of instructions per second). SPP 1200 Instructions Completed and Clock cycles report only.
- `Chunks`—Lists the total number of chunks executed by a single thread in the parallel region. A *chunk* is a unit of work executed on a single thread in a parallel loop. This unit of work corresponds to a packet of iterations of a parallel loop assigned to execute on a single thread. CXpa currently does not track the number of iterations in a chunk (chunk size).
- `CPU Time`—CPU time spent executing the region (accumulated time for all threads).
- `CPU/Wall`—For serial code, the ratio of CPU time to wall clock time measures CPU utilization. For parallel regions, this metric indicates the concurrency achieved through parallelism.
- `Iteration Count`—Minimum, maximum, and average loop iteration counts are shown because the iteration count can vary from one invocation of a loop to another:
  - `Min`—Fewest number of iterations of this loop for a single invocation.
  - `Max`—Greatest number of iterations of this loop for a single invocation.
  - `Avg`—Average number of iterations of this loop for a single invocation.

- `Latency`—The amount of time spent accessing memory to locate data or instructions not found in the processor's cache.
- `(less children)`—Does not include values for called routines. However, 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.
- `(less inner)`—Excluding time spent in inner loops.
- `Line`—Source line number of the region.
  - Basic blocks—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.
  - Routines—Starting source file line number of the routine.
  - Serial and parallel loops—Line numbers for optimized loops annotated with a lowercase letter indicate that the loop was split into two or more loops during optimization. For parallel loops, the line number is the starting line number that maps back to the original source code.
- `Locally Resolved Cache Misses`—Cache misses that are resolved by using the hypernode crossbar to access memory found on the same hypernode as the processor; the CTI rings are not used.
- `m`—Time is expressed in milliseconds.
- `NL`—Nesting level of the loop after optimization.
- `Number of`—Number of times that the selected event occurred.
- `Optimization`—Abbreviation for the transformation(s) that the compiler performed on the loop. The abbreviations possible are:
  - `B:`*n*—Loop blocking; *n* is the blocking factor, which is the number of iterations that were blocked together.
  - `cU:`*n*—The loop was completely unrolled; *n* is the loop unrolling factor, which is the number of iterations that were unrolled.
  - `D`—Distributed
  - `Ds`—Dynamic selection
  - `Hs`—Hoisted
  - `I`—Interchanged
  - `P`—Parallel
  - `PS`—Parallel strip-mined
  - `pU:`*n*—The loop was partially unrolled. *n* is the loop unrolling factor, which is the number of iterations that were unrolled.
  - `UL`—Uninstrumentable

For more information about automatic optimizations performed by the compiler and manual optimization techniques for Convex Exemplar SPP Series machines, refer to the *Exemplar Programming Guide* (DSW-067).

- `Optimized Loops (cumulative, including spawn/join overhead)`—These metrics are cumulative across all threads executing in the parallel region and include spawn and join overhead. These values are graphed in the 2D Profile window for parallel loops.
- `Optimized Loops (by thread, excluding spawn/join overhead)`—These metrics are calculated on a per-thread basis for all threads executing in the parallel region and do not include spawn and join overhead. These values are graphed in the 3D Profile window for parallel loops.
- `PC Value`—Starting address of a basic block.
- `plus children`—Includes values for called routines.
- `plus inner`—Including time spent in inner loops.
- `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 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: • Deselect 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 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.

- `read only`—Cache miss event collection was specified for misses that occurred during reads (loads) only.
- `Remotely Resolved Cache Misses`—Cache misses that were resolved by using the CTI rings to access memory on another hypernode.
- `TID`—Kernel thread ID number for each thread executing the parallel region.
- `Times Exec`—Number of times the routine or basic block was executed or, for loops, the number of loop invocations.
- `Total Blocks`—Total number of basic blocks in a routine.
- `Total Blocks Executed`—Total number of blocks that were executed and the percentage of blocks executed.
- `Wall Clock`—Time to solution for all threads executing the region. Wall clock time includes time when threads are idle.
- `write only`—Cache miss event collection was specified for misses that occurred during writes (stores) only.

Related Topics

Basic Block reports
Loop reports
Reports
Introducing metrics
Parallel Region reports
Routine reports

Related Windows

Analysis Report window
Region Subset Selection 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:

- **Call counts**—Routine execution counts.
- **Computation**—CPU time and the percentage of total CPU time (% column), including and excluding time spent in called routines.
- **Time to Solution**—Wall clock time, percentage of total wall clock time, (% column), and CPU/wall clock time.
- **Events**—Available event reports and metrics differ according to machine architecture and the type of event collected during the run of the program that generated the PDF file. These reports include:
  - Off-processor event reports (SPP 1000 only)—Locally resolved cache misses, remotely resolved cache misses, or locally and remotely resolved cache misses.
  - On-processor event reports (SPP 1200 only)—Data cache misses and latency; data cache accesses; instruction cache misses and latency; and instructions completed and clock cycles.

For all routine reports, metrics are displayed:

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

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

Refer to the "Report fields" online help topic or section of this book for information about fields (columns), abbreviations, and annotations that can appear in Routine reports.

Reports

For routines, four types of reports are available: Call Counts, Computation, Time to Solution, and Events. These reports are described in the following sections, along with sample reports.

Call counts

The Call Counts report displays the number of times each profiled routine in your program was executed (called). A sample Call Counts report is shown below.

```
================================================================================
                        Routine Performance Analysis
================================================================================
                                   Call Counts
   Times Exec     PS                          Routine Name
---------------- -- ------------------------------------------------------------
      1000000      convolregion
            1      initialize
            1      convol
            1 e    start
            1      matcon

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

Computation

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

- Total CPU time spent in each profiled routine
- Percentage of program's total CPU time for each profiled 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)
  CPU Time      %               CPU Time      %                 PS Routine Name
------------  -----           ------------  -----               -- ------------
      22.219   53.4%                22.219   53.4%                  convolregion
       2.573    6.2%                41.276   99.3%                  convol
       0.040    0.1%                 0.306    0.7%                  initialize
       2.422m   0.0%                41.585  100.0%                e start
       0.765m   0.0%                41.583  100.0%                   matcon

(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 each profiled routine
- Percentage of total wall clock time spent in each profiled routine
- The ratio of CPU time to wall clock time for each profiled routine

If the CPU/wall clock ratio is low, it indicates that there may be some type of performance bottleneck caused by one or more of the following:

- 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 serial regions, if the CPU/wall clock ratio is high, the region is compute-bound.

For parallel regions, the CPU/wall clock ratio is the concurrency factor for the parallel region. Values for CPU/wall clock time that approach *n*, where *n* is the number of processors, indicate good parallel concurrency.

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.

A sample Time to Solution report for routines is shown below. The program that generated the PDF contained regions that executed in parallel on 8 CPUs. Routine `convolregion` (which perform the core computation and executes in parallel) exhibits good parallel concurrency.

```
================================================================================
                          CXpa Version 3.2 Profile

Executable      : /src/demos/convol/convol.-O3.exe
Profile Data    : /src/demos/convol/convol.-O3.exe.pdf
Process State   : exited
CPU Time        : 258.553  secs
Wall Clock Time : 55.352  secs
Architecture    : CONVEX SPP-1200 (8 cpus)
================================================================================
                        Routine Performance Analysis
================================================================================

                                Time to Solution
        (less children)                    (plus children)
 Wall Clock     %    CPU/Wall   Wall Clock     %    CPU/Wall   PS Routine Name
------------  -----  --------- ------------  -----  ---------  -- ------------
      47.888  86.5%       5.34        47.888  86.5%       5.34     convolregion
       6.615  11.9%       0.05        48.238  87.1%       5.36     convol
       6.584  11.9%       0.00        55.344 100.0%       4.67  e  start
       0.513   0.9%       0.01         0.513   0.9%       0.11     initialize
       9.270m  0.0%       0.09        48.760  88.1%       5.30     matcon

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

Events

If you have chosen to collect events, CXpa displays event reports for routines. The types of event reports displayed vary according to machine architecture and the type of event or events collected for the program run that generated the PDF file.

- Refer to the *Exemplar SPP 1000/1200 Architecture* (Order No. DHW-014) reference for more information about these architectures.
- Refer to the "Introducing metrics" online help topic or section of this book for more details on the types of event metrics that can be collected on a specific architecture.

Cache misses and latency

Cache misses and latency reports vary according to the architecture and the exact type of cache miss event collected for the run of the program that generated the PDF file being analyzed.

On SPP 1000 machines, you can configure off-processor performance counters to collect cache miss counts and latency time for total cache misses, cache misses that were resolved locally (on the same hypernode as the processor—CTI not used), or cache misses that were resolved remotely (on a different hypernode—CTI used). The cache miss report heading indicates the type of event collected for that run of your program, and can be one of the following:

- Locally resolved cache misses and latency (read only, write only, or both)
- Remotely resolved cache misses and latency (read only, write only, or both)
- Locally and remotely resolved cache misses and latency (read only, write only, or both)

On SPP 1200 machines, you can configure on-processor performance counters to collect data cache accesses, misses, and latency or instruction cache misses and latency. The cache misses and latency report heading indicates the type of event collected, and can be one of the following (data cache accesses are displayed in a separate report):

- Data cache misses and latency
- Instruction cache misses and latency

Cache misses and latency reports for routines enable you to examine:

- Total number of cache miss events that occurred in a routine or routines.
- Latency—The total wall clock time spent accessing memory to locate data or instructions not found in the processor's cache.
- % CPU—Ratio of latency time to CPU time, expressed as a percentage.

  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 milliseconds, then the data is probably not significant.

- Differences in latency and cache miss counts indicate data access patterns for routines that cause memory bank contention among threads or cache thrashing.

The following sample cache miss report for routines was generated on an SPP 1000 system. For this program run, locally resolved cache misses were collected for both read and write operations. Locally resolved cache miss counts indicate the number of cache misses that are resolved by using the hypernode crossbar to access memory found on the same hypernode as the processor; the CTI (Convex Coherent Toroidal Interconnect) rings are not used.

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

                  Locally Resolved Cache Misses and Latency
         (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.

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

Data cache accesses (SPP 1200 only)

On SPP 1200 systems, you can configure on-processor event counters to collect data cache misses, accesses, and latency. A separate data cache accesses report provides the following information:

- Total number of data cache accesses
- Data cache hit rate (`% Hits` column)—The data cache hit rate is calculated as follows:

$$data\_cache\_hit\_rate = \frac{total\_data\_cache\_accesses - data\_cache\_misses}{data\_cache\_accesses} \times 100$$

If the data cache hit rate (`% Hits`) is low, this indicates cache thrashing.

A sample Data Cache Accesses report for routines is shown below.

```
================================================================================
                        Routine Performance Analysis
================================================================================
                              Data Cache Accesses
        (less children)                    (plus children)              Routine
   Number of              % Hits     Number of              % Hits  PS  Name
---------------           ------  ---------------           ------ --  -------
    10340938705            98.6%    10340938705            98.6%    convolregion
       15199949             99.2%      10451917755            98.6%      convol
         189206            96.7%         1481063            97.2%     initialize
          24179             92.9%      10453427804            98.6% e   start
           4807             88.5%      10453403625            98.6%     matcon

(e) incomplete, contains exit.
```

Instructions completed and clock cycles (SPP 1200 only)

On SPP 1200 systems, you can configure on-processor event counters to collect instructions completed and clock cycles. A separate Instructions Completed and Clock Cycles report provdes the following information:

- Number of instructions completed
- Average clock cycles (cycles per instruction)
- Average MIPS (millions of instructions per second) rate, which is only calculcated if wall clock time is also collected

The average clock cycles metric measures the efficiency of instruction scheduling. On SPP 1200 systems, the maximum number of instructions that can be executed in a single clock cycle is 2. This means that the theoretical peak value for the average clock cycles metric on this architecture is 0.5.

If the value for average clock cycles is high, and it is associated with a frequently called routine that performs only a small amount of work (for example, a routine that simply assigns a new value to a variable and returns), inlining the routine could improve the instruction scheduling by eliminating the routine call overhead.

Code sections that contain many patterns of consecutive loads and stores followed by immediate use of requested operands can also result in high average clock cycles.

In some cases, the instruction scheduling can be improved by compiling the routine at a higher optimization level or, if programming at the assembly level, changing the order of instructions. Refer to the *Hewlett-Packard PA-RISC 1.1 Architecture and Instruction Set Reference Manual* (Manual Part No. 09740-90039) for information on how to achieve optimal instruction scheduling.

The average MIPS (millions of instructions per second) rate is calculated during analysis if wall clock time is also collected. The formula CXpa uses to calculate the average MIPS rate is as follows:

$$Average\_MIPS\_rate = \frac{Number\_of\_instructions\_completed}{Wall\_clock\_time\_in\_sec}$$

The theoretical peak MIPS rate for Hewlett-Packard PA-RISC 7100/7200 processors with a 10-ns clock cycle at a clock rate of 100 MHz is 200 MIPS. There is a direct correlation between average clock cycles and average MIPS rates; decreasing average clock cycles will result in increased average MIPS rates.

A sample Instructions Completed and Clock Cycles report for routines is shown below.

```
================================================================================
                        Routine Performance Analysis
================================================================================
                  Instructions Completed and Clock Cycles
        (less children)                    (plus children)
                Avg Clock Avg MIPS                 Avg Clock Avg MIPS Routine
     Number of      Cycles  Rate        Number of     Cycles  Rate PS  Name
------------------- ------ ----- ------------------- ------ ----- -- -------
         1723077485    1.2    82          1723077485    1.2    82    convolregion
          148537089    1.4    70           2814206267    1.3    77     convol
            1055764    3.8     0              8499130    3.9     6    initialize
              49067    4.9     0           2822765035    1.3    76 e  start
              10571    7.5     1           2822715968    1.3    76     matcon

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

Context

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

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

Related Topics

Basic Block reports
Introducing metrics
Loop reports
Parallel Region reports
Report fields
Reports

Related Windows

Analysis Report window
Region Subset Selection dialog

Related Commands

`analyze`

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.

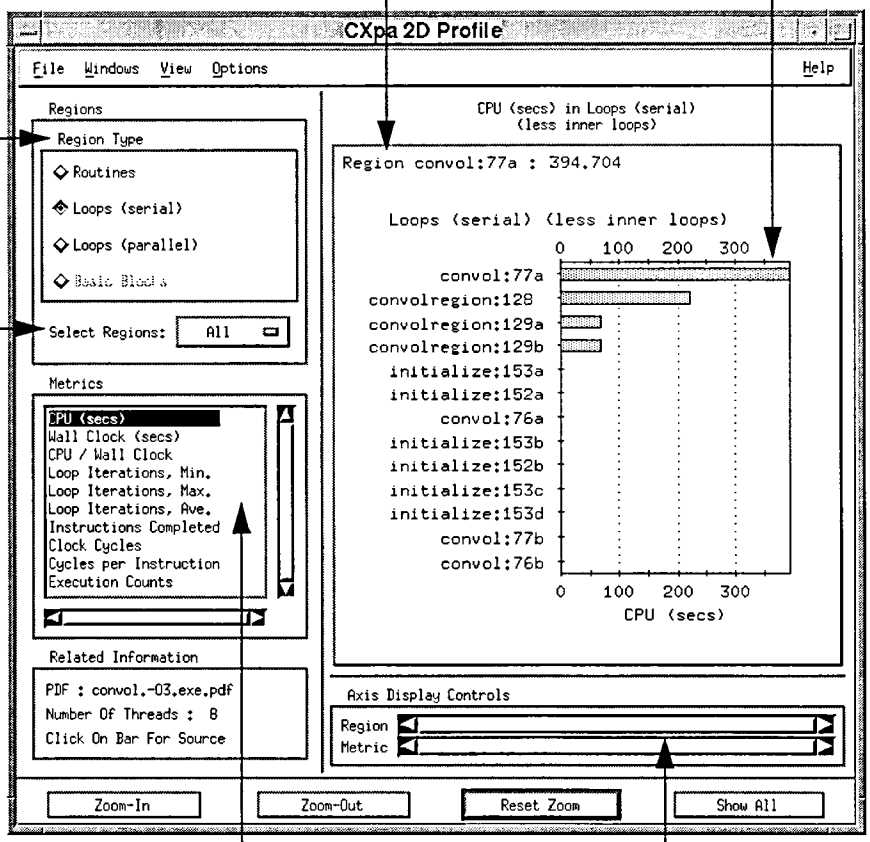
2D Profile window

Interactive data value display updates as you move the mouse over the graph, enabling you to view exact data values for regions in the graph.

Click with the left mouse button on any bar in the graph to view source code associated with a region

Select the type of region to graph

Select all regions of the selected type or a subset to graph



Select the type of metric to graph.
The list is updated as different source code regions are selected, depending on the metrics available in the current PDF file.

Independently scroll Region and Metric axes

2D Profile window

Description

The 2D Profile window displays a customizable two-dimensional graph 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.

In the 2D Profile window you can

- Dynamically select different regions and metrics to graph.
- Click with the left mouse button on any bar in the graph to display associated source code.
- Save 2D profile graphs in PostScript, xwd, or ASCII formats (select the Save Profile option from the File menu).
- View or specify the range of metric values displayed or the number of regions displayed at one time in the 2D graph using the Zoom dialog or buttons along the bottom of the window. This can be useful when there are a large number of data items to graph and you want to focus on a subset of the data.

  To access the Zoom dialog, select the Zoom option from the View menu.
- View exact data values for regions in the graph by moving the mouse over regions in the graph (data values are displayed in the upper left corner of the graph).
- Use the scrollbars in the Axis Display Controls panel to scroll the Region and Metric axes independently.
- Sort data in 2D profile graphs alphabetically, by load order, from lowest to highest, or from highest to lowest. The default sort order is from highest to lowest. To access sort options, select the Sort option from the View menu.

By default, CXpa graphs metric values for all selected regions or for the maximum number of regions for which labels can legibly be displayed, whichever is less. If all selected regions are not displayed, use the Region Axis Display Control scrollbar to navigate the graph or use one of the other zoom options to change the graph display.

Menus

Name	Meaning
File	Contains options for saving the graph in a PostScript, ASCII, or xwd file format 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 (Sort) and changing the range of values or source code regions displayed in the graph (Zoom).
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 options for invoking the CXpa help system.

Regions

Contains buttons that change the type of source code region to graph on the Region axis.

Name	Meaning
Routines	Graph selected metric for routines.
Loops (serial)	Graph selected metric for serial loops.
Loops (parallel)	Graph selected metric for parallel loops.
Basic Blocks	Graph selected metric for basic blocks.
Select Regions	Option menu containing two items: **All**—Graph profiling data for all regions of the type specified above. **Subset**—Brings up the Region Subset Selection dialog where you can specify a customized list of routines to graph profiling data for.

Metrics

Contains a scrolled list of metrics available in the current PDF where you can select the type of metric graphed on the Metric axis.

The available metric choices depend on the regions selected for profiling and the metrics that were collected for the run of your program that produced the PDF file.

Refer to the "Introducing metrics" online help topic for metric descriptions and a list of metrics available by architecture.

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.

Axis Display Controls

Using the scrollbars in the Axis Display Controls panel, you can scroll the Region and Metrics axes independently.

Buttons

Name	Meaning
Reset Zoom	Restores the default graph display. By default, CXpa graphs metric values for all selected regions or for the maximum number of regions for which labels can legibly be displayed, whichever is less. If all selected regions are not displayed, use the Region Axis Display Control scrollbar to navigate the graph or use one of the other zoom options to change the graph display.
Zoom-In	Reduces the number of regions displayed in the 2D graph at one time by half.
Zoom-Out	Doubles the number of regions displayed in the 2D graph at one time.
Show All	Displays the entire graph. If the number of regions is so large that the region labels cannot be displayed legibly, labels are not displayed. If this happens, click the Reset Zoom or Zoom-In button to restore the labels and use scrollbars to navigate the graph.

Context

The 2D Profile window appears when you

- Press the Create 2D Profile button in the Analysis Control 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` or specified `-windows 2DProfile` when invoking CXpa on the command line.

Related Topics

Introducing metrics

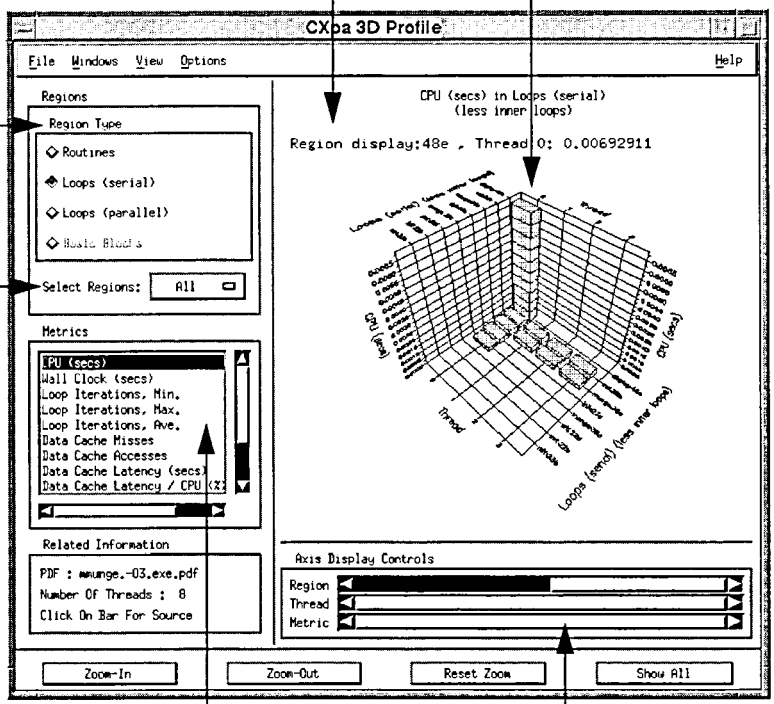
Related Windows

3D Profile window
Analysis Control window
Analysis Report window
Executable Manager window
Filter Profile dialog
Profile Selection dialog
Region Subset Selection dialog
Save Profile dialog
Sort dialog
Visibility Selection dialog
Zoom dialog

3D Profile window

Interactive data value display updates as you move the mouse over the graph, enabling you to view exact data values for regions in the graph.

Click with the left mouse button on any bar in the graph to view source code associated with a region.



Select the type of region to graph

Select all regions of the selected type or a subset to graph

Select a metric to graph on the Metric axis. The list is updated as different source code regions are selected, depending on the metrics available in the current PDF file.

Independently scroll Region, Thread, and Metric axes

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.

3D Profile window

In the 3D profile window, you can

- Dynamically select different regions and metrics to graph.
- Click with the left mouse button on any bar in the graph to display source code associated with the corresponding region.
- Save 3D profile graphs in PostScript, xwd, or ASCII file formats (select the Save Profile option from the File menu).
- View or specify the range of metric values displayed or the number of regions or threads displayed at one time in the 3D graph using the Zoom dialog or buttons along the bottom of the window. This can be useful when there are a large number of data items to graph and you want to focus on a subset of the data.

  To access the Zoom dialog, select the Zoom option from the View menu.
- View exact data values for regions in the graph by moving the mouse over regions in the graph. Data values are displayed in the upper left corner of the graph.
- Use the scrollbars in the Axis Display Controls panel to scroll the Region, Thread, and Metric axes independently.
- Sort data in 3D profile graphs alphabetically, by load order, from lowest to highest, or from highest to lowest. The default sort order is from highest to lowest. To access sort options, select the Sort option from the View menu.
- Rotate the graph by placing the mouse pointer over the graph and holding down the middle mouse button.
  - To restrict the rotation to a single axis, press the **x**, **y**, or **z** key after pressing the mouse button, but before moving the mouse to rotate the graph.
  - To return the graph to its original position, press the Reset Zoom button.

By default, CXpa graphs metric values for all selected regions or for the maximum number of regions for which labels can legibly be displayed, whichever is less. If all selected regions or threads are not displayed, use the Region or Thread Axis Display Control scrollbars to navigate the graph or use one of the other zoom options to change the graph display.

Menus

Name	Meaning
File	Contains options for saving the graph in a PostScript, ASCII, or xwd file format 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 (Sort) and changing the range of values, number of source code regions, or number of threads displayed in the 3D graph (Zoom).
Options	Contains an option for including or excluding values for called routines and/or inner loops in the 3D graph.
Help	Contains options for invoking the CXpa help system.

Regions

Contains buttons that change the type of program region to graph on the Region axis.

Region level option	Action
Routines	Graph selected metric for routines.
Loops (serial)	Graph selected metric for serial loops.
Parallel (parallel)	Graph selected metric for parallel loops.
Basic Blocks	Graph selected metric for basic blocks.
Select Regions	Option menu containing two items: **All**—Graph profiling data for all regions of the type specified above. **Subset**—Brings up the Region Subset Selection dialog where you can specify a customized list of routines to graph profiling data for.

Metrics

Contains a scrolled list of metrics available in the current PDF where you can select the type of metric graphed on the Metric axis.

The available metric choices depend on the regions selected for profiling and the metrics that were collected for the run of your program that produced the PDF file.

Refer to the "Introducing metrics" online help topic or section of this book for metric descriptions and a list of metrics available by architecture.

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.

Axis Display Controls

Using the scrollbars in the Axis Display Controls panel, you can scroll the Region, Thread, and Metrics axes independently.

Buttons

Name	Meaning
Reset Zoom	Restores the default graph display. By default, CXpa graphs metric values for all selected regions or for the maximum number of regions for which labels can legibly be displayed, whichever is less. If all selected regions or threads are not displayed, use the Region or Thread Axis Display Control scrollbars to navigate the graph or use one of the other zoom options to change the graph display.
Zoom-In	Reduces the number of regions displayed in the 3D graph at one time by half. The number of threads displayed remains constant.
Zoom-Out	Doubles the number of regions displayed in the 3D graph at one time. The number of threads displayed remains constant.
Show All	Displays the entire graph. If the number of regions or threads is so large that the region labels cannot be displayed legibly, labels are not displayed. If this happens, click the Reset Zoom or Zoom-In button to restore the labels and use the scrollbars to navigate the graph.

Context

The 3D Profile window appears when you

- Press the Create 3D Profile button in the Analysis Control 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` or specified `-windows 3DProfile` when invoking CXpa on the command line.

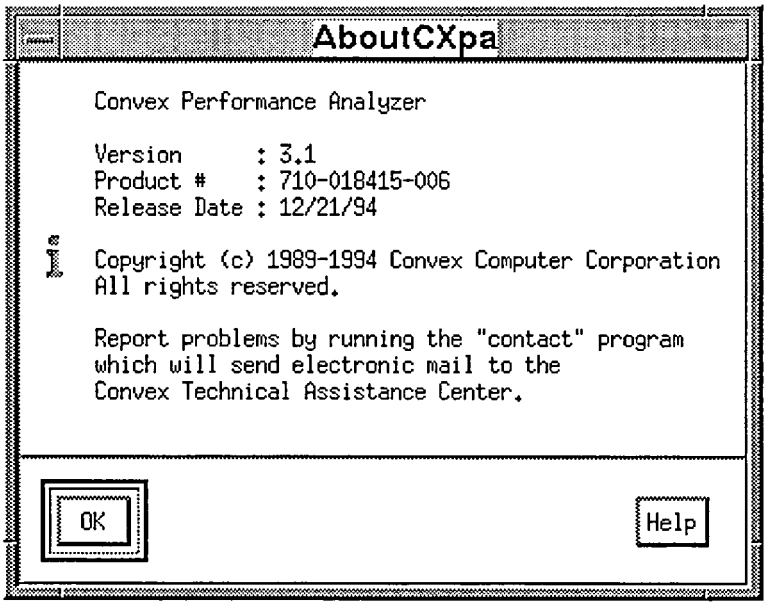
Related Topics

Introducing metrics

Related Windows

2D Profile window
Analysis Control window
Analysis Report window
Executable Manager window
Filter Profile dialog
Profile Selection dialog
Region Subset Selection dialog
Save Profile dialog
Sort dialog
Visibility Selection dialog
Zoom dialog

About CXpa dialog



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

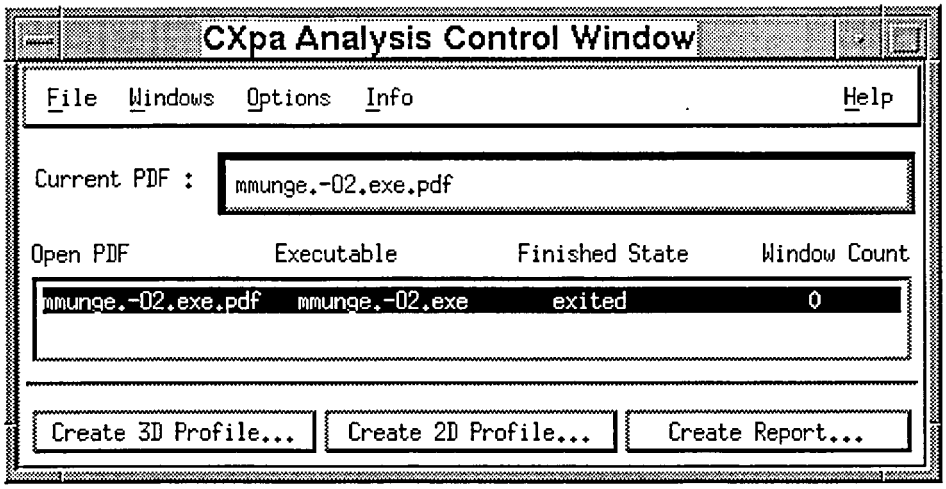
Buttons

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 by selecting the Source Code item from the Windows menu.

Invoking CXpa with a PDF only

If you only 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: `All`, `None`, `2DProfile`, `3DProfile`, `Report`, or `Source`. The default is `None`.

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, bring up the Open PDF dialog by selecting the Open PDF item from the File menu. When you have finished making your selection, 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 in the Analysis Control window. 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 the file a.out.pdf in the current directory and displays the Analysis Control window.

Related Topics

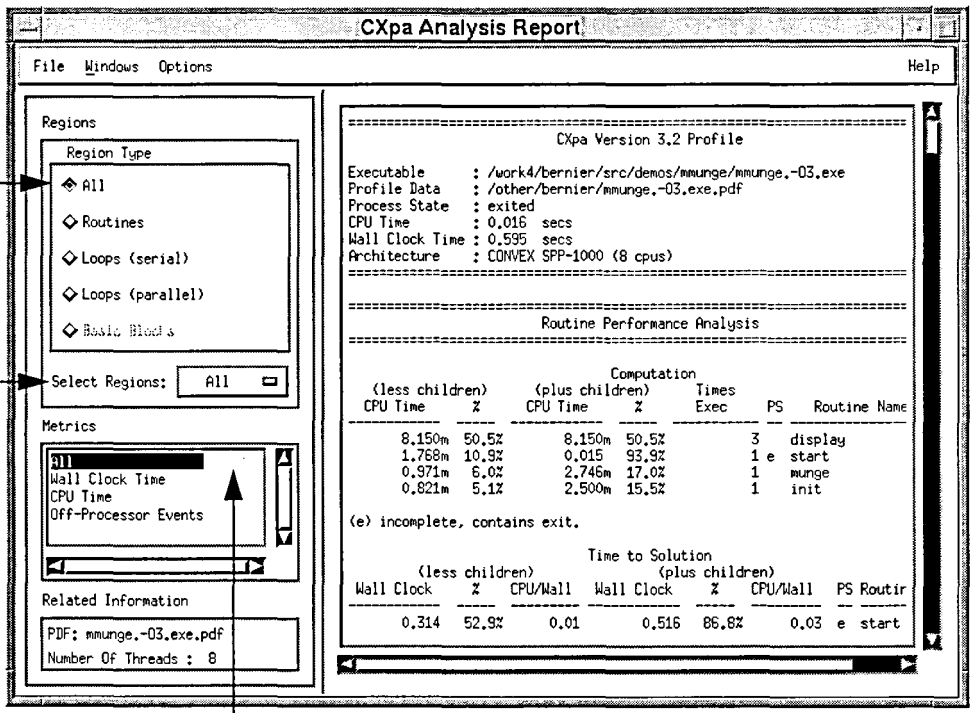
Analyzing PDFs only

Setting the PDF

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

Analysis Report window



Select the type of region for which you you want to view profile reports

Select all routines or a subset of routines containing regions of the type selected above

Select the type of metrics to view in profile reports. The list is updated as different source code regions are selected, depending on the metrics available in the current PDF file.

Description

The Analysis Report window displays 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 Region Type section and then highlighting one of the metrics selections in the scrolled list. You can also create customized reports by specifying a subset of routines that contain regions of the currently selected type (refer to the "Region Subset Selection dialog" online help topic or section in this book for more information).

You can select whether you want to display performance data in reports on a per-process basis (the default) or on a per-thread basis, by selecting the Filter Report option from the Options menu. Threaded data in reports 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," and "Basic Block report."

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 on a per-process basis (the default) or on a per-thread basis.
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 Type option	Action
All	Report selected metrics for all profiled regions.
Routines	Report selected metrics for routines.
Loops (serial)	Report selected metrics for serial loops.
Loops (parallel)	Report selected metrics for parallel loops.
Basic blocks	Report selected metrics for basic blocks.

Select Regions	Contains an option menu with two items: **All**—Report profiling data for all regions of the type specified above. **Subset**—Bring up the Region Subset Selection dialog where you can specify a subset of routines that contain regions of the currently selected type to include profiling data for in reports.

Metrics

Contains a scrolled list of metrics available in the current PDF where you can select the type of metrics you want to view in CXpa profile reports.

The available metric choices depend on the regions selected for profiling and the metrics that were collected for the run of your program that produced the PDF file.

Refer to the "Introducing metrics" online help topic or section of this book for metric descriptions and a list of metrics available by architecture.

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.

Context

The Analysis Report window appears when you

- Press the Report button on the Executable Manager window or the Create Report button on the Analysis Control window.
- Select the Report 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 `Report` or specified `-windows Report` when invoking CXpa from the command line.

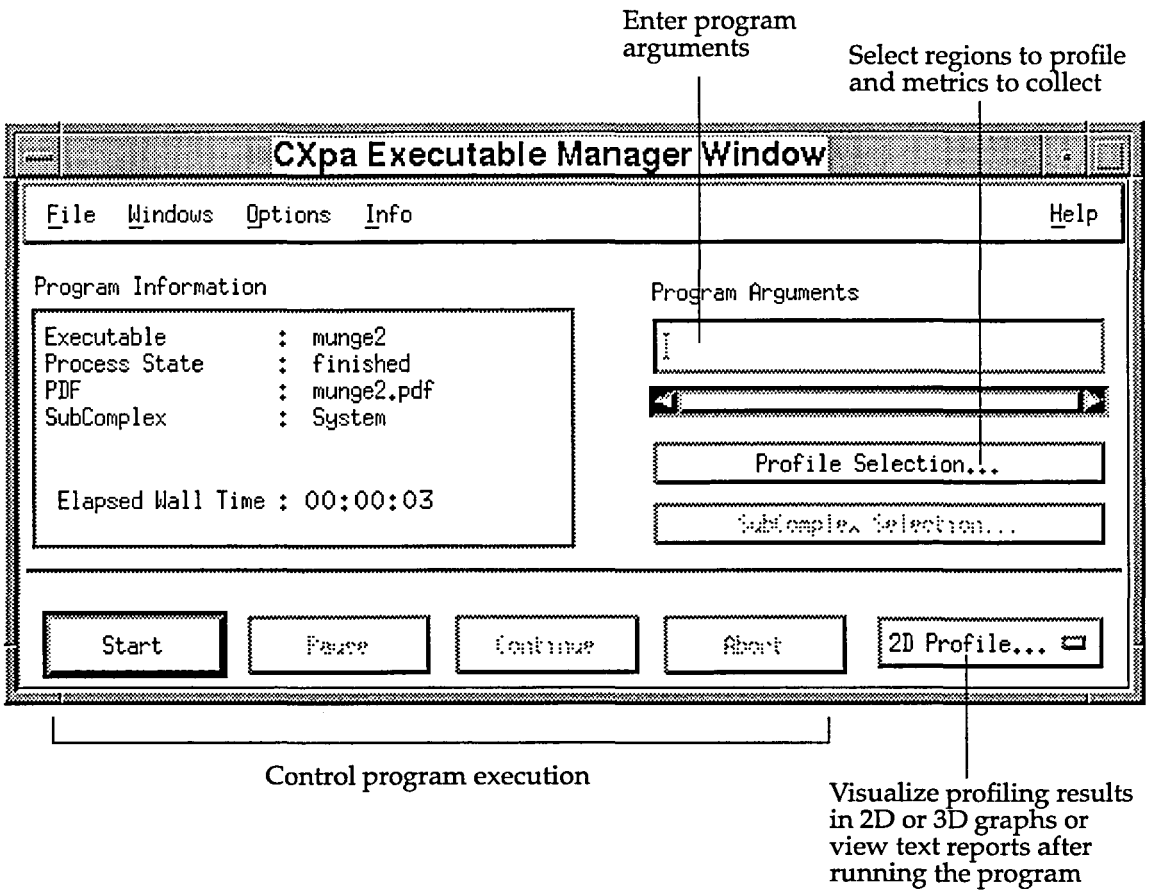
Related Windows

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

Related Topics

Basic Block report
Loop reports
Report fields
Routine reports
Introducing metrics
Parallel Region reports
Reports
Selecting metrics in X window mode

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 compiler options for CXpa (`-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, wall clock time, and execution counts for all routines in your program. Use these defaults the first time you profile your program under CXpa. 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, wall clock 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 systems, 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(s) option 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 in a standalone environment (on a "quiet" or dedicated subcomplex).

   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 option 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 item from the Windows menu.

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

Menus

Name	Meaning
File	Contains items to choose a new PDF or exit CXpa.
Windows	Contains items for creating additional windows for viewing 2D and 3D profile graphs, performance reports, or source files.

Options	Contains items 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 items that display information about the PDF, the executable being profiled, and your current CXpa session.
Help	Contains various items 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.
SubComplex	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	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 option 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 the file a.out in the current directory.

Related Windows

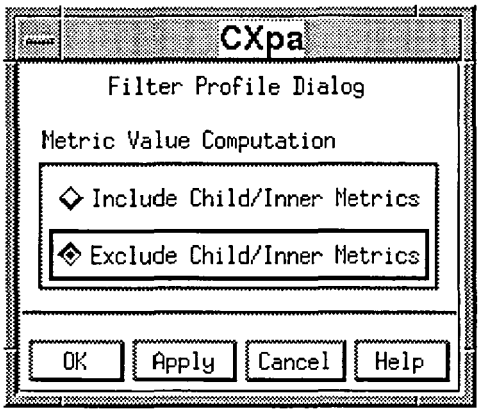
2D Profile window
Analysis Control window
Info Executable dialog
New PDF dialog
Profile Selection dialog
Source Search Path dialog
Visibility Selection dialog
3D Profile window
Analysis Report window
Info PDF dialog
Info Session dialog
Source Code window
Subcomplex Selection dialog

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 dialog



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, closes the dialog, and updates the graphs displayed in 2D and 3D Profile windows.

Filter Profile dialog

Apply	Applies the new settings without closing the dialog, and updates the graphs displayed in 2D and 3D Profile windows.
Cancel	Closes this dialog without making any changes.
Help	Displays a help page for this dialog.

Context

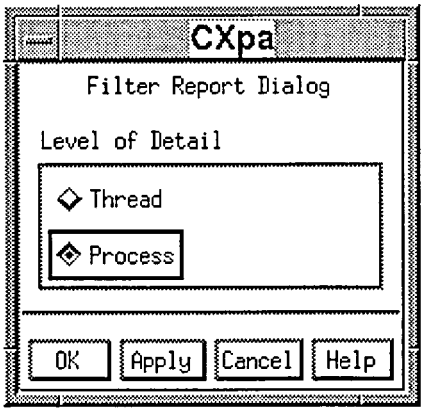
The Filter Profile dialog appears when you choose the Filter Profile item from the Options menu on the 2D Profile window or the 3D Profile window.

Related Windows

3D Profile window

2D Profile window

Filter Report dialog



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.

Filter Report dialog

Cancel	Closes this dialog without making any changes.
Help	Displays a help page for this dialog.

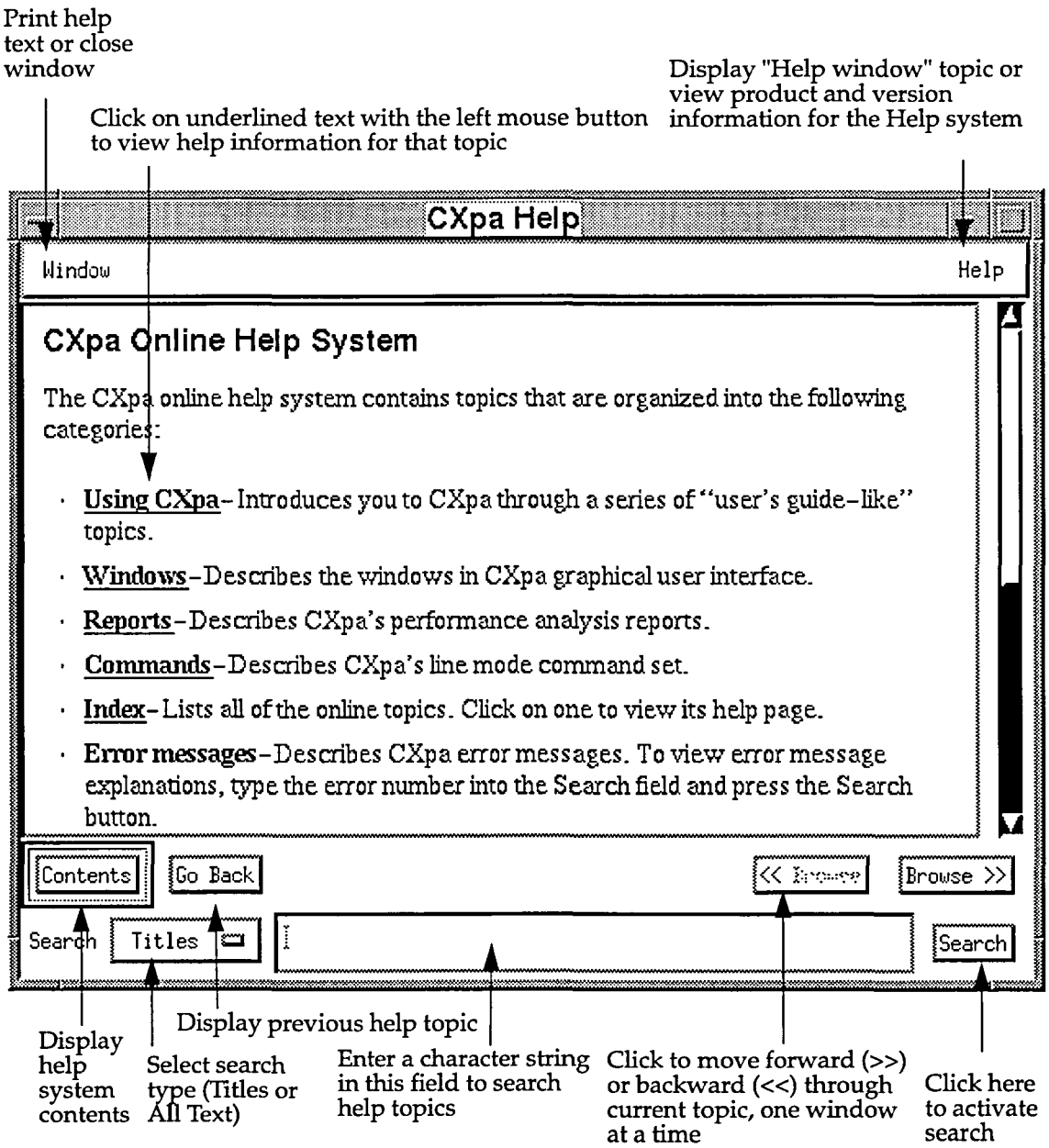
Context

The Filter Report dialog appears when you choose the Filter Report item from the Options menu on the Analysis Report window.

Related Windows

Analysis Report window
Region Subset Selection 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
<< Previous
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

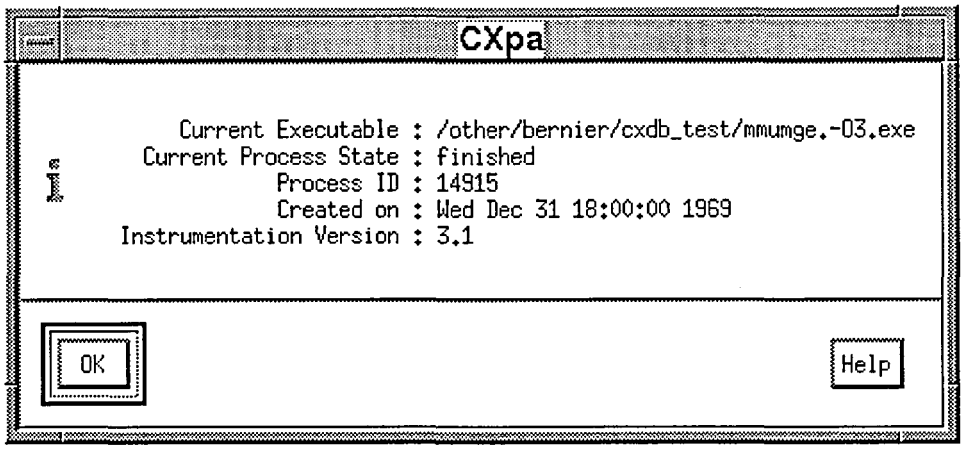
- Select the Window Overview, CXpa Overview, Using Help, or Tutorial item from the Help menu in the upper right corner of any CXpa window.
- Click the Help button on any CXpa dialog

Related Topics

Using online help

Help window

Info Executable dialog



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 dialog

Buttons

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

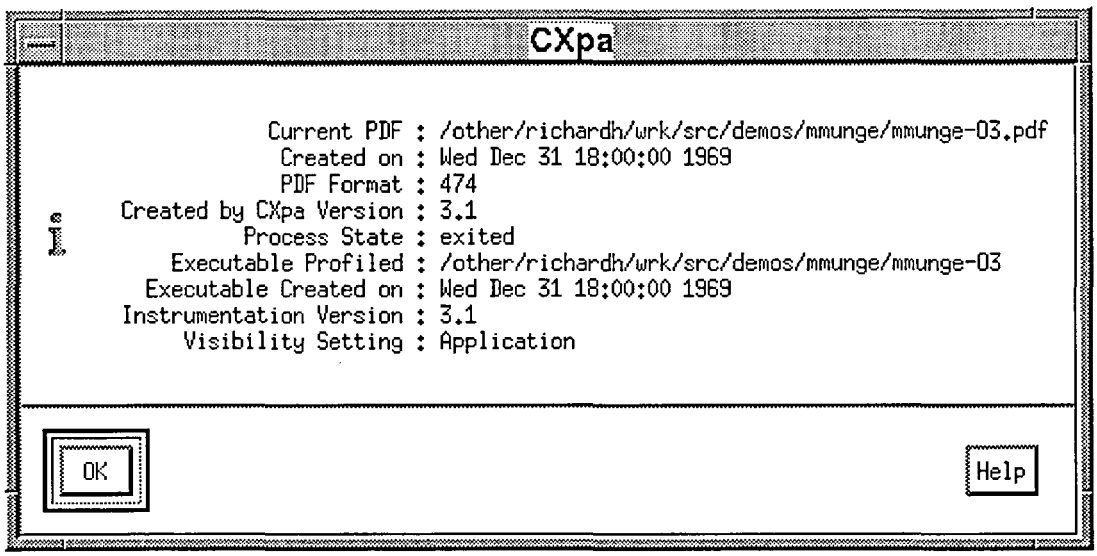
Context

The Info Executable dialog appears when you select the Executable item 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 dialog



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.

Info PDF dialog

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.

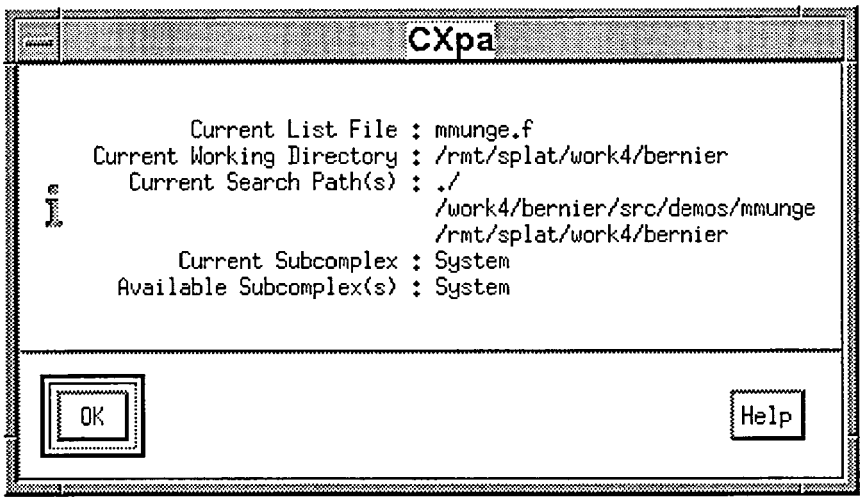
Context

The Info PDF dialog appears when you select the PDF item 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 dialog



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.
Current Subcomplex	Lists the name of the subcomplex that your process is set to run on.
Available Subcomplex(s)	Lists available subcomplexes on your system.

Info Session dialog

Buttons

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

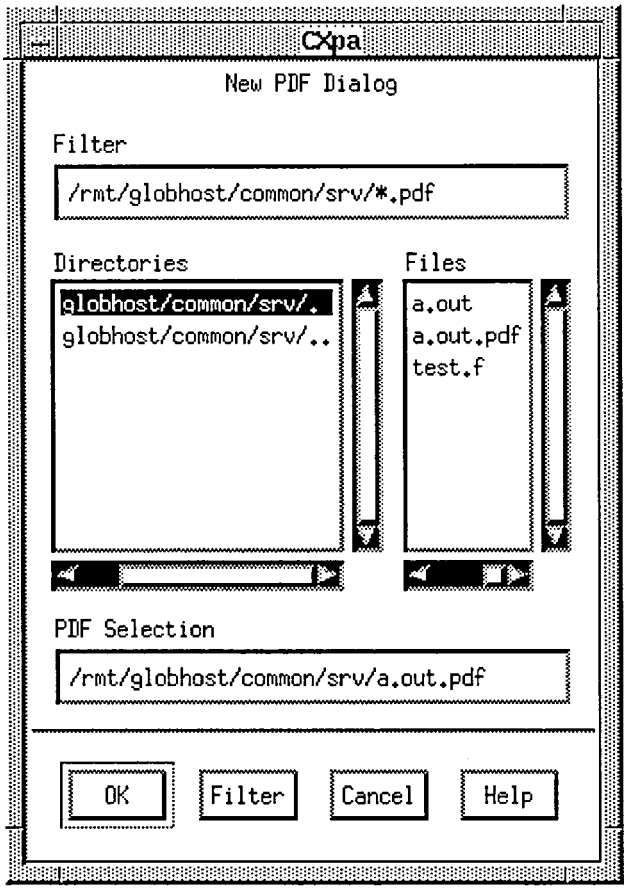
Context

The Info Session dialog appears when you select the Session item from the Info menu in the Executable Manager or Analysis Control window.

Related Windows

Analysis Control window
Executable Manager window
Info Executable dialog
Info PDF dialog
Subcomplex Selection dialog

New PDF dialog



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 item 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

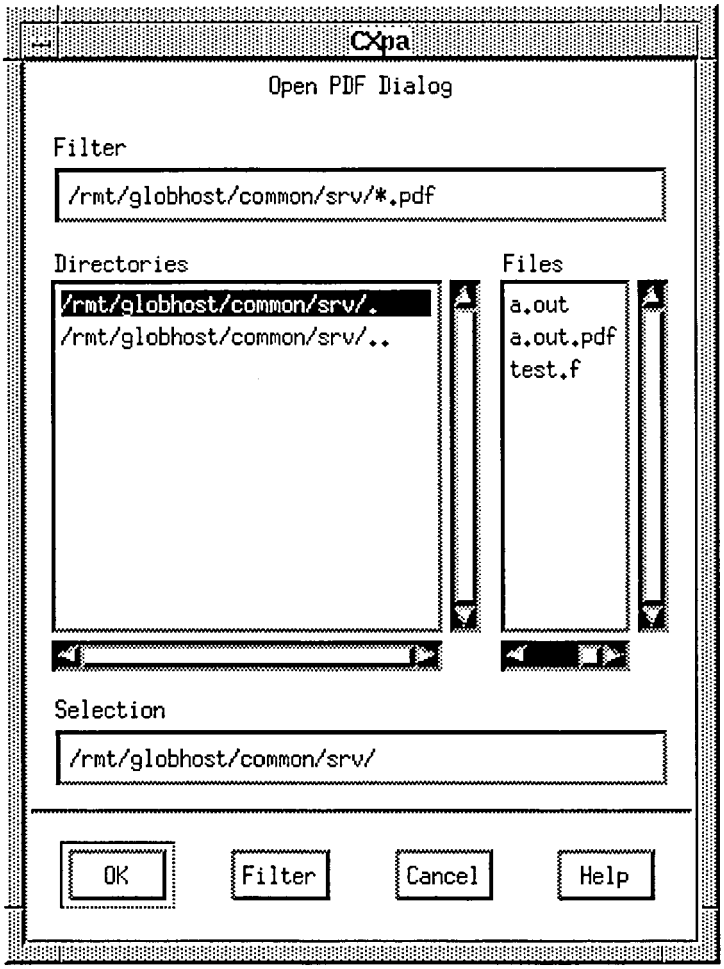
Analysis Control window
Executable Manager window
Filter Profile dialog
Open PDF dialog

Related Topics

Analyzing PDFs only
Setting the PDF

New PDF dialog

Open PDF dialog



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, perform the following steps:

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 using any one of the following methods:

- 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 item from the File menu on the Analysis Control window.

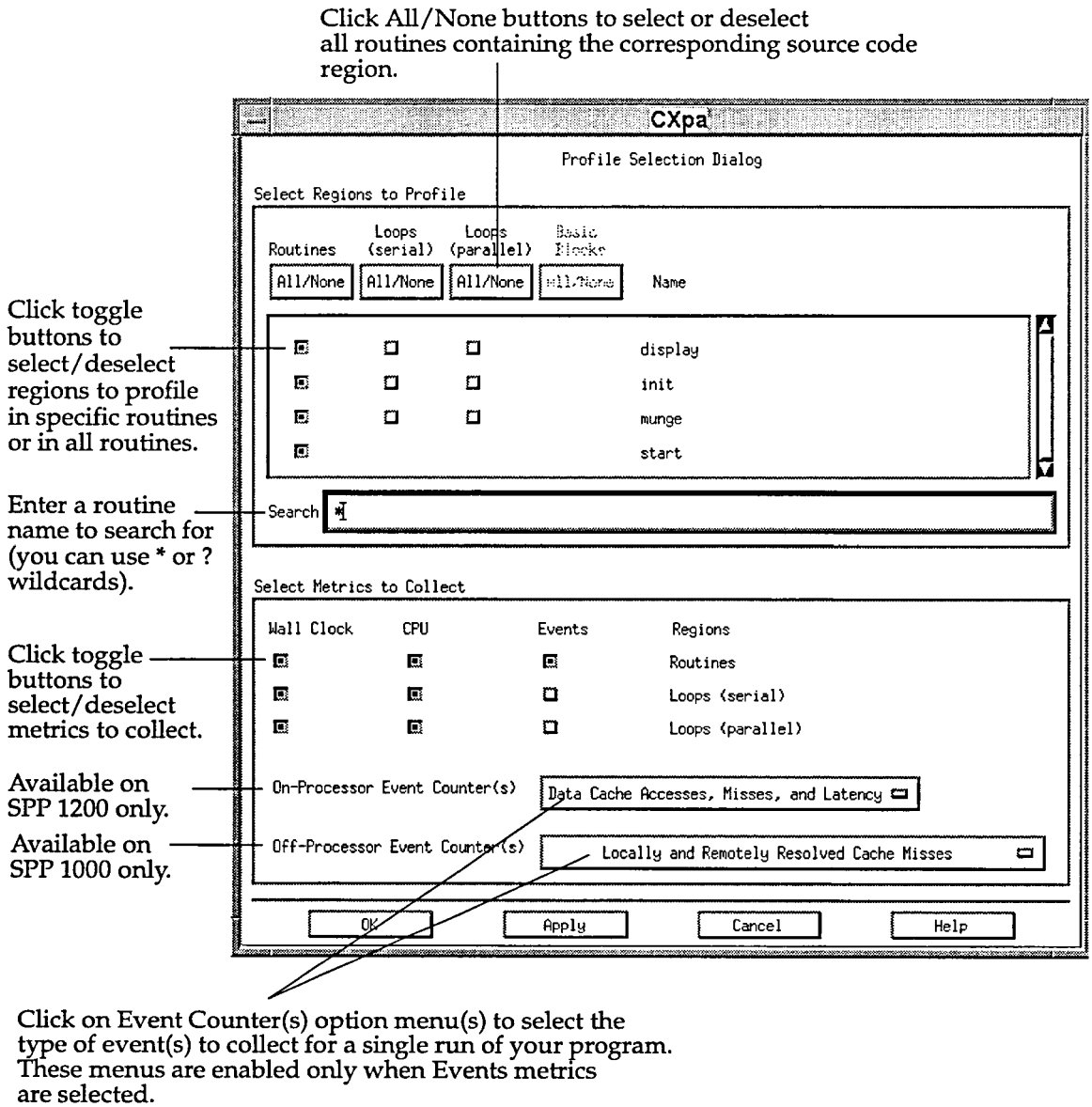
Related Windows

Analysis Control window
Filter Profile dialog
Info PDF dialog
New PDF dialog

Related Topics

Analyzing PDFs only
Setting the PDF

Profile Selection dialog



Description

Use the Profile Selection dialog to select or deselect source code regions for profiling and to specify the types of metrics (including events) 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.

Recommended guidelines for selecting regions and metrics

NOTE: Selecting all source code regions in all routines for profiling is not recommended, due to increased profiling time and the amount of profiling intrusion that may be incurred.

NOTE: 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.

Use the following guidelines for selecting regions to profile:

- The first time you profile your program under CXpa, select all routines in your program for profiling at the routine region level and collect wall clock and CPU time metrics.

  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 that consume the most wall clock time or CPU time, select only those routines for profiling at the loop or parallel region level, and then rerun your program under CXpa.

  On subsequent runs of your program, you can also choose to collect different metrics, such as cache misses and latency. With fewer source code regions selected, less time is spent in the timing routines CXpa uses to collect performance data.

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 (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

   If choose to collect Events, you must specify the type of event or events you want to collect as described below in step 4. Otherwise, continue with step 5.

4. If you have chosen Events as one of the metrics in the Select Metrics to Collect section of the Profile Selection dialog, the On-Processor or Off-Processor Event Counter(s) option menus are now available for you to select an event type or types.

   Click and hold down the left mouse button on the Event Counter(s) 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.

   The number and type of events you can select vary according to machine architecture:
   - Refer to the "Introducing metrics" online help topic or section in this book for a discussion of available event metrics for SPP 1000 and SPP 1200 systems.
   - Refer to the "Selecting Metrics in X window mode" online help topic or section in this book for more information about selecting events.

5. 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 specific routines

To select specific region types for profiling in specific 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
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 available On-Processor or Off-Processor Event Counter(s) option menus. (These menus are not enabled until you enable event collection.) Refer to the "Introducing metrics" and "Selecting Events in X window mode" online help topics or topics in this book for more information.
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.

Option menus

On-Processor Event Counter(s) (SPP 1200 only)

Off-Processor Event Counter(s) (SPP 1000 only)

Displays an option menu where you can select the type of hardware event to collect. The number and types of events you can select differ between SPP 1000 and SPP 1200 Series systems.

Refer to the "Introducing Metrics" and "Selecting metrics in X window mode" online help topics or sections in this book for descriptions of on-processor and off-processor hardware 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, serial loops, parallel loops, or basic blocks).
OK	Applies any changes you have made and closes this dialog.

Profile Selection 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.

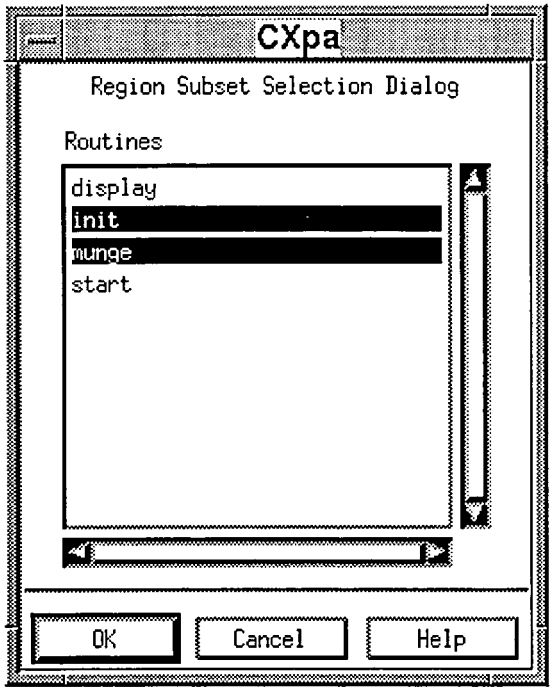
Related Windows

Analysis Report window
Executable Manager window
Visibility Selection dialog

Related Topics

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

Region Subset Selection dialog



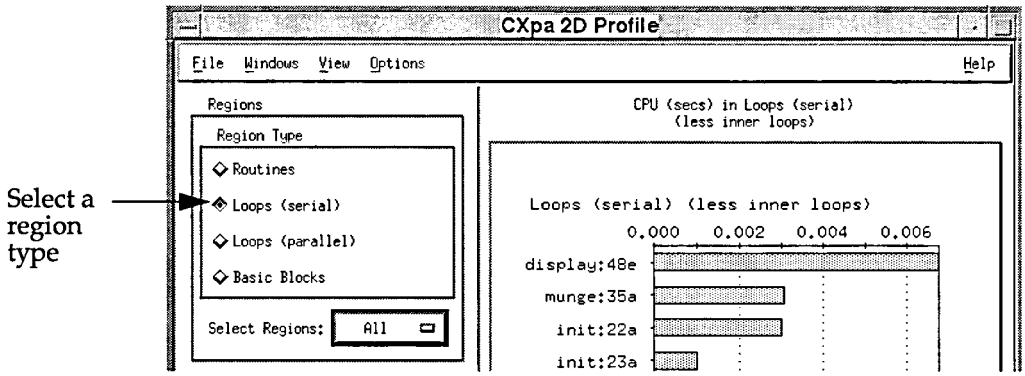
Description

The Region Subset Selection dialog allows you to create a customized report, 2D profile graph, or 3D profile graph by specifying a subset of routines that contain regions of the currently selected region type (routine, serial loop, parallel loop, or basic block) for analysis.

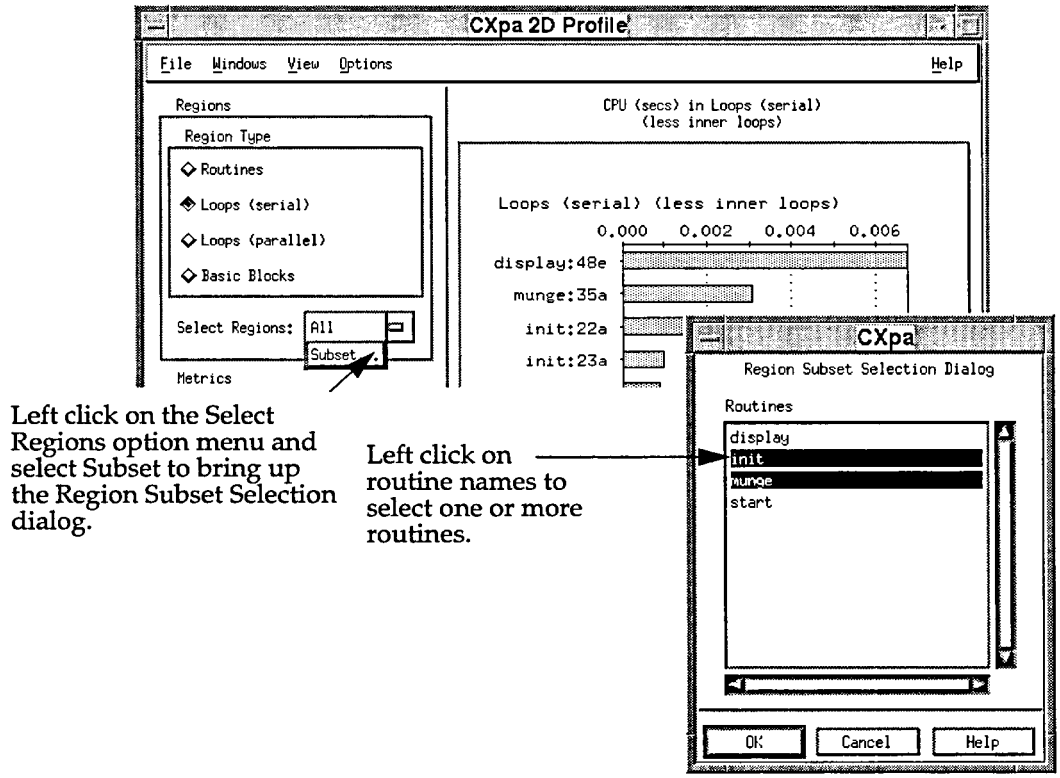
To create a customized profile graph or report

1. In the 2D Profile, 3D Profile, or Analysis Report window, make sure you have selected the region type for which you want to customize a profile. You can select only one.

Region Subset Selection dialog



2. Select the Subset item from the Select Region options menu to bring up a Region Subset Selection dialog with a scrolled list of routines containing profiled regions at the selected region level.



3. Select the routines that you want to include in the graph or report by clicking on them with the mouse. You can select multiple routines.
4. Press the OK button to display the new graph or report.

   If you press OK without selecting any profile regions, CXpa displays a info dialog displaying the message "`INFO A70: No regions selected for a customized 2D or 3D profile.`" Close the info dialog, then either select regions to profile or click Cancel to dismiss the Region Subset Selection dialog.

Fields

Heading	Meaning
Routines	Contains a scrolled list of routines in your program that contain regions of the currently selected type. Left click with the mouse on one or more routines in the list to select or deselect them for graphing or inclusion in text reports.

Buttons

Name	Action
OK	Accepts the changes you have made, closes the dialog, and applies the changes to the graph.
Cancel	Does not apply any of the changes you have made and closes the dialog.
Help	Displays a help page for this dialog.

Context

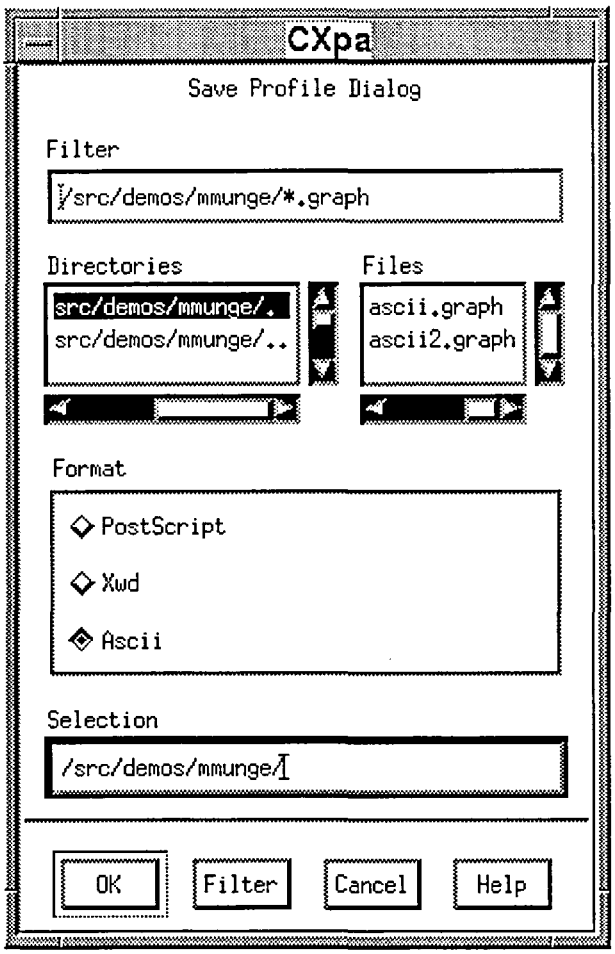
The Region Subset Selection dialog appears when you select the Subset option from the Select Regions option menu in the Regions panel in the 2D Profile, 3D Profile, or Analysis Report window.

Related Windows

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

Region Subset Selection dialog

Save Profile dialog



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. Only the area of the graph that is visible in the window is saved to the file. These files are immediately available for use by other utilities that accept these formats, as shown in the following table.

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.

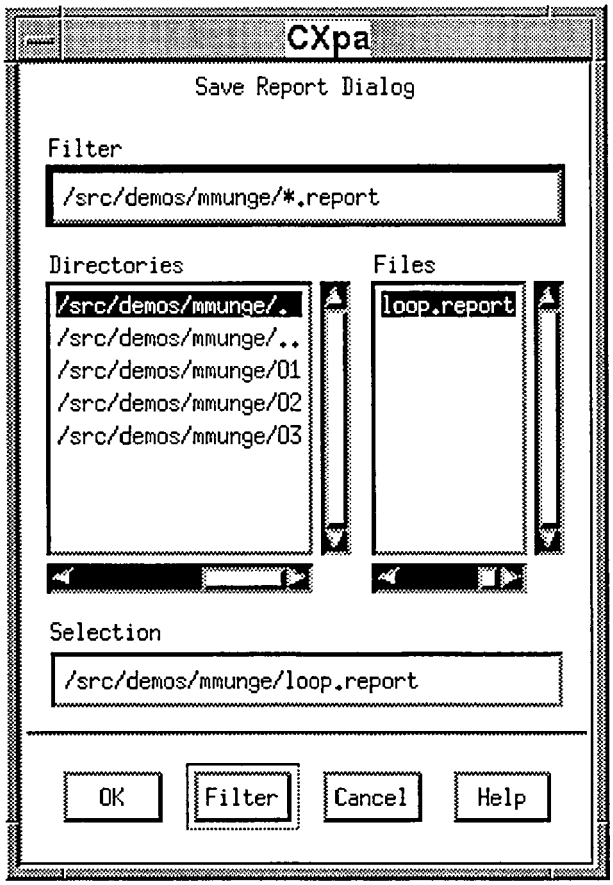
Context

The Save Profile dialog appears when you select the Save Profile item from the File menu on the 2D Profile or 3D profile windows.

Related Windows

2D Profile window
Filter Profile dialog
3D Profile window
Region Subset Selection dialog

Save Report dialog



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.

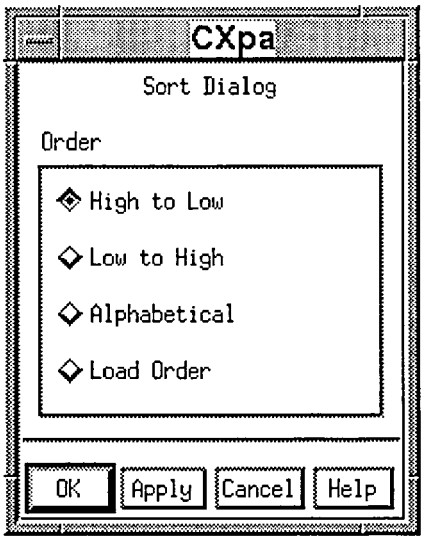
Context

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
Filter Report dialog
Region Subset Selection dialog

Sort dialog



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.

High to Low	Sorts the data from highest to lowest (default).
Low to High	Sorts the data from lowest to highest.
Alphabetical	Sorts the data alphabetically.
Load order (default)	Lists the data in the order that the routines were given to the link editor.

Sort dialog

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.

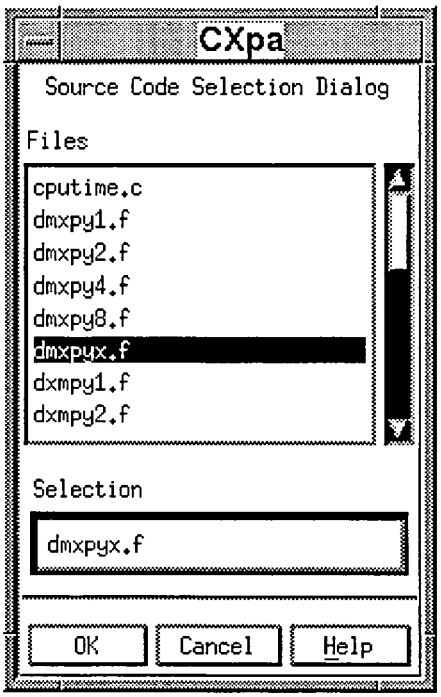
Context

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
Region Subset Selection dialog
Zoom dialog

Source Code Selection dialog



Description

The Source Code Selection dialog allows you to choose which source file to display in the Source Code window.

Changing source files

Use the following procedure to change the source file displayed 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.

Source Code Selection dialog

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 selected for 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

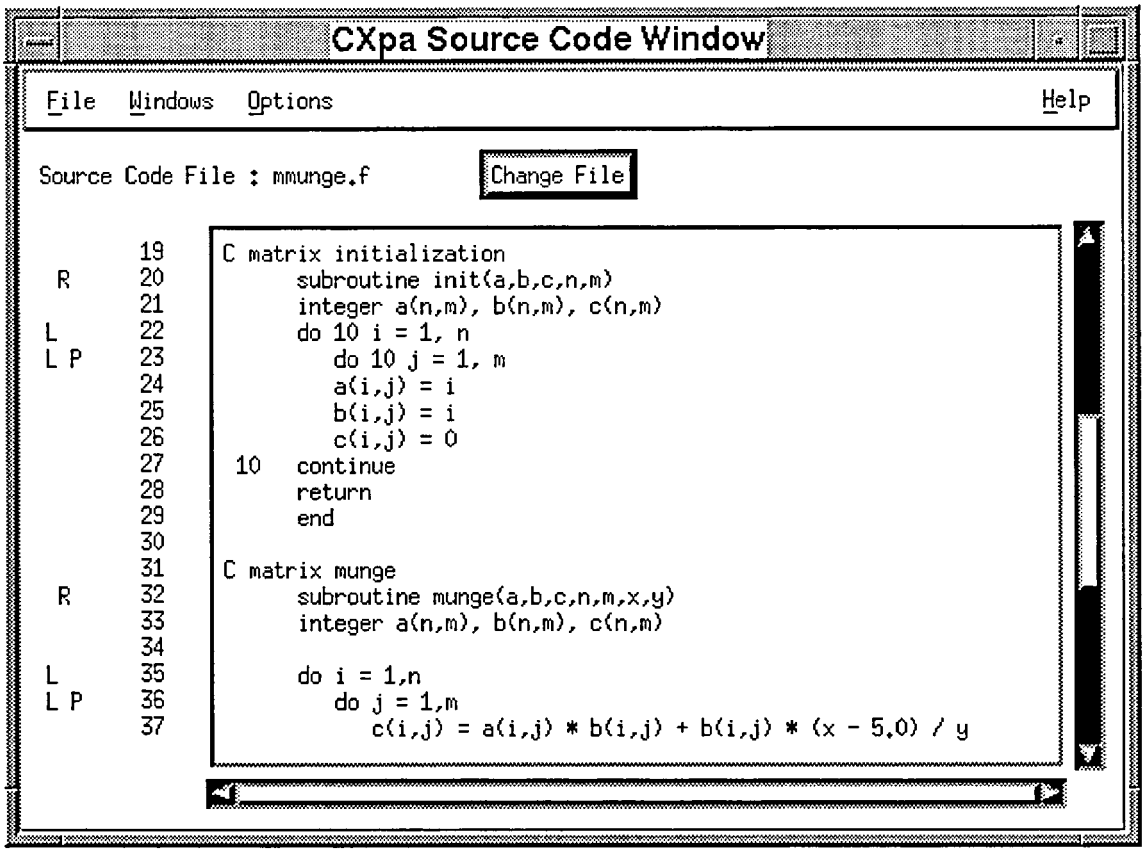
The Source Code Selection dialog appears when you press the Change File button on the Source Code window.

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

Use the following procedure to change the source file displayed 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 item to close the window.
Windows	Contains items to open additional CXpa windows for viewing 2D and 3D profile graphs, performance reports, or source files.
Options	Contains an item for changing the search path CXpa uses to locate source code files (Source Search Path).
Help	Contains various items that invoke the CXpa help system.

Buttons

Name	Meaning
Change File	Displays a Source Code Selection dialog where you can choose another source file to display in the Source Code window.

Context

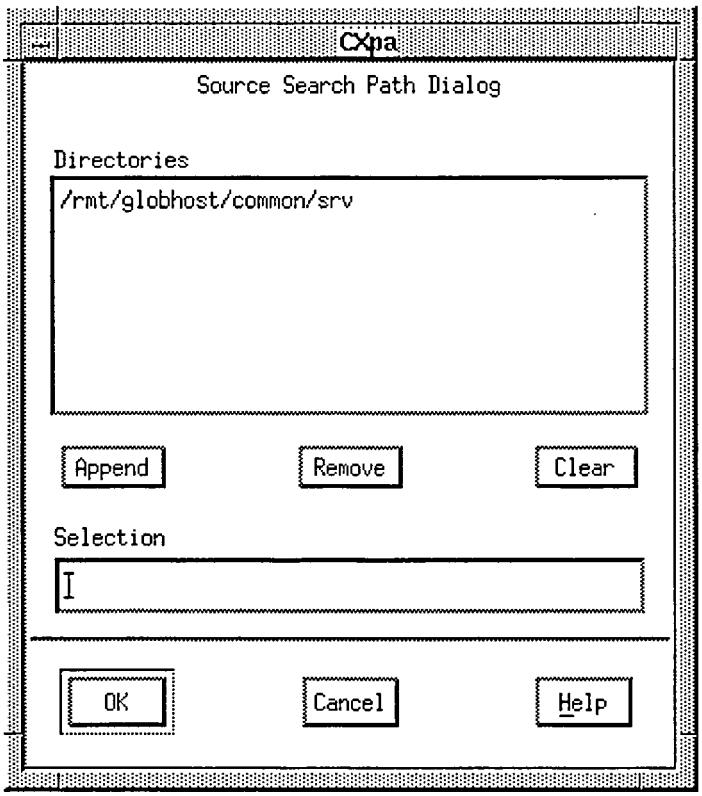
The Source Code window appears when you

- Select the Source Code item from the Windows menu in any CXpa window.
- Click on a bar in the graph of the 2D or 3D Profile windows.
- 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 `Source` or specified `-windows Source` when invoking CXpa from the command line.

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 Search Path dialog



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

Perform the following steps 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

Perform the following steps 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.

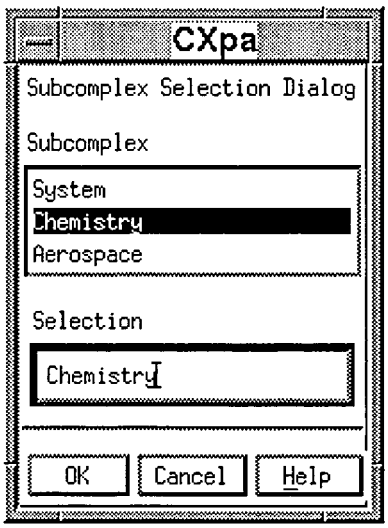
Context

The Source Search Path dialog appears when you select the Source Search Path item 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

Subcomplex Selection dialog



Description

The Subcomplex Selection dialog allows you to 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. Perform the following steps 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 (the subcomplex your process is set to run on).

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

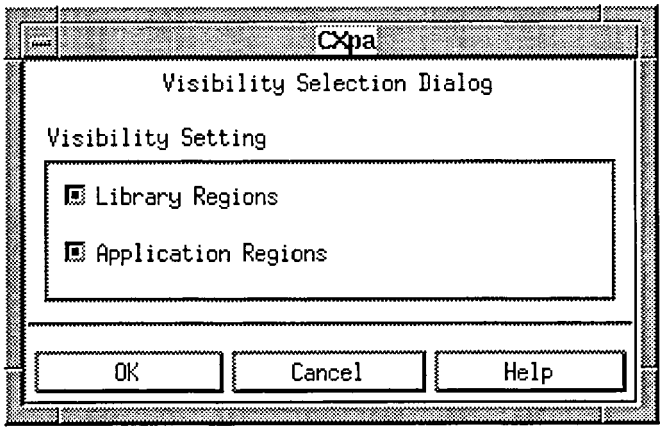
The Subcomplex Selection dialog appears when press the Subcomplex Selection button on the Executable Manager window. The Subcomplex Selection button is not available if there is only one subcomplex configured on your system.

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 Info Session dialog

Visibility Selection dialog



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 dialog

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.

Perform the following steps 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 item from the Options menu on the Executable Manager window or the Analysis Control window.

Related Windows

2D Profile window
3D Profile window
Analysis Control window
Analysis Report window
Executable Manager window
Profile Selection dialog
Region Subset Selection dialog
Source Search Path dialog

Xdefaults

Description

Default X resource settings (Xdefaults) for the X Window System interface of CXpa are specified in the file /usr/lib/X11/app-defaults/Cxpa.

To modify these resource settings, copy the default specifications into the file that contains your own X resource specifications (usually .Xdefaults or .Xresources). Then, modify the specifications to suit your needs.

After modifying the resource specifications, you must enter the following command in your xterm window:

xrdb -merge ~/*resource_file*

resource_file is the name of the file that contains your resource specifications (usually .Xdefaults or .Xresources).

NOTE: If any of these resource specifications conflict with the settings used by your window manager, the window manager may override your CXpa resource specifications.

The Xdefaults in the /usr/lib/X11/app-defaults/Cxpa file are organized into the following categories:

- Generic resources
- Text input field resources
- 2D and 3D profile graph resources
- Session behavior resources

CXpa also supports standard X/Motif general resource specifications. Refer to your X Window System documentation for more information.

Generic application resources—These resources define basic properties for all CXpa windows:

`Cxpa*fontList: <font>`

NOTE: Displaying a 2D profile with a large font specified in an application X resource can result in an incorrectly sized initial 2D profile display. This can also occur when no data is collected for the selected metric. To work around the problem, resize the 2D Profile window manually, then click the "Show All" button, or specify a smaller, fixed font.

```
Cxpa*background: <color>
Cxpa*foreground: <color>
Cxpa*selectColor: <color>
```

```
Cxpa*highlightColor: <color>
```

Text input field resources—The following resources define foreground and background color and keyboard editing functions for CXpa text input fields:

```
Cxpa*XmText.background: <color>
Cxpa*XmText.foreground: <color>

Cxpa*XmTextField.background: <color>
Cxpa*XmTextField.foreground: <color>

Cxpa*XmText*translations: #override\
      Ctrl<Key>a: beginning-of-line() \n\
      Ctrl<Key>b: backward-character() \n\
      Ctrl<Key>d: delete-next-character() \n\
      Ctrl<Key>e: end-of-line() \n\
      Ctrl<Key>f: forward-character() \n\
      Ctrl<Key>h: delete-previous-character()\n\
      Ctrl<Key>k: delete-to-end-of-line() \n\
      Ctrl<Key>u: delete-to-start-of-line() \n\
      Meta<Key>b: backward-word() \n\
      Meta<Key>f: forward-word()

Cxpa*XmTextField*translations: #override\
      Ctrl<Key>a: beginning-of-line() \n\
      Ctrl<Key>b: backward-character() \n\
      Ctrl<Key>d: delete-next-character() \n\
      Ctrl<Key>e: end-of-line() \n\
      Ctrl<Key>f: forward-character() \n\
      Ctrl<Key>h: delete-previous-character()\n\
      Ctrl<Key>k: delete-to-end-of-line() \n\
      Ctrl<Key>u: delete-to-start-of-line() \n\
      Meta<Key>b: backward-word() \n\
      Meta<Key>f: forward-word()
```

2D and 3D graph resources—These resources define foreground and background color for CXpa 2D and 3D Profile window graphs:

```
Cxpa*xrtBackgroundColor: <color>
Cxpa*xrtForegroundColor: <color>
Cxpa*xrt3dBackgroundColor: <color>
Cxpa*xrt3dForegroundColor: <color>
```

Session behavior resources—These resources define auto-positioning behavior for xterms CXpa creates and automatic window creation behavior for graph, source code, and report windows:

`Cxpa*autoPosition:` *<Boolean>*

By default, auto-positioning is enabled (`True`). Set this resource to `False` if you are running a virtual window manager (such as `tvtwm`) and you find that CXpa positions xterms that it creates (such as the process interface window) in unexpected locations (i.e., not in the current virtual root window).

`Cxpa*defaultWindow:` *<windows>*

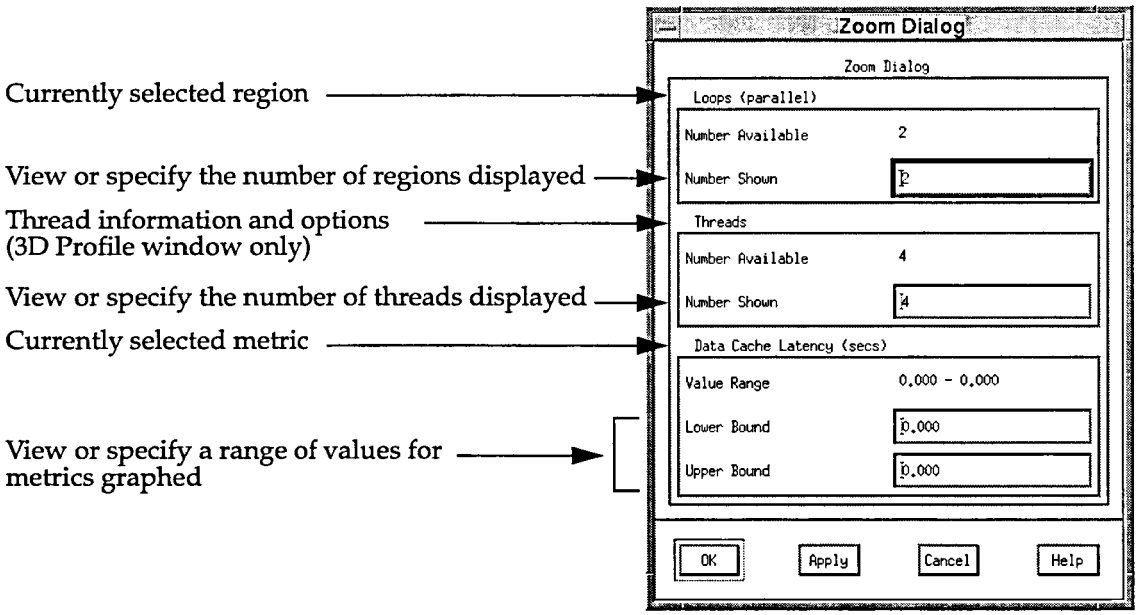
Use this resource to specify the window or windows that are created automatically when you invoke CXpa in analysis mode (without specifying an executable). You can specify multiple windows. Separate each value with a space. The default is `None`. Valid values for *<windows>* are as follows:

- `2DProfile`—Automatically create a 2D Profile window.
- `3DProfile`—Automatically create a 3D Profile window.
- `All`—Automatically create a 2D Profile window, 3D profile window, Analysis Report window, and Source Code window in analysis mode.
- `None`—Disable automatic creation of windows in analysis mode (this is the default).
- `Report`—Automatically create an Analysis Report window
- `Source`—Automatically create a Source Code window.

Related Windows

2D Profile window
3D Profile window
Analysis Control window
Analysis Report window
Executable Manager window
Source Code window

Zoom dialog



Description

Use the Zoom dialog to view or specify the range of metric values graphed, the number of regions displayed, or the number of threads displayed along a given axis in a 2D or 3D graph. 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.

Labels in the Zoom dialog indicated the currently selected region level and metric type and are updated as different metrics and regions are selected in the associated 2D or 3D Profile window.

2D graph zoom options

The following zoom options are available for the 3D graph:

- To change the number of source code regions displayed in the current window, enter a new value in the Number Shown field for currently selected metric.

- To specify a different range or limit the range of metric values graphed along the Metric (horizontal) axis, enter a new value in the Upper- and/or Lower-bound fields for the currently selected metric.

3D graph zoom options

The following zoom options are available for the 3D graph:

- To change the number of source code regions graphed along the Region axis, enter a new value in the Number Shown field for the currently selected region.
- To change the number of threads graphed, enter a new value in the Number Shown field in the Threads panel of the dialog.
- To specify a different range or limit the range of data values (metrics) graphed, enter a new value in the Upper- and/or Lower-bound fields for the currently selected metric.

Fields

Field	Description/Valid values
Value Range	Displays the range of data values (metrics) that can be shown in the graph.
Lower Bound	Sets the lower-bound of the range for the specified 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 range for a given axis. This value must be greater than the value specified in the Lower Bound field but can be larger than the maximum data value.
Available	Displays number of available data items (regions or threads) displayed in the graph.
Number Shown	Shows the number of items (regions or threads) currently displayed. Valid values are integers from one to the number available. Values larger than the number available are ignored.

Buttons

Name	Action
OK	Updates the graph display according to the selected values and closes this dialog.
Apply	Updates the graph display according to the selected values, but does not close the dialog.

Cancel	Closes this dialog without making any changes.
Help	Displays a help page for this dialog.

Context

The Zoom dialog appears when you choose the Zoom item from the View menu of the 2D Profile or 3D Profile window.

Related Windows

2D Profile window
3D Profile window
Filter Profile dialog
Region Subset Selection dialog
Visibility Selection dialog

Commands

6

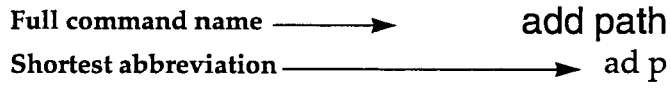
This chapter contains a reference page for each CXpa command. Commands can be entered in line mode (when you invoke CXpa with the `-nw` option) at the `(CXpa)` command prompt, used in CXpa script files, or executed in batch mode.

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 to 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 selectable`
`list`
`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 for SPP Series systems are as follows:

```
counts
cpu
wall_clock
events
```

<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. You can only specify one region type.

<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 "Introducing metrics,"Reports," "Loop reports," "Parallel Region reports," "Routine reports," "Report fields," and "Basic Block report" online help topics or sections of this book.

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. In this case, CPU time, wall clock time, and events will be collected (iteration/execution counts are always 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 1000 systems.

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 event 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 the file report_output is overwritten.

```
(CXpa) analyze events cpu
```

The above command creates and displays 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`.

```
(CXpa) analyze counts
```

The above command creates and displays iteration/execution counts reports only for all profiled regions in the program.

Related Commands

`deselect`
`rerun`
`run`
`select`
`set pdf`

Related Topics

Basic Block report
Introducing metrics
Introducing source code regions
Loop reports
Report fields
Reports
Routine reports
Parallel Region reports

collect
col

Specify the metrics that you want to collect while profiling your program.

Syntax

```
collect [counts] [cpu] [wall_clock] [events]
```

Parameter	Meaning
Specify one or more of the following parameters with the collect command. Separate multiple parameters with a space.	
counts	By default, iteration/execution counts are always collected. To restrict metric collection to iteration/execution counts, specify the `counts` parameter only with the `collect` command (for example, `collect counts`).
cpu	Collects CPU time and iteration/execution counts.
wall_clock	Collects wall clock time.
events	Collects events metrics that you can configure with the `set events` command.

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. Refer to the "`set events`" online help topic or command reference page in this book for more information.

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 routine all
2.(CXpa) collect cpu wall_clock events
3.(CXpa) set events data_cache
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 routine all` command tells CXpa to select all available routines in your program for profiling.
2. The `collect cpu wall_clock events` command tells CXpa which metrics to collect. In this case, CPU time, wall clock time, and events will be collected (by default, iteration/execution counts are also collected). You must then use the `set events` command to specify the type of event to collect.
3. The `set events data_cache` command configures the SPP 1200 on-processor event counters to collect data cache misses, accesses, and latency. The `data_cache` parameter is available on SPP 1200 systems only.
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 loop all
(CXpa) collect wall_clock cpu
```

The above commands select all loops in your program for profiling and enable CXpa to collect wall clock time and CPU time for loops.

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>*] [**-help**] [**-nap**] [**-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>*
-help	Display the CXpa usage message, which lists and describes command line options.
-nap	Disables auto-positioning of xterms in X window mode. You can use this option if you are running a virtual window manager (such as `tvtwm`) and you find that CXpa positions xterms that it creates (such as the process interface window) in unexpected locations (i.e., not in the current virtual root window). You can also fix the problem by setting the X application resource `Cxpa*autoPosition:` to `False` (the default setting is `True`).
-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.

-win *<windows>*

Specify the window or windows that are created automatically when you invoke CXpa in analysis mode (without specifying an executable). The default is `None`. Valid values for *<windows>* are as follows:

```
2DProfile
3DProfile
All
None
Report
Source
```

You can specify multiple values. Separate multiple values on the command line with a space.

-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, serial loops, parallel loops, 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. CXpa is a licensed producted. If you do not obtain a license at start-up, contact the site administrator for your system for assistance.
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`**

   CXpa is a licensed producted. If you do not obtain a license at start-up, contact the site administrator for your system for assistance.

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**—Name of the executable that you are profiling.
- **Current Process State**—Current state of the process you are profiling. The states include `running`, `paused`, `terminated`, `exited`, and `not started`.
- **Process ID**—Executable's process ID (PID).
- **Created on**—Date that the current executable was created.
- **Instrumentation Version**—Version of the CXpa profiling routines linked into your program with a CXpa compiler option (`-cxpa`, `-cxpar`, `-cxpab`, `-cxpamon`).

PDF information

- **Current PDF**—Name of the current performance data file (PDF).
- **Created on**—Date and time that the PDF was created.
- **PDF Format**—Format version number of the PDF.
- **Created by CXpa Version**—Version number of CXpa that created the PDF.
- **Process State**—Process state recorded in the PDF.
- **Executable Profiled**—Name of the executable you are profiling.
- **Executable Created on**—Date that the executable that produced the current PDF file was created.
- **Instrumentation Version**—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**—The name of the source file CXpa currently displays with the `list` or `list selectable` commands.
- **Current Working Directory**—Current directory.
- **Current Search Path(s)**—Search path that CXpa uses to find source files.
- **Current Subcomplex**—Subcomplex that your process is currently set to run on.
- **Available Subcomplex(s)** —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
       Current Subcomplex : System
  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'
 1 p  5 DO I=1,5
 1    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.

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.

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.

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
```

In the above example, all parallel loop regions in routine `sub2` are selected for profiling.

```
(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 events to collect at source code regions selected for profiling.

Syntax

`set events` <*event*(s)> **[`read_only`] [`write_only`]**

Parameter	Meaning
<*event(s)*>	Specifies the type of event or events to collect. The type and number of events that can be collected per program run differ according to machine architecture:

- On SPP 1200 systems, you can collect one set of on-processor events per program run.
- On SPP 1000 systems, you can collect one type of off-processor event per program run.

SPP 1000 off-processor events

Off-processor events on SPP 1000 systems are monitored by event counters on the Convex CPU agent chip. Valid off-processor events parameters are as follows:

`local_misses`— Configures SPP 1000 off-processor event counters to collect the number of times that data not found in the processor cache was found in local memory (memory allocated to that processor's hypernode). Requests for memory on the same hypernode use the hypernode crossbar.

`remote_misses`—Configures SPP 1000 off-processor event counters 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). Requests for memory on other hypernodes use the CTI rings.

`local_and_remote_misses`—Configures SPP 1000 off-processor event counters to collect the number of times that data not found in the processor cache was found in local and remote memory, combined.

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

`read_only`

The `read_only` parameter can only be specified for off-processor events (`local_and_remote_misses`, `remote_misses`, or `local_misses`). This option specifies that CXpa should collect these events during read operations. A *read miss* is a data cache miss caused by a load. By default, the off-processor event counters on SPP 1000 systems collect all events caused by reads or writes.

`write_only`

The `write_only` parameter can only be specified for off-processor events (`local_and_remote_misses`, `remote_misses`, or `local_misses`). This option 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, the off-processor event counters on SPP 1000 systems collect all events caused by reads or writes.

SPP 1200 on-processor events

SPP 1200 systems use HP PA-RISC 7200 processors which have on-processor event counters. These can be configured to collect one of the following sets of events per program run using the parameters listed below:

`data_cache`—Configures SPP 1200 on-processor event counters to collect total data cache access counts, miss counts, and latency.

`instruction_cache`—Configures SPP 1200 on-processor event counters to collect total instruction cache miss counts and latency.

`instruction_counts`—Configures SPP 1200 on-processor event counters to collect completed instruction counts and CPU clock cycles.

Description

The `set events` command specifies the collection of events 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 on SPP 1000 and SPP 1200 architectures, 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 sequence of commands shows 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 1000 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 remote_misses write_only
```

Remote misses can only be collected on multinode SPP 1000 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.

```
(CXpa) set events instruction_counts remote_misses
```

The above example is specific to SPP 1200 systems. The `instruction_counts` parameter configures the on-processor event counters to collect completed instruction counts and clock cycles, while the `remote_misses` parameter configures the off-processor event counters to monitor the number of data cache misses that were resolved remotely.

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

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 you want to run your program on. 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 hypernodes of an Exemplar SPP Series system (the subcomplex defines the boundary within which all threads belonging to a process execute).

The default value is the subcomplex from which you invoked CXpa. You need only 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 or to list the current subcomplex, 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)

```
       Current Subcomplex : System
  Available Subcomplex(s) : Chemistry, System
(CXpa) set subcomplex Chemistry
```

The last two lines in the output of the `info` command in the above example lists the current subcomplex (subcomplex your process is set to run on) and 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 the following:

- **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.

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 : 3.3
Product # : 710-018415-009
Release Date : 7/14/95

Copyright (c) 1989-1995 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`

version

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 reexecute 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 reexecute 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	Reexecute 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	Reexecute 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 reexecute 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	Line <*count*> in file <*filename*> was not compiled for or rejected for <*collection type*> profiling.
	Type	ERROR
	Explanation	Recompile the source file containing the line with one of the CXpa compiler options. Refer to the "CXpa limitations" online help topic or section in the *CXpa Reference for Exemplar Systems.*
A25	Message	Source file <*filename*> has changed since the PDF was created.
	Type	INFO
	Explanation	Line numbers reported in the Source Window may be incorrect. Regenerate the PDF to correct any line number inaccuracies.
A26	Message	No <*collection type*> to <*operation*> at line <*count*> in file <*filename*>.
	Type	ERROR
	Explanation	Reexecute 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 and their state of selection.

ID		Description
A27	Message	Source file must be specified with this command.
	Type	ERROR
	Explanation	Reexecute 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 reexecute it.
A30	Message	PDF contains incomplete data.
	Type	ERROR
	Explanation	Regenerate the PDF or select a different PDF.
A31	Message	PDF contains no data.
	Type	INFO
	Explanation	The PDF was created with no regions or metrics selected. Select the desired regions and metrics and regenerate the PDF.
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	Executable has been modified since start-up, please restart CXpa.
	Type	ERROR
	Explanation	The executable (<*filename*>) has been modified since product invocation. Please quit and restart CXpa to read in the new executable.
A34	Message	Obsolete.
	Type	ERROR
	Explanation	None.
A35	Message	Obsolete.
	Type	ERROR
	Explanation	None.

ID		Description
A36	Message	Obsolete.
	Type	ERROR
	Explanation	None.
A37	Message	Obsolete.
	Type	ERROR
	Explanation	None.
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 reexecute the command (in line/batch mode) or choose a PDF with the File menu option, New PDF, and reexecute 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	Obsolete.
	Type	ERROR
	Explanation	None.
A44	Message	Obsolete.
	Type	ERROR
	Explanation	None.
A45	Message	Directory <*filename*> does not exist.
	Type	ERROR
	Explanation	Reexecute the command (in line/batch mode) or GUI action (in X window mode) with a different directory name.

ID		Description
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 Profile 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 reexecute 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 for profiling.
	Type	ERROR
	Explanation	Recompile the source file containing the routine with one of the CXpa compiler options. Refer to the "CXpa limitations" online help topic or section in the *CXpa Reference for Exemplar Systems.*
A54	Message	Cannot <*operation*> alternate entries to routine <*routine identifier*>.
	Type	ERROR
	Explanation	Reexecute the command (in line/batch mode) or GUI action (in X window mode) using the main entry point to the routine.

ID		Description
A55	Message	Routine <*routine identifier*> is not unique, qualify using filename:routine.
	Type	ERROR
	Explanation	Reexecute 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	Cannot <*operation*> <*collection type*> in routine <*routine identifier*>.
	Type	ERROR
	Explanation	Reexecute the command with a different routine name. Use the "list selectable" command (in line/batch mode) or the Region Selection dialog (in X window mode) to view the available source regions and their state of selection.
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 sources files compiled for profiling with CXpa. Recompile source file(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.
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.

ID		Description
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	None.
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.
A70	Message	No regions selected for a customized 2D or 3D profile.
	Type	INFO
	Explanation	Bring up the Region Subset Selection dialog by selecting an item from the Select Regions options menu, and select one or more regions to profile.

ID		Description
T1	Message	<*string*>.
	Type	ERROR
	Explanation	Product test message.
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.
C23	Message	Tools out of date with respect to compiler. Use a newer version of CXdb or CXpa.
	Type	WARNING
	Explanation	A more recent version of CXdb or CXpa should be used with this executable in order to take advantage of the new CTI information emitted by the compiler. New CTI information will be ignored.
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	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.

ID		Description
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.
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.

ID		Description
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.
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.

ID		Description
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.
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.

Glossary

A

ALL
Acronym for Convex Assembler, Loader, and Libraries.

B

bank conflict
An attempt to access a particular memory bank before a previous access to the bank is complete.

basic block
A linear sequence of machine instructions with a single entry and a single exit.

blocking factor, loop
Integer representing the stride of the outer strip of a pair of loops created by blocking.

C

cache
A small, high-speed buffer memory used in modern computer systems to hold temporarily those portions of the contents of the main memory that are, or are believed to be, currently in use. Cache memory is physically separate from main memory and can be accessed with substantially less latency. Convex SPP Series computers employ separate data and instruction cache memories.

cache, fully associative
A form of cache memory that attempts to prevent encached data from being overwritten by storing an incoming element in a cache location that is determined using a hashing algorithm. The incoming element's virtual address is irrelevant. Unlike a direct mapped cache, a fully associative cache will never displace a cache line unless the cache is full. SPP1200 Series machines use 2-kbyte, on-chip, fully associative assist caches to mitigate cache thrashing.

cache, direct mapped
A form of cache memory which addresses encached data by a function of the data's virtual address. On SPP1000 Series computers, the cache address is identical to the least significant 20 bits of the data's virtual address. This means cache thrashing can occur when the virtual addresses of two data items are an exact multiple of 1 Mbyte (20 bits) apart. On SPP1200 Series computers, the main cache address is identical to the least significant 18 bits of the data's virtual address. Cache thrashing can therefore occur on SPP1200 machines when the virtual addresses of two data items are an exact multiple of 256 kbytes (18 bits) apart.

cache hit
A cache hit occurs if data to be loaded resides in the cache.

cache line
A chunk of contiguous data that is copied into a cache in one operation. On SPP Series computers, processor cache lines consist of 32 bytes of data. When a processor cache miss occurs and data must be fetched from outside the processor cache, the requested data is brought in as part of a 32-byte cache line. CTIcache lines consist of 64 bytes of data (two contiguous processor cache lines). When a CTIcache miss occurs, the requested data is brought into the CTIcache as part of a 64-byte cache line.

cache misses, data
Number of times requested data was not found in the processor's data cache.

cache misses, instruction
Number of times that referenced instructions were not found in the processor's instruction cache.

cache thrashing
Cache thrashing occurs when two or more data items that are frequently needed by the program map to the same cache address. In this case, each time one of the items is encached, it overwrites another needed item, causing constant cache misses and impairing data reuse.

chunk
A unit of work in a loop that has been parallelized by the compiler consisting of a number of loop iterations.

chunk count
Number of chunks (packets of loop iterations) assigned to execute on a particular thread in a parallel loop.

clock cycle
The duration of the square wave pulse sent throughout a computer system to synchronize operations. The clock cycle time for Convex SPP 1000 Series systems is ~10 nanoseconds.

clock cycles per instruction
A measure of the efficiency of instruction scheduling. CXpa calculates this metric during analysis if instruction counts and clock cycles are collected. On SPP 1200 systems, the maximum number of instructions that can be executed in a single clock cycle is 2. This means that the theoretical peak value for the clock cycles per instruction metric on this architecture is 0.5.

coherency (cache)
A term frequently applied to caches. If a data item is referenced by a particular processor on a multiprocessor system, the data is copied into that processor's cache and is updated there if the processor modifies the data. If another processor references the data while a copy is still in the first processor's cache, a mechanism is needed to ensure that the second processor does not use an outdated copy of the data from memory. The state that is achieved when both processors' caches always have the latest value for the data is called cache coherency. On SPP Series computers, an item of data may reside concurrently in several processors' caches.

compiler
Software that converts high-level language programs into machine code. Compilers for parallel systems improve performance of applications by automatically determining the best way to execute program procedures when several processors are available.

Compiler Parallel Support library (CPSlib)
A library of thread management and synchronization routines that can be used to control parallelism on Exemplar systems. Using CPSlib requires you to manually control all aspects of parallelism, synchronization, and data partitioning.

concurrency factor, parallel
The ratio of CPU time to wall clock time. For parallel regions, this metric is an indicator of the speed-up achieved through parallelism. Values that approach n, where n is the number of processors, indicate good parallel concurrency.

CPU Agent
The gate array on Convex SPP Series systems that provides a high-speed interface between the pairs of PA-RISC processors in a functional block and the crossbar. Also called the Agent and the CPA.

CPU time
Time the CPU spent in the program (user CPU time), not including time waiting for I/O or running other programs. If a program can use multiple processors, the CPU time may be greater than the wall clock time.

CPU/wall clock
Ratio of CPU time to wall clock time. For serial regions, if the CPU/Wall clock ratio is high, the region is compute-bound. For parallel regions, this ratio corresponds to the concurrency factor, which is an indicator of the speed-up achieved through parallelization. Values that approach *n*, where *n* is the number of processors, indicate good parallel concurrency. For both parallel and serial regions, if the CPU/Wall clock ratio is low, this could indicate a performance bottleneck caused by I/O calls, system calls, or memory accesses.

crossbar
A switching device that connects the CPUs, banks of memory, and I/O controller on a single hypernode of a Convex SPP Series system. Because the crossbar is nonblocking, all ports can run at full bandwidth simultaneously unless there is contention for a particular port.

CTIcache
A partition of physical memory that exists on each hypernode and is used to store copies of global data fetched from other hypernodes.

CTI (Coherent Toroidal Interface) ring
The ring interconnect that connects all the hypernodes of a multihypernode Convex SPP Series system together in a ring topology. While the CTI ring is derived from the IEEE SCI standard, complete compatibility is sacrificed to provide lower latencies.

D

data cache
On Convex SPP 1000 systems, a 1-Mbyte, write-back, direct-mapped cache memory with a 1-clock cycle access time. This cache holds prefetched and current data. On Convex SPP 1200 systems, there are two data caches—a 256-kbyte off-chip, write-back, direct-mapped main cache and a 2-kbyte, fully associative, on-chip assist cache; the access time is also 1 clock cycle.

data cache hit
A data cache hit occurs if data to be loaded resides in the processor's data cache.

data cache hit rate

The percentage of total data cache accesses that were data cache hits. If the data cache hit rate is low, this indicates cache thrashing. CXpa uses the following formula to calculate the data cache hit rate:

$$data\_cache\_hit\_rate = \frac{total\_data\_cache\_accesses - data\_cache\_misses}{data\_cache\_accesses} \times 100$$

data cache miss

A data cache miss occurs if data to be loaded does not reside in the processor's data cache.

dynamic selection

The process by which the compiler chooses the appropriate clone of a code section (typically, a loop) at runtime, in order to achieve the lowest execution time.

E

event counts

The number of times an event, such as a data or instruction cache miss, occurred.

G

globally shared memory (GSM)

A feature of some parallel systems that lets all processors transparently access any location in main system memory. This lets a single, scalable operating system control the resources of the system. The arrangement presents a familiar environment to developers and users. On SPP Series computers, globally shared memory is distributed among hypernodes, but any hypernode's memory is accessible from any processor on any hypernode. This type of memory is sometimes referred to as shared virtual memory or global virtual memory.

H

hypernode
In Exemplar SPP Series systems, a set of up to eight processors and physical memory organized as a symmetric multiprocessor (SMP) running a single image of the operating system microkernel. An SPP system consists of one or more hypernodes, with a high speed CTI ring connecting the hypernodes.

I

instruction cache
On SPP1000 systems, a 1-Mbyte cache memory with a 1-clock cycle access time. This cache holds prefetched instructions and permits the simultaneous decoding of one instruction with the execution of a previous instruction. On SPP1200 systems it is 256 kbytes in size with a 1-clock cycle access time.

instruction cache miss
An instruction cache miss occurs when a memory reference for an instruction cannot be resolved in the processor cache.

J

join
The synchronized termination of parallel execution by spawned tasks or threads.

L

latency time

The amount of time spent accessing memory to locate data or instructions not found in the processor's data or instruction cache.

latency/counts
Average latency time per event for a code region. For example, if total data cache miss counts are collected, then the event latency/counts value computed by CXpa during analysis represents the average amount of time it took to resolve a data cache miss.

latency/CPU
Ratio of event latency to CPU time for a code region, expressed as a percentage. CXpa calculates this metric during analysis if data or instruction cache miss events, latency, and CPU time are collected. When this percentage is high, it means that the program spent the majority of its CPU time in the region "reaching" for data or instructions that were not encached rather than computing with encached data or instructions.

load
An instruction used to move the contents of a memory location into a register.

locality of reference
An attribute of a memory reference pattern that refers to the likelihood of an address of a memory reference being physically close, in terms of the number of clock cycles necessary to access it, to the CPU making the reference.

localization
Data localization. Optimizations designed to keep frequently used data in the processor data cache, thus eliminating the need for more costly CTIcache or main memory accesses.

locally resolved cache misses
Cache misses that are resolved by using the hypernode crossbar to access memory found on the same hypernode as the processor; the CTI rings are not used.

loop blocking
A loop transformation that strip-mines and interchanges a loop to provide optimal reuse of the encachable loop data.

loop distribution
The restructuring of a loop nest to create simple loop nests. Loop distribution creates two or more loops, called distributed parts, which can serve to make parallelization more efficient by increasing the opportunities for loop interchange and isolating code that must run serially from code that can be parallelized. It can also improve data localization and other optimizations.

loop interchange
The reordering of nested loops. Loop interchange is generally done to increase the granularity of the parallelizable loop(s) present or to allow more efficient access to loop data.

loop replication
The process of transforming one loop into more than one loop to facilitate an optimization. The optimizations that replicate loops are `IF-DO` and `if-for` optimizations, dynamic selection, loop unrolling, unroll and jam, and loop blocking.

loop strip mining
The transformation of a single loop into two nested loops. Conceptually, this is how parallel loops are created by default. A conceptual outer loop advances the initial value of the inner loop's induction variable by the parallel strip length. The parallel strip length is based on the trip count of the loop and the amount of code in the loop body. Strip mining is also used by the data localization optimization.

loop unrolling
A loop transformation performed by the compiler that involves increasing a loop's step value and replicating the loop body, with each replication appropriately offset from the induction variable so that all iterations are performed, given the new step. Unrolling can be total or partial.

Total unrolling involves eliminating the loop structure completely, replicating the loop body a number of times equal to the iteration count, and replacing the iteration variable with constants. Total unrolling is generally performed on loops with small iteration counts.

Partial unrolling is performed on loops with larger or unknown iteration counts. It retains the loop structure, but replicates the body a number of times equal to the unrolling factor, and adjusts references to the iteration variable accordingly.

M

memory bank conflict
An attempt to access a particular memory bank before a previous access to the bank is complete.

MIPS
Millions of instructions per second.

MIPS rate
CXpa uses the following formula to calculate the MIPS rate:

$$MIPS\_rate = \frac{Number\_of\_instructions\_completed}{Wall\_clock\_time\_in\_sec}$$

The theoretical peak MIPS rate for Hewlett-Packard PA-RISC 7100/7200 processors with a 10-ns clock cycle at a clock rate of 100 MHz is 200 MIPS.

N

NUMA
Nonuniform memory access. This term describes memory access times in systems such as the SPP Series, in which different types of memory result in nonuniform access times.

O

off-processor events
Off-processor events are monitored by event counters (when available) on the Convex CPU agent chip. On an SPP 1000 system, these event counters can be configured to monitor the number of data cache miss events that are resolved locally and/or remotely (remote miss events imply use of the CTI rings; local miss events do not use the CTI rings) and whether those events occurred due to reads (loads) or writes (stores).

on-processor events
On-processor events are monitored by event counters (when available) located on the HP PA-RISC processors used in SPP Series systems. These event counters monitor events from the processor's point of view only.

optimization
The refining of application software programs to minimize processing time. Optimization takes maximum advantage of a computer's hardware features and minimizes input/output traffic and idle processor time.

optimization level
The degree to which source code is optimized by the compiler. The Convex Fortran and C compilers have five levels of optimization, level `-no`, `-O0`, `-O1`, `-O2`, and `-O3`. The default level for Convex compilers is `-O2`.

oversubscription
In the context of parallel threads, a process attribute that permits the creation of more threads within a process than the number of processors available to the process.

P

parallel optimization
The transformation of source code into parallel code (parallelization) and restructuring of code to enhance parallel performance.

parallelization
The process of transforming serial code to a form of code that can run simultaneously on multiple CPUs while preserving semantics. At optimization level `-O3`, the Convex compilers automatically parallelize loops in your program and recognize compiler directives with which you can manually specify parallelization of both loops and tasks.

parallelization, loop
The process of splitting a loop into several smaller loops, each of which operates on a subset of the data of the original loop, and generating code to run these loops on separate processors in parallel.

PDF (performance data file)
A binary file CXpa creates 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.

process
A collection of one or more execution streams within a single logical address space; an executable program. A process is made up of one or more threads.

R

read
A memory operation in which the contents of a memory location are copied and passed to another part of the system.

read miss
A data or instruction cache miss that occurred as a result of a read operation.

reduced instruction set computer (RISC)
An architectural concept that applies to the definition of the instruction set of a processor. A RISC instruction set is an orthogonal instruction set that is easy to decode in hardware and for which a compiler can generate highly optimized code. The PA-RISC processor used in SPP Series computers employs a RISC architecture.

remotely resolved cache misses
Cache misses that were resolved by using the CTI rings to access memory on another hypernode. Accessing memory on other hypernodes using the CTI rings takes significantly longer than accessing memory on the same hypernode via the hypernode crossbar.

S

Scalable Coherent Interface (SCI)
Scalable Coherent Interface. This is defined by IEEE standard 1596-1992. The interface is physically defined as a pair of 18-bit, differential ECL, unidirectional links which are clocked at 250 MHz. Each link provides 16 bits of data with two control signals. Data is sampled on both the rising and falling edges of the clock. This interface provides the basis for the CTI rings used in Convex SPP Series systems; however, total compatibility with the standard has been sacrificed to provide increased performance.

Scalable Parallel Processor (SPP)
Scalable parallel processor systems combine the accessibility and proven technology of single-processor workstations with symmetric multiprocessors' ease of programming and ability to address large problems. SPPs can be scaled or expanded to serve additional users when necessary. SPP systems are capable of handling hundreds of high-performance processors. Compute capacity, memory, and other subsystems also scale when necessary. A key design goal for SPP systems is to enable performance to increase linearly with respect to its number of processors.

spawn
To activate existing threads.

store
An instruction used to move the contents of a register to memory.

subcomplex
In Convex SPP Series systems, a logical entity that provides control over the allocation of processors and physical memory to different applications and users.

symmetric multiprocessor (SMP)
A multiprocessor computer in which all the processors have equal access to all machine resources. Symmetric multiprocessors have no master or slave processors; the operating system runs on any or all of the processors.

system subcomplex
In a Convex SPP Series system, a subcomplex that is automatically created at boot time by the operating system to run system processes, including `init` and processes spawned by `init`. The Subcomplex Manager will not allow users to destroy this subcomplex, nor remove the last processor from this subcomplex.

T

thread
An independent execution stream that is executed by a CPU. One or more threads, each of which can execute on a different CPU, make up each process. Memory, files, signals, and other process attributes are generally shared among threads in a given process, enabling the threads to cooperate in solving the common problem. Threads are created and terminated by instructions that can be automatically generated by Convex compilers, inserted by adding compiler directives to source code, or coded explicitly using library calls or assembly-language.

thread identifier (thread ID)
An integer identifier associated with a particular thread. See thread identifier, kernel (ktid) and thread identifier, spawn (stid).

thread identifier, kernel (ktid)
A unique integer identifier (not necessarily sequential) assigned when a thread is created.

thread identifier, spawn (stid)
A sequential integer identifier associated with a particular thread that has been spawned. stids are only assigned to spawned threads, and they are assigned within a spawn context; therefore, duplicate stids may be present amongst the threads of a program, but stids are always unique within the scope of their spawn context. stids are assigned sequentially and run from 0 to one less than the number of threads spawned in a particular spawn context.

W

wall clock time
Time to solution or elapsed time for a program, including disk accesses, memory accesses, input/output, and operating system overhead. Compare with CPU time.

write
A memory operation in which a memory location is updated with new data.

write miss
A data or instruction cache miss that occurred as a result of a read operation.

Glossary

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

ORDER NUMBER
DSW-605

DOCUMENT NUMBER
710-004730-008