Performance Co-Pilot™ for IRIX®
Advanced User's and Administrator's Guide

007–2614–006

Version 2.3

CONTRIBUTORS

Production by Karen Jacobson

Engineering and written contributions by David Chatterton, Michael Gigante, Mark Goodwin, Tony Kavadias, Seppo Keronen, Johnathon Knispel, Ken McDonell, Max Matveev, Ania Milewska, Daniel Moore, Heidi Muehlebach, Ivan Rayner, Nathan Scott, Timothy Shimmin, Terry Schultz, and Bill Tuthill

Cover design by Sarah Bolles, Sarah Bolles Design, and Dany Galgani, SGI Technical Publications.

New Features in This Guide

In this update supporting the PCP 2.3 release, documentation describing PCP software embedded in the IRIX operating system has been moved to the *Performance Co-Pilot for IRIX User's and Administrator's Guide*.

PCP is presented as two sets of products; the foundation product components that are provided as part of the IRIX operating system (`pcp_eoe`) and are documented in *Performance Co-Pilot for IRIX User's and Administrator's Guide* and the enhancement product components that are available to add to the existing PCP distribution (`pcp` and `pcp_gifts`) and are documented in this manual.

Record of Revision

Version	Description
004	July 1999 Revised to support the Performance Co-Pilot release 2.1 for SGI systems running the IRIX 6.2, 6.3, 6.4, and 6.5 operating systems.
005	March 2001 Revised to support the Performance Co-Pilot release 2.2 for SGI systems running the IRIX 6.2, 6.3, 6.4, and 6.5.11 and later operating systems.
006	December 2002 Revised to support the Performance Co-Pilot release 2.3 release for SGI systems running the IRIX 6.5.18 and later operating systems.

Contents

Figures

Tables

Examples

Procedures

About This Guide

This guide describes the Performance Co-Pilot (PCP) software package of advanced performance tools for the SGI family of graphical workstations and servers.

The *Performance Co-Pilot for IRIX Advanced User's and Administrator's Guide* documents the PCP enhancement features that are in the Performance Co-Pilot (PCP) software package (`pcp` and `pcp_gifts`), which users purchase separately.

The *Performance Co-Pilot for IRIX User's and Administrator's Guide* documents the PCP features that are embedded in the IRIX operating system. It is a prequel to the *Performance Co-Pilot for IRIX Advanced User's and Administrator's Guide*, which only provides descriptions about tools and services which are not provided in the foundation components of PCP.

PCP provides a systems-level suite of tools that cooperate to deliver integrated performance monitoring and performance management services spanning the hardware platforms, operating systems, service layers, Database Management Systems (DBMSs), and user applications.

"About This Guide" includes short descriptions of the chapters in this book, directs you to additional sources of information, and explains typographical conventions.

What This Guide Contains

This guide contains the following chapters:

- Chapter 1, page 1, provides an introduction, a brief overview of the software components, and conceptual foundations of the PCP product.
- Chapter 2, page 11, describes the product structure of PCP and license constraints for advanced PCP features and provides a reference to basic installation and configuration information necessary to get PCP running on your systems.
- Chapter 3, page 15, summarizes user interface components of the graphical tools and text-based utilities introduced by installing the `pcp` product.
- Chapter 4, page 21, describes the basic interactive performance monitoring tools available in PCP, including `pmchart`, `pmgadgets` and `pmdumptext`.

- Chapter 5, page 47, discusses the various three-dimensional (3D) visualization tools that are provided to enable high-level monitoring, management, and diagnosis for performance problems.
- Chapter 6, page 71, covers the Performance Metrics Inference Engine (`pmie`) is a tool that provides automated monitoring of, and reasoning about, system performance within the Performance Co-Pilot (PCP) framework. This chapter only provides descriptions about tools and services which are not provided in the foundation components of PCP.
- Chapter 7, page 77, covers the PCP services and utilities that support archive logging for capturing accurate historical performance records. This chapter only provides descriptions about tools and services which are not provided in the foundation components of PCP.
- Chapter 8, page 81, describes the procedures necessary to ensure that the PCP configuration is customized in ways that maximize the coverage and quality of performance monitoring and management services.
- Appendix A, page 89, provides a comprehensive list of the acronyms used in this guide, in the man pages, and in the release notes for Performance Co-Pilot.

Audience for This Guide

This guide is written for the system administrator or performance analyst who is directly using and administering PCP applications. It is assumed that you have installed IRIS InSight for viewing online books, or have access to the *IRIX Admin* manual set, including *IRIX Admin: System Configuration and Operation*, and the *Personal System Administration Guide* as hard–copy documents.

Related Resources

The *Performance Co-Pilot Programmer's Guide*, a companion document to the *Performance Co-Pilot for IRIX Advanced User's and Administrator's Guide*, is intended for application developers who want to use the PCP framework and services for exporting additional collections of performance metrics, or for delivering new or customized applications to enhance performance management.

IRIX Admin: System Configuration and Operation describes how to perform general system configuration and operation tasks under the IRIX operating system used with

SGI workstations and servers. The *Personal System Administration Guide* provides similar information for graphics workstations.

Additional resources include man pages, release notes, and SGI Web sites.

Man Pages

The IRIX man pages provide concise reference information on the use of IRIX commands, subroutines, and system resources. There is usually a man page for each PCP command or subroutine. To see a list of all the PCP man pages, enter the following command:

```
man -k performance
```

To see a particular man page, supply its name to the `man` command, for example:

```
man pcp
```

The man pages are divided into the following seven sections:

(1) General commands

(2) System calls and error numbers

(3) Library subroutines

(4) File formats

(5) Miscellaneous

(6) Demos and games

(7) Special files

When referring to man pages, this guide follows a standard UNIX convention: the section number in parentheses follows the item. For example, `pmda`(3) refers to the man page in section 3 for the `pmda` command.

Release Notes

Release notes provide specific information about the current product release, available online through the `relnotes` command. Exceptions to the printed and online documentation are found in the release notes. The `grelnotes` command provides a graphical interface to the release notes of all products installed on your system. For additional information, see the `relnotes`(1) and `grelnotes`(1) man pages.

SGI Web Sites

The following Web sites are accessible to everyone with general Internet access:

URL	Description
`http://www.sgi.com`	The SGI general Web site, with search capability
`http://www.sgi.com/software`	Links to Performance Co-Pilot product information
`http://oss.sgi.com/projects/pcp`	Some parts of the PCP infrastructure that have also been released as open source

Obtaining Publications

To obtain SGI documentation, go to the SGI Technical Publications Library at `http://docs.sgi.com`.

Conventions

The following conventions are used throughout this document:

Convention	Meaning
`command`	This fixed-space font denotes literal items such as commands, files, routines, path names, signals, messages, and programming language structures.
variable	Italic typeface denotes variable entries and words or concepts being defined.
`user input`	This bold, fixed-space font denotes literal items that the user enters in interactive sessions. (Output is shown in nonbold, fixed-space font.)
[ ]	Brackets enclose optional portions of a command or directive line.
...	Ellipses indicate that a preceding element can be repeated.
ALL CAPS	All capital letters denote environment variables, operator names, directives, defined constants, and macros in C programs.
()	Parentheses that follow function names surround function arguments or are empty if the function has no arguments; parentheses that follow IRIX commands surround man page section numbers.

Reader Comments

If you have comments about the technical accuracy, content, or organization of this document, contact SGI. Be sure to include the title and document number of the manual with your comments. (Online, the document number is located in the front matter of the manual. In printed manuals, the document number is located at the bottom of each page.)

You can contact SGI in any of the following ways:

- Send e-mail to the following address:

  techpubs@sgi.com

- Use the Feedback option on the Technical Publications Library Web page:

  `http://docs.sgi.com`

- Contact your customer service representative and ask that an incident be filed in the SGI incident tracking system.

- Send mail to the following address:

  Technical Publications
  SGI
  1600 Amphitheatre Parkway, M/S 535
  Mountain View, California 94043–1351

- Send a fax to the attention of “Technical Publications” at +1 650 932 0801.

SGI values your comments and will respond to them promptly.

Introduction to Performance Co-Pilot

This chapter provides an introduction to Performance Co-Pilot (PCP), an overview of its individual components, and conceptual information to help you use this product.

The following sections are included:

- Section 1.1 covers the intended purposes of PCP.
- Section 1.2, page 1, describes PCP tools and agents.

1.1 Objectives

Performance Co-Pilot (PCP) provides a range of services that may be used to monitor and manage system performance. These services are distributed and scalable to accommodate the most complex system configurations and performance problems.

For an overview of PCP, read "Introduction to Performance Co-Pilot" in *Performance Co-Pilot for IRIX User's and Administrator's Guide*.

1.2 Overview of Component Software

Performance Co-Pilot (PCP) is composed of text-based tools, optional graphical tools, and related commands. Each tool or command is fully documented by a man page. These man pages are named after the tools or commands they describe, and are accessible through the `man` command. For example, to see the `pminfo`(1) man page for the `pminfo` command, enter this command:

```
man pminfo
```

Many PCP tools and commands are accessible from an **Icon Catalog** on the IRIX desktop, grouped under **PerfTools**. In the **Toolchest Find** menu, choose **PerfTools**; an **Icon Catalog** appears, containing clickable PCP programs. To bring up a Web-based introduction to Performance Co-Pilot, click the **AboutPCP** icon.

A list of PCP tools and commands, grouped by functionality, is provided in the following sections.

1.2.1 Performance Monitoring and Visualization

The following tools provide the principal services for the PCP end-user with an interest in monitoring, visualizing, or processing performance information collected either in real time or from PCP archive logs:

`arraytop`	Displays the active local and global array sessions for all hosts in the SGI array *arrayname*.
`arrayvis`	Displays a three-dimensional (3D) bar chart of CPU, memory, disk, and network utilization for the hosts within the array *arrayname*.
`ashtop`	Displays the top processes in the array session `ash` on all machines in the array named *arrayname*.
`clustervis`	Displays a set of 3D bargraphs that show user-mode CPU utilization and network traffic for a collection of hosts. It is a front-end wrapper for `pmview`.
`dkvis`	Displays a 3D bar chart showing activity in the disk subsystem. It is a front-end wrapper for `pmview`.
`mpvis`	Displays a 3D bar chart of multiprocessor CPU utilization. It is a front-end wrapper for `pmview`.
`mpimon`	Is a script that launches either `pmlogger` or `pmchart` to collect MPI function and communication metrics. It is for an instrumented Message Passing Interface (MPI) application.
`mpivis`	Displays a set of 3D bargraphs that show Message Passing Inferface (MPI) performance metrics for a collection of hosts. It is a front-end wrapper for `pmview`.
`nfsvis`	Displays a 3D bar chart showing Network File System (NFS) client and server request activity, for systems on which the optional NFS software product has been installed. It is a front-end wrapper for `pmview`.
`nodevis`	Visualizes SGI Origin node statistics on platforms that support this hardware.
`osvis`	Displays 3D bar charts covering many aspects of system performance, including disk use, job load,

	memory, CPU activity, and network I/O. It is a front-end wrapper for `pmview`.
`oview`	Visualizes the performance of SGI 3000 series and SGI 2000 series of systems, showing a dynamic display of node topology and performance.
`pmchart`	Displays trends over time for arbitrarily selected performance metrics from one or more hosts, or from one or more performance metric domains.
`pmdumpmineset`	Is a wrapper for `pmdumptext` that produces data files suitable for importing into the MineSet data mining product.
`pmdumptext`	Outputs the values of performance metrics collected live or from a PCP archive, as ASCII text.
`pmem`	Reports per-process memory usage statistics. Both virtual size and prorated physical memory usage are reported.
`pmgadgets`	Creates a small window containing a collection of graphical gadgets of assorted type and style, driven by performance metrics supplied by the PCP framework. Any numeric metric can be used to animate a gadget. This command is not normally invoked directly by users.
`pmgcisco`	Monitors interface throughput for Cisco routers using the `pmgadgets` tool.
`pmgcluster`	Is a miniature operating system performance monitor for multiple hosts. It builds a specification for the `pmgadgets` tool, which displays a miniature visual overview of CPU, memory, disk, and network activity for a collection of hosts.
`pmgevctr`	Uses `pmgadgets` to display an animated gadget that reports activity in the CPU and memory subsystems along with selected the R10K/R12K event counters and SGI Origin router metrics.
`pmgshping`	Monitors service quality and availability as measured by the `shping` PMDA using the `pmgadgets` tool.

`pmgsys`	Determines the hardware configuration of a remote or local system, constructs a suitable specification for a system-level visual monitor, and launches the `pmgadgets` tool to animate the monitor using IRIX performance metrics.
`pmgweb`	Is a miniature Web-server performance monitor for multiple hosts. It builds a specification for the `pmgadgets` tool, which displays a miniature visual overview of a range of Web-server performance metrics including requests per second, bytes transferred, and network errors.
`pmie`	Evaluates predicate-action rules over performance metrics domain, for performance alarms, automated system management tasks, dynamic tuning configuration, and so on. It is an inference engine.
`pmieconf`	Creates parameterized rules to be used with the PCP inference engine (`pmie`).
`pminfo`	Displays information about arbitrary performance metrics available from PCP, including help text with `-T`.
`pmkstat`	Provides a text-based display of metrics that summarize system performance at a high level, suitable for ASCII logs or inquiry over a modem.
`pmlogsummary`	Calculates and reports various statistical summaries of the performance metric values from a PCP archive.
`pmprobe`	Probes for performance metric availability, values, and instances.
`pmsocks`	Allows the execution of PCP tools through a network firewall system provided `sockd` services are supported.
`pmtime`	Provides a graphical user interface for PCP applications requiring time control. This command is not normally invoked directly by users.
`pmval`	Provides a text-based display of the values for arbitrary instances of a selected performance metric, suitable for ASCII logs or inquiry over a modem.

`pmview`	Supports dynamic displays of clusters of related performance metrics as groups of utilization blocks (or towers) on a common base plane. The `pmview` tool is a generalized three-dimensional (3D) Open Inventor application. This command is not normally invoked directly by users.
`procvis`	Displays a set of 3D bargraphs showing the CPU utilization of an individual process. It is a front-end wrapper for `pmview`.
`psmon`	Selects a subset of the actively running processes and launches either `pmchart` or `pmlogger` to collect per-process metrics for those processes.
`routervis`	Visualizes SGI Origin router utilization on platforms that support this hardware.
`webvis`	Displays a set of 3D bargraphs that show system-level Web-server activity. The display includes a histogram of Web-page request size, network metrics, CPU, memory, network, and disk utilization. It is a front-end wrapper for `pmview`.
`weblogvis`	Displays a set of three-dimensional (3D) bargraphs that show the requests received by a Web server as determined from activity within the server's log files. It shows a histogram of request rates, total requests, and idle time. It is a front-end wrapper for `pmview`.
`webpingvis`	Displays a set of 3D bargraphs that show Web-server response times for a collection of hosts. It is a front-end wrapper for `pmview`.
`xbowvis`	Visualizes the Crossbow (XBow) packet and error rates on platforms that support this hardware. It is a front-end wrapper for `pmview`.
`xlv_vis`	Visualizes XLV volume activity and performance.

1.2.2 Collecting, Transporting, and Archiving Performance Information

PCP provides the following tools to support real-time data collection, network transport, and archive log creation services for performance data:

`mkaf`	Aggregates an arbitrary collection of PCP archive logs into a *folio* to be used with `pmafm`.
`mkpmemarch`	Creates a PCP archive suitable for use with the `pmem` tool.
`pmafm`	Interrogates, manages, and replays an archive folio as created by `mkaf`, or the periodic archive log management scripts, or the record mode of other PCP tools.
`pmcd`	Is the Performance Metrics Collection Daemon (PMCD). This daemon must run on each system being monitored, to collect and export the performance information necessary to monitor the system.
`pmcd_wait`	Waits for `pmcd` to be ready to accept client connections.
`pmdaarray`	Uses `pmie` to collect, aggregate, and summarize performance metrics across all of the hosts of an SGI array.
`pmdaash`	Extracts array session performance metrics. These metrics represent the IRIX process accounting records, aggregated according to the Array Session Handle (`ash`) for each process.
`pmdabrocade`	Measures the bytes read and written across each port of a Brocade fiber channel switch.
`pmdadmf`	Extracts performance metrics from the Data Migration Facility (DMF) logfiles.
`pmdahippi`	Extracts performance metrics from the HIPPI network interface.
`pmdahotproc`	Exports performance metrics from an instance domain of processes restricted to an interesting or “hot” set. It is a PMDA.
`pmdamailq`	Exports performance metrics describing the current state of items in the `sendmail` queue. It is a PMDA.
`pmdacisco`	Extracts performance metrics from one or more Cisco routers. It is a Performance Metrics Domain Agent (PMDA).

`pmdampi`	Exports metrics from MPI applications linked with the `pcp_mpi` shared library. The metrics include counts and accumulated time for selected MPI functions and statistics on MPI buffer usage.
`pmdasendmail`	Exports mail activity statistics from `sendmail`. It is a PMDA.
`pmdashping`	Exports performance metrics for the availability and quality of service (response-time) for arbitrary shell commands. It is a PMDA.
`pmdasummary`	Derives performance metrics values from values made available by other PMDAs. It is a PMDA.
`pmdatrace`	Exports transaction performance metrics from application processes that use the `pcp_trace` library. It is a PMDA.
`pmdaweblog`	Scans Web-server logs to extract metrics characterizing.
`pmdawebping`	Makes requests for selected Universal Resource Locations (URLs) from Web servers and returns metrics on the response time and status of the requests.
`pmdumplog`	Displays selected state information, control data, and metric values from a PCP archive log created by `pmlogger`.
`pmimport`	Converts arbitrary time-stamped data into a PCP archive. Shipped configurations enable SAR data files from `sadc` to be translated into PCP archives.
`pmlc`	Exercises control over an instance of the PCP archive logger `pmlogger`, to modify the profile of which metrics are logged and/or how frequently their values are logged.
`pmlogcheck`	Performs integrity check for PCP archives.
`pmlogconf`	Creates or modifies `pmlogger` configuration files for most common logging scenarios. It is an interactive script.
`pmlogextract`	Reads one or more PCP archive logs and creates a temporally merged and reduced PCP archive log as output.

`pmlogger`	Creates PCP archive logs of performance metrics over time. Many tools accept these PCP archive logs as alternative sources of metrics for retrospective analysis.
`pmtrace`	Provides a simple command line interface to the trace PMDA and its associated `pcp_trace` library.
`webping`	Sends one or more HTTP requests every interval (default is 5) seconds, reporting on timings to connect and receive the HTML header and body.

1.2.3 Operational and Infrastructure Support

PCP provides the following tools to support the PCP infrastructure and assist operational procedures for PCP deployment in a production environment:

`dkmap`	Creates a map of disk real estate usage.
`dkping`	Opens the named disk for reading and checks for a response.
`dkprobe`	Initializes disk performance metrics at boot time for some IRIX versions. It may be called from `/etc/init.d/pcp`.
`hipprobe`	Probes the state of the configured HIPPI interfaces. Used by the `shping` PMDA.
`memclaim`	Allocates and holds physical memory, simulating a reduction in physical memory.
`pcp`	Summarizes that state of a PCP installation.
`pmbrand`	Manages the “branded” file of valid PCP licenses.
`pmdate`	Displays the current date and/or time, with an optional offset.
`pmdbg`	Describes the available facilities and associated control flags. PCP tools include internal diagnostic and debugging facilities that may be activated by run-time flags.
`pmerr`	Translates PCP error codes into human-readable error messages.

`pmhostname`	Reports hostname as returned by `gethostbyname`. Used in assorted PCP management scripts.
`pmie_check`	Administration of the Performance Co-Pilot inference engine (`pmie`).
`pmlaunch`	Contains metrics specification formats and a set of scripts for use by tools that are launching, and being launched by, other tools with no knowledge of each other. It is a configuration directory.
`pmlock`	Attempts to acquire an exclusive lock by creating a file with a mode of 0.
`pmlogger_*`	Allows you to create a customized regime of administration and management for PCP archive log files. The `pmlogger_check`, `pmlogger_daily`, and `pmlogger_merge` scripts are intended for periodic execution via the `cron` command.
`pmnewlog`	Performs archive log rotation by stopping and restarting an instance of `pmlogger`.
`pmnsadd`	Adds a subtree of new names into a PMNS, as used by the components of PCP.
`pmnscomp`	Compiles a PMNS in ASCII format into a more efficient binary representation.
`pmnsdel`	Removes a subtree of names from a PMNS, as used by the components of the PCP.
`pmnsmerge`	Merges multiple PMNS files together, as used by the components of PCP.
`pmpost`	Appends the text message to the end of the PCP notice board file (`/var/adm/pcplog/NOTICES`).
`pmrun`	Is a graphical utility for launching PCP commands with optional arguments from the IRIX desktop.
`pmsnap`	Creates performance snapshots suitable for Web publishing from PCP archives using `pmsnap`. The `pmsnap` script is intended for periodic execution via the `cron` command.

`pmstore`	Reinitializes counters or assigns new values to metrics that act as control variables. The command changes the current values for the specified instances of a single performance metric.

Installing and Configuring Performance Co-Pilot

The sections in this chapter describe the basic installation and configuration steps necessary to run Performance Co-Pilot (PCP) on your systems. The following major sections are included:

- Section 2.1, describes the main packages of PCP software and how they must be installed on each system.
- Section 2.2, page 12, describes the licensing issues necessary to operate PCP in a distributed computing environment.
- Section 2.3, page 13, describes where to find information on installing and testing PCP.

2.1 Product Structure

In a typical deployment, Performance Co-Pilot (PCP) would be installed in a collector configuration on one or more hosts, from which the performance information could then be collected, and in a monitor configuration on one or more workstations, from which the performance of the server systems could then be monitored.

PCP is presented as two sets of products; the foundation product components that are provided as part of the IRIX operating system (`pcp_eoe`) and the enhancement product components that are available to add to the existing PCP distribution (`pcp` and `pcp_gifts`). To install and use the `pcp` subsystems, `pcp_eoe` must be installed from your IRIX operating system distribution CDs. The `pcp_eoe` and `pcp` modules can be installed at the same time, but `pcp_eoe` is a prerequisite for `pcp` to operate.

PCP is packaged into a number of basic subsystem types that reflect the functional role of the product components. These subsystems may be installed using the `inst` or `swmgr` command:

Core — The `pcp_eoe.sw.eoe` and `pcp.sw.base` subsystems must be installed on every PCP enabled host, that is, on both PCP monitor and PCP collection systems.

Monitor — The `pcp_eoe.sw.monitor` and `pcp.sw.monitor` subsystems must be installed on every PCP monitor host. Subsystems `pcp_eoe.books.help` and `pcp.books.help` should be installed to provide help

	support for the GUI monitoring tools; see the `sgihelp`(1) man page.
Collector	No additional installation is required because the Performance Metrics Collection Daemon (`pmcd`) is in the `pcp_eoe.sw.eoe` subsystem.
Demo	The `pcp.sw.demo` subsystems provide source code for example applications and PMDAs that serve as templates for developing new modules to extend the PCP coverage of performance metrics or the capabilities of monitoring tools.
Other	The other `pcp.sw.*` subsystems provide the support for the optional PMDAs, and when required, need to be installed on the PCP collector host, and subsequently configured before they become active.
Gift	The `pcp_gifts.sw.*` subsystems provide optional applications and services that may be individually installed as required.
Documentation	The `pcp.man.*` and `pcp.books.*` subsystems provide release notes, man pages, interactive tutorials, and IRIS InSight books, and may be installed as needed.

For complete information on the installable software packages, see the Performance Co-Pilot release notes. For additional information, see the `relnotes`(1) or `grelnotes`(1) man pages.

2.2 License Constraints

On Performance Co-Pilot (PCP) monitoring systems, all of the display, visualization, and automated reasoning tools are licensed using “nodelocked” FLEXlm licenses. On PCP collection systems, the Performance Metrics Collection Daemon (PMCD) is also licensed using “nodelocked” FLEXlm licenses. Refer to the PCP release notes for details.

The other PCP tools and services (for example, the Performance Metrics Domain Agents (PMDAs) or `pmlogger`) may be installed and executed without license constraints.

Some of the PCP maintenance tools for updating the Performance Metrics Name Space (PMNS), interrogating the Performance Metrics Collection Subsystem (PMCS), dumping an archive log, and so on, are not constrained by any license restrictions.

The `pmbrand` command manages the `/var/pcp/pmns/Brand` file, which contains binary information about PCP capabilities enabled by the various valid licenses on the system. If you are unsure of the license status for a particular host, `pmbrand` verifies and prints the current license information on that system, producing output similar to the following:

```
/usr/pcp/bin/pmbrand -l
Licenses for system 690794d70
     PCP Collector
     PCP Monitor
```

2.3 Installing and Testing Performance Co-Pilot

For more information on installing and testing PCP, see Chapter 2, "Installing and Configuring Performance Co-Pilot" in *Performance Co-Pilot for IRIX User's and Administrator's Guide*.

Common Conventions and Arguments

This chapter deals with the user interface components that are common to most of the graphical tools and text-based utilities that make up the monitor portion of Performance Co-Pilot (PCP).

This chapter only contains information pertaining to features which are introduced by installing the `pcp` product. For additional information about common commands and arguments used by the PCP suite of tools (including those in pcp_eoe), see Chapter 3, "Common Conventions and Arguments" in *Performance Co-Pilot for IRIX User's and Administrator's Guide*.

These are the major sections in this chapter:

- Section 3.1, page 16, shows a picture of the **PerfTools** icons.
- Section 3.2, page 16, details other options to use with PCP tools.
- Section 3.3, page 19, describes some of the environment variables supported by PCP tools.
- Section 3.4, page 19, describes how to execute PCP tools that must retrieve performance data from the Performance Metrics Collection Daemon (PMCD) on the other side of a TCP/IP security firewall.

Many of the utilities provided with PCP conform to a common set of naming and syntactic conventions for command line arguments and options. This section outlines these conventions and their meaning. The options may be generally assumed to be honored for all utilities supporting the corresponding functionality.

In all cases, the man pages for each utility fully describe the supported command arguments and options.

Command line options are also relevant when starting PCP applications from the desktop using the `Alt` double-click method. This technique launches the `pmrun` program to collect additional arguments to pass along when starting a PCP application.

3.1 PerfTools Icon Catalog

The conventions and arguments described in this chapter are common to all tools and utilities in the **PerfTools Icon Catalog** group, shown in Figure 3-1.

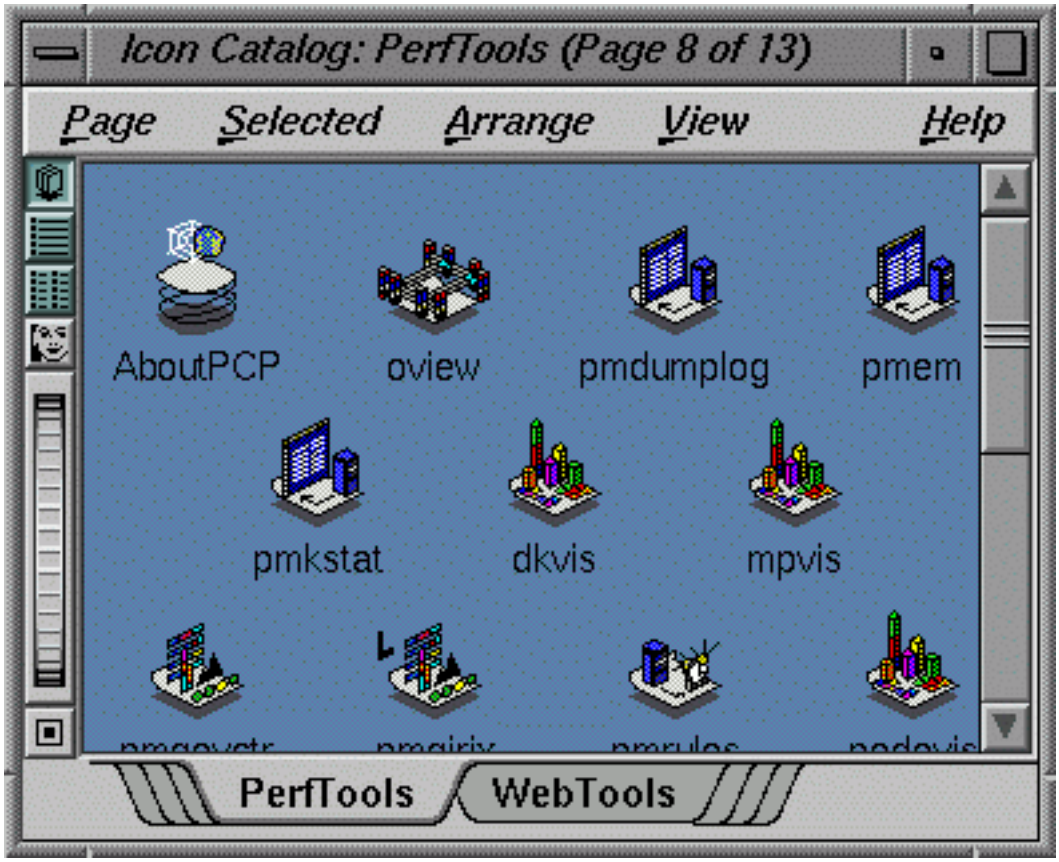

Figure 3-1 PerfTools Icon Catalog Group

3.2 General PCP Tool Options

The following sections provide information relevant to most of the PCP tools. It is presented here in a single place for convenience.

3.2.1 Common Directories and File Locations

The following files and directories are used by the PCP tools as repositories for option and configuration files and for binaries:

`/etc/pcp.env`

Script to set PCP run-time environment variables.

`/etc/pcp.conf`

PCP configuration and environment file.

`/etc/pmcd.conf`

Configuration file for Performance Metrics Collection Daemon (PMCD). Sets environment variables, including `PATH`.

`/usr/etc/pmcd`

The PMCD binary.

`/etc/config/pmcd.options`

Command line options for PMCD.

`/etc/config/pmlogger.options`

Command line options for `pmlogger` launched from `/etc/init.d/pcp`.

`/etc/init.d/pcp`

The PMCD startup script.

`/usr/sbin`

Directory containing PCP tools such as `pmkstat`, `pminfo`, and `oview`.

`/usr/pcp`

Directory containing shareable PCP-specific files and repository directories.

`/var/pcp`

Directory containing non-shareable (that is, per-host) PCP specific files and repository directories. There are some symbolic links from the `/usr/pcp` directory hierarchy pointing into the `/var/pcp` directory hierarchy.

`/usr/pcp/bin`

PCP tools that are typically not executed directly by the end user such as `pmbrand`, `pmnscomp`, and `pmlogger`.

`/usr/pcp/lib`

Miscellaneous PCP libraries and executables.

`/var/pcp/pmdas`

Performance Metric Domain Agents (PMDAs), one directory per PMDA.

`/usr/pcp/pmdas`

An alternate repository for some PMDAs. Certain entries here are symbolic links into `/var/pcp/pmdas`.

`/var/pcp/config`

Configuration files for PCP tools, typically with one directory per tool.

`/usr/pcp/demos`

Demonstration data files and example programs.

`/var/pcp/Tutorial`

PCP Tutorial, in HTML format.

`/var/adm/pcplog`

By default, diagnostic and trace log files generated by PMCD and PMDAs. Also, the PCP archive logs are managed in one directory per logged host below here.

`/var/pcp/pmns`

Files and scripts for the Performance Metrics Name Space (PMNS).

3.3 PCP Environment Variables

When you are using PCP tools and utilities and are calling PCP library functions, a standard set of defined environment variables are available in the `/etc/pcp.conf` file. These variables are generally used to specify the location of various PCP pieces in the file system and may be loaded into shell scripts by sourcing the `/etc/pcp.env` shell script. They may also be queried by C and C++ programs using the `__pmGetConfig` library function. If a variable is already defined in the environment, the values in the `pcp.conf` file do not override those values; that is, the values in `pcp.conf` serve only as installation defaults. For additional information, see the `/etc/pcp.conf`(4), `/etc/pcp.env`(4), and `__pmGetConfig` man pages.

The following environment variables are recognized by PCP (these definitions are also available on the `PCPIntro`(1) man page):

`PCP_LICENSE_NOWARNING`

Many of the PMAPI client programs require that a valid software license be present on the host on which the client is running (the license is node-locked). In the case that such a valid license is present, but is due to expire within the next 30 days, a message or pop-up notifier appears informing the user of this condition. These warnings can be disabled by setting this variable in the environment.

Other variables are available and are explained in Chapter 2, "Installing and Configuring Performance Co-Pilot" in the *Performance Co-Pilot User's and Administrator's Guide*.

3.4 Running PCP Tools through a Firewall

In some production environments, the Performance Co-Pilot (PCP) monitoring hosts are on one side of a TCP/IP firewall, and the PCP collector hosts may be on the other side.

If the firewall service is being provided by a product that supports the `sockd` (SOCKS) protocols for packet forwarding through the firewall, then the PCP tool

pmsocks may be used; see the pmsocks(1) man page. Otherwise, it is necessary to arrange for packet forwarding to be enabled for those TCP/IP ports used by PCP, namely 4321 (or the value of the PMCD_PORT environment variable) for connections to PMCD and a finite range of consecutive port numbers starting at 4330 (or the value of the PMLOGGER_PORT environment variable) to allow pmlc connections to pmlogger instances.

3.4.1 The pmsocks Command

The pmsocks command and its related files and scripts allow PCP clients running on hosts located on the internal side of a TCP/IP sockd firewall system to monitor remote hosts on the other side of the firewall system. The basic syntax is as follows, where *tool* is an arbitrary PCP application, typically a monitoring tool:

```
pmsocks tool args
```

The pmsocks script prepares the necessary environment variables and then executes the PCP tool specified in *tool* across the firewall. For example, this command runs dkvis with metrics fetched from remotehost on the other side of the firewall:

```
pmsocks dkvis -h remotehost
```

The configuration file is/etc/pcp_socks.conf, and the network-specific information in this file is set to correspond with your network. Complete information on this customization can be found in the pmsocks(1) man page.

Monitoring System Performance

This chapter describes the performance monitoring tools available in Performance Co-Pilot (PCP). This product provides a group of commands and tools for measuring system performance. Each tool is described completely by its own man page. The man pages are accessible through the `man` command. For example, the man page for the tool `pmchart` is viewed by entering the following command:

```
man pmchart
```

The following major sections are covered in this chapter:

- Section 4.1, page 21, describes `pmchart`, a useful charting tool that graphically monitors system performance.
- Section 4.2, page 42, presents `pmgadgets`, a graphical tool that displays system performance in a small area.
- Section 4.3, page 46, discusses `pmdumptext`, a utility that shows the current values for named performance metrics.

Further monitoring tools covering performance visualization and automated reasoning about performance are described in Chapter 5 and Chapter 6.

The following sections describe the various graphical and text-based PCP tools used to monitor local or remote system performance.

Other tools and commands available in PCP are described in Chapter 3, “Common Conventions and Arguments”, in the Performance Co-Pilot User’s and Administrator’s Guide.

4.1 The `pmchart` Tool

The `pmchart` utility supports interactive selection and plotting of trends over time for arbitrarily selected performance metrics from one or more hosts and one or more domains of performance metrics. First, you enter the following command:

```
pmchart
```

You then see the **Performance Co-Pilot Chart** window shown in Figure 4-1.

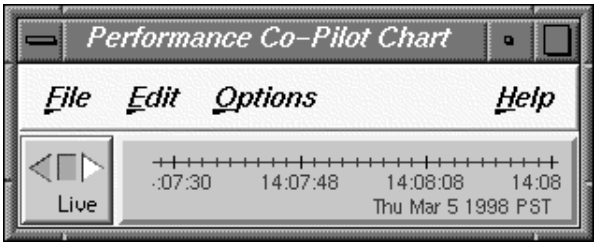

Figure 4-1 `pmchart` **Performance Co-Pilot Chart** Window

Normally, `pmchart` operates in *live* mode where performance metrics are fetched in real time and plotted against a time axis. The user can choose performance metrics and monitor the current values for these metrics from any host that is accessible on the network and has the PMCD server running.

When launched with the `-a` command line option, `pmchart` can also replay PCP archive logs of performance metrics created by `pmlogger`.

The man page for `pmchart` explains how to configure charts based on performance metrics, using either the **Open View** option of the **File** menu or the **New Plot** option of the **File** menu. Once charts have been configured and applied, the charts are placed in an expanded **Performance Co-Pilot Chart** window, as shown in Figure 4-2.

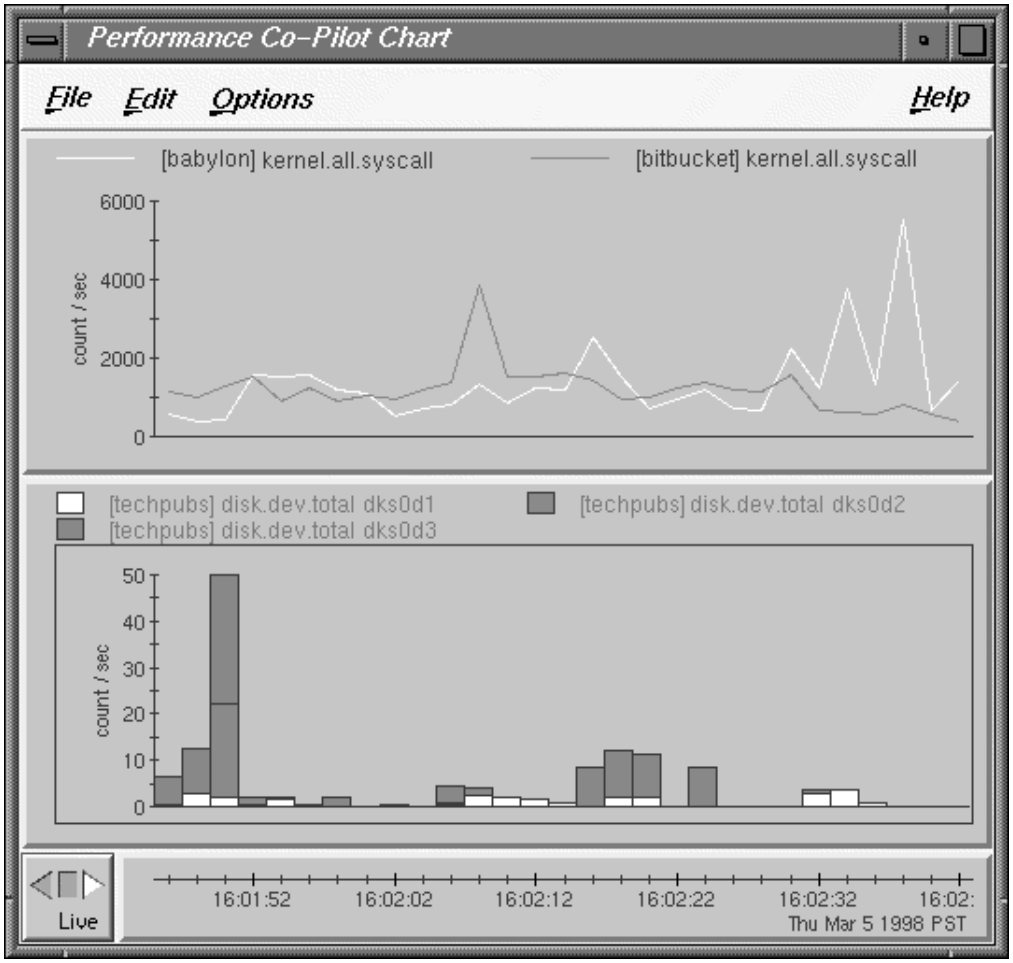

Figure 4-2 Two Charts and Metrics from Three Hosts in `pmchart`

All metrics in the Performance Metrics Name Space (PMNS) with numeric value semantics can be graphed. By default, `pmchart` initially allows the user to select metrics to be plotted from the local host. However, the graphical user interface allows other hosts or archives to be chosen at any time as alternate sources of performance metrics and all metrics (independent of their source) are plotted on a common time axis.

For horizontal lines at major tick marks, see Section 4.1.3, page 28.

The `-h` command line option nominates an alternate default host to be used in preference to the local host.

The `-a` command line option may be used to start `pmchart` processing performance metrics from one or more PCP archive logs. The first named archive becomes the default source of performance metrics. This mode is particularly useful for retrospective comparisons and for postmortem analysis of performance problems, where a remote system is not directly accessible or a performance analyst is not available on site.

The `pmchart` utility examines the semantics of selected metrics, and where sensible, uses the metadata provided by the Performance Metrics Collection Subsystem (PMCS) to convert fetched metric values to a rate before plotting. In the case where different metrics are plotted in the same chart (for example, against a common Y-axis), the metrics must have the same dimension (taking into account any automatic rate conversion), but `pmchart` may scale metric values where necessary, to produce comparable values with common units and scale.

When replaying archive logs, the user may interactively control the current replay time, direction of replay, and replay rate, using the PCP time control dialog.

4.1.1 Mouse Controls

The `pmchart` tool uses the mouse buttons as follows:

Left	The primary mouse button may be used to select the current chart by clicking anywhere in a specific chart. The current chart always has a border drawn around the graph area and its legend of metric names rendered in red. The **Edit** menu contains a variety of choices that operate only on the current chart. This mouse button also interacts with menus and dialog boxes in the usual manner.
Middle	The middle mouse button is unused.
Right	The secondary mouse button may be used to display metric values in a dialog box. Click this mouse button in the graph drawing area of any chart to display information about the nearest metric and its value at that point as plotted. The **Metric Value Information** dialog box remains visible until you dismiss it, and can

be refreshed with new metric values by clicking this mouse button again, or updated automatically using the **Show most Recent** toggle button.

4.1.2 `pmchart` Select Performance View

A *view* in `pmchart` is a predefined collection of charts, typically constructed to display some common performance scenario. Default views are included in the PCP distribution, others are part of the various PCP add-on products, and others may be created by the `pmchart` end user. The **Open View...** option in the **File** menu launches a **Select Performance View** dialog box similar to Figure 4-3.

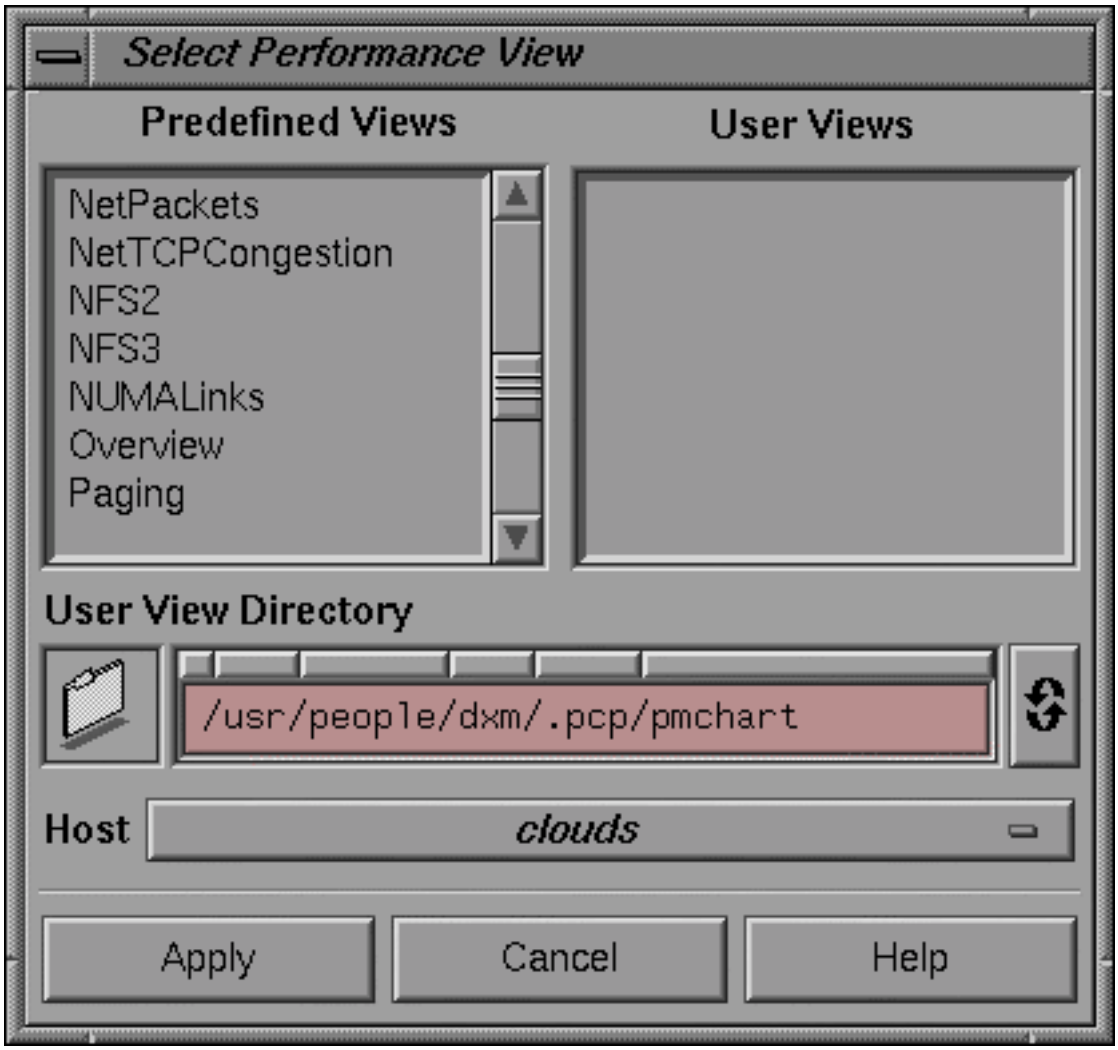

Figure 4-3 `pmchart` **Select Performance View** Dialog

You may use this dialog to select one of the available views. The default PCP views include the following:

`BufferCache` Cumulative amount of data read and written between system buffers and user memory or block devices.

`CPU`	Processor utilization (user, system, memory break, interrupt, I/O wait, and idle time) aggregated over all CPUs.
`NUMAlinks`	Usage of NUMAlink node connectors, if this hardware is present.
`Disk`	Cumulative number of read and write transfers for all disk devices.
`DiskCntrls`	Cumulative number of read and write transfers for all drives attached to each disk controller on the system.
`FileSystem`	Percentage of each filesystem in use (percent full).
`LoadAvg`	System load averaged over intervals of 1, 5, and 15 minutes.
`Memory`	Memory used by the kernel, filesystems, user processes, and free space.
`NetBytes`	Network interface activity—octets transmitted on various interfaces.
`NetConnDrop`	TCP drops, connection drops, timeout drops, and TCP accepts.
`NetPackets`	Rate of TCP and UDP packets received and sent.
`NetTCPcongestion`	TCP packets retransmitted, retransmit timeouts, and TCP packets sent.
`NFS2, NFS3`	Client and server NFS operation rates.
`Overview`	Composite charts of CPU, LoadAvg, Memory, Disk, and NetBytes.
`Paging`	Page-in and page-out rates from the virtual memory subsystem.
`PMCD`	Message rates and CPU time used by PMCD or associated PMDAs.
`Swap`	System swap space allocated, reserved, and unused.
`Syscalls`	Rate of `exec`, `fork`, `read`, `write`, and total system calls.

You can create your own custom views using the metric selection facilities, and save your views for later using the **Save View...** option of the **File** menu.

4.1.3 Displaying Horizontal Lines

You can have `pmchart` display horizontal lines, usually in a lighter background color, at major tick marks by calling `pmchart` with the following arguments (quotes required):

```
% pmchart -xrm "PmChart*xrtYGridUseDefault: True"
```

For greater convenience, you can place the following line in your `$HOME/.Xresources` file, to have `pmchart` always display horizontal lines:

```
PmChart*xrtYGridUseDefault: True
```

4.1.4 `pmchart` Metric Selection

The `pmchart` **Metric Selection** window, shown in Figure 4-4, allows interactive navigation of the Performance Metrics Name Space (PMNS) to create new chart configurations.

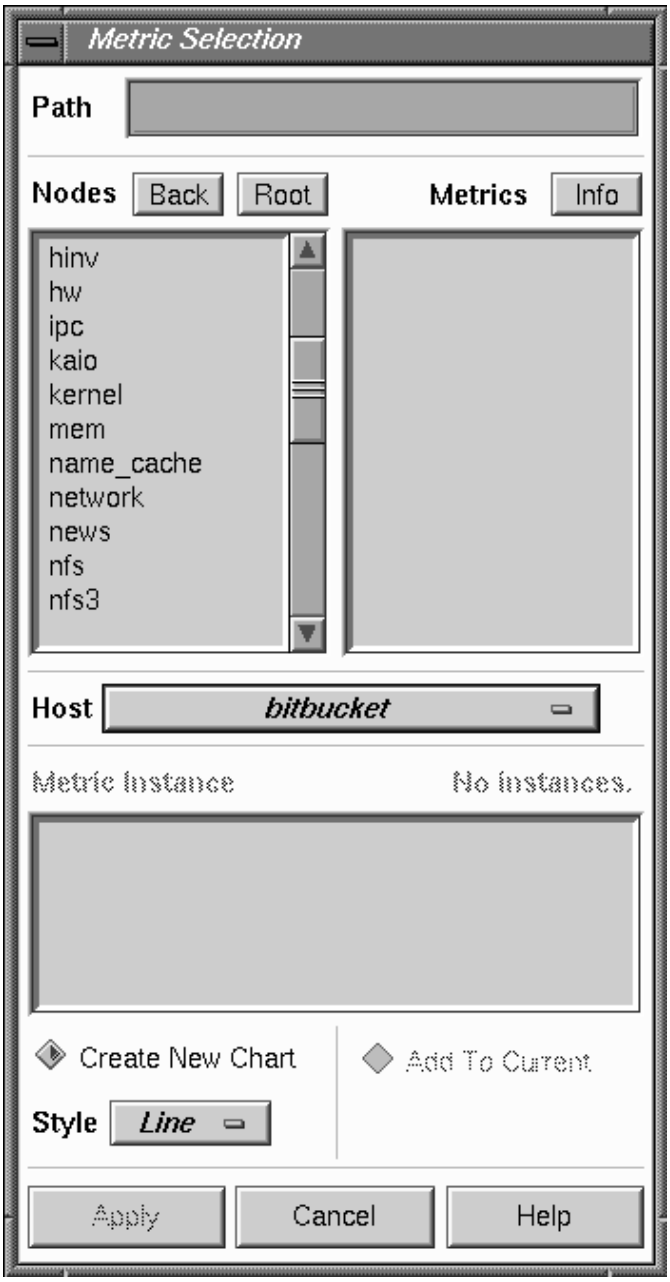

Figure 4-4 `pmchart` **Metric Selection** Dialog

You can choose metrics, display information about metrics, change the current host or archive, select metric instances, and plot metric values on a common time axis. You bring up this window by choosing **New Plot...** from the **File** menu of `pmchart`.

Metric selection proceeds by navigating through the tree-structured PMNS. If you enter a partial metric specification in the **Path** field in the **Metric Selection** dialog, you can avoid having to navigate through the PMNS for the metrics you need. For example, if you enter `network.interface`, the window changes dynamically, as shown in Figure 4-5.

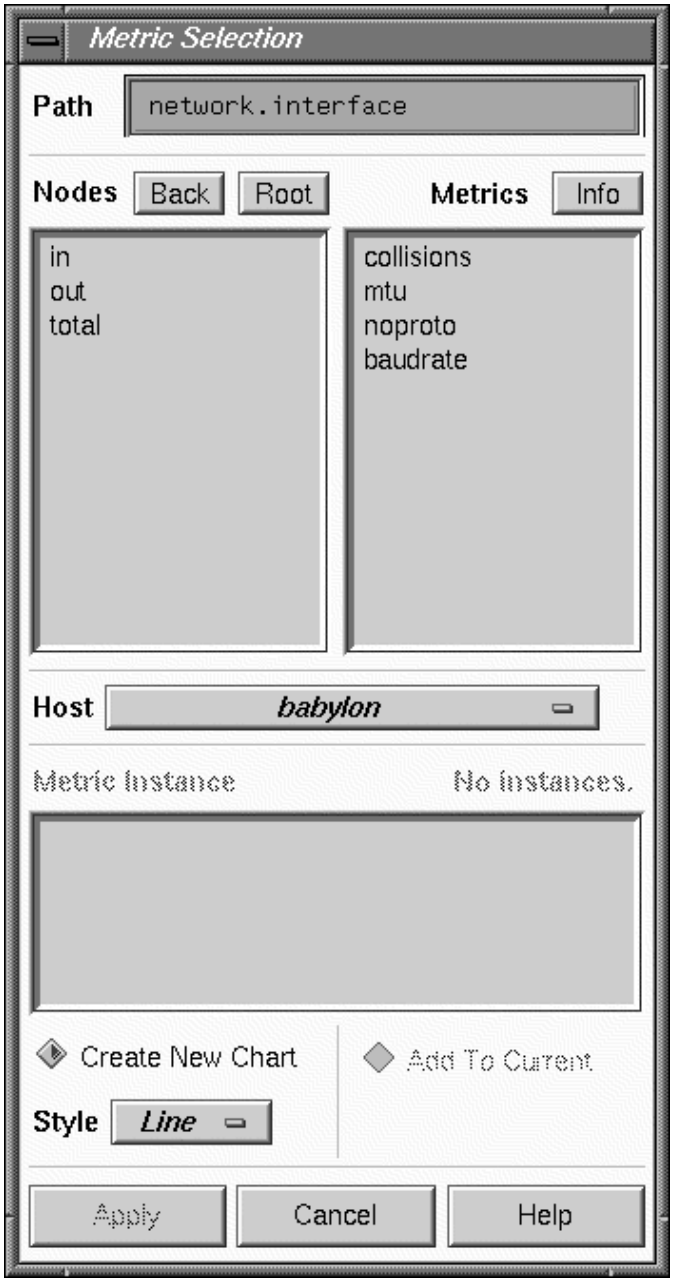

Figure 4-5 Further **Metric Selection**

You can continue the selection process by choosing non-leaf nodes from the **Nodes** list, and finally a leaf node from the **Metrics** list. At this stage, the **Path** corresponds to a leaf node in the PMNS, as shown in Figure 4-6.

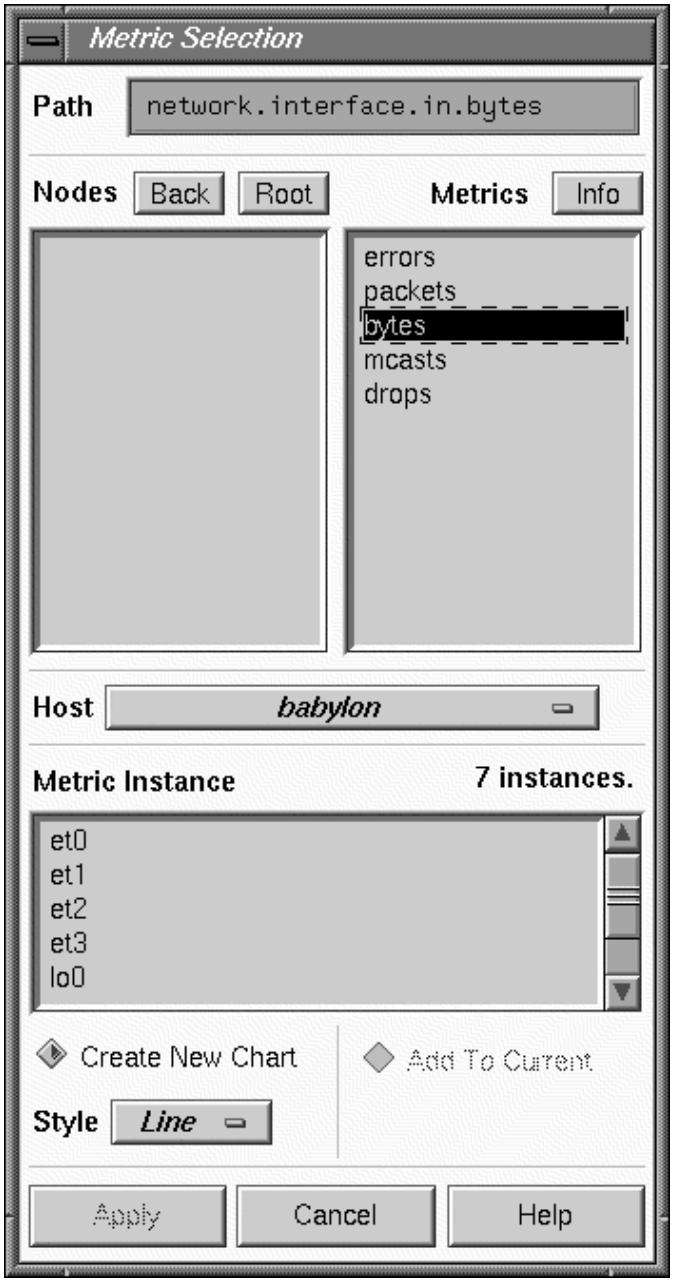

Figure 4-6 Selecting a Leaf Node in the PMNS (Performance Metric)

Once a metric has been selected, the **Info** button in the **Metric Selection** dialog launches the **Metric Information** dialog, as shown in Figure 4-7.

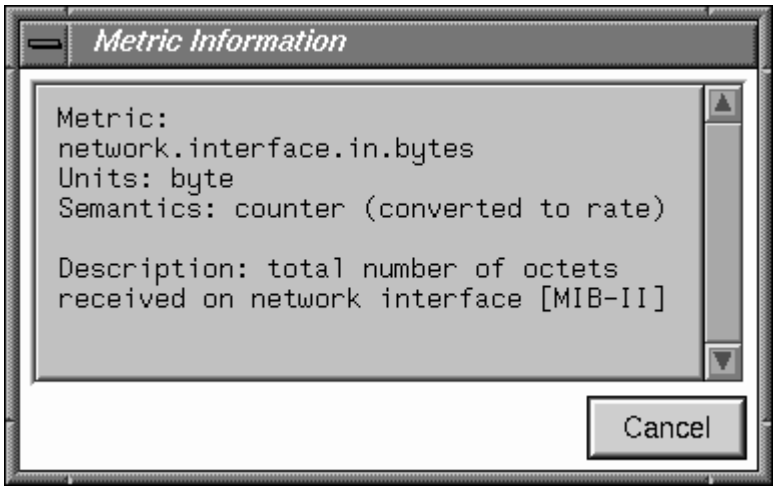

Figure 4-7 Metric Information Dialog

This dialog displays the name, unit, and semantics for the currently selected metric, along with the verbose help text that describes the metric, and optionally a description of the underlying instance domain.

Finally, you may have to select from several instances of a metric. In the example shown in Figure 4-7, you wish to monitor the input packet rate for some network interface(s). For the current source of performance metrics, there are two network interfaces configured. You must select one or more instances, as shown in Figure 4-8.

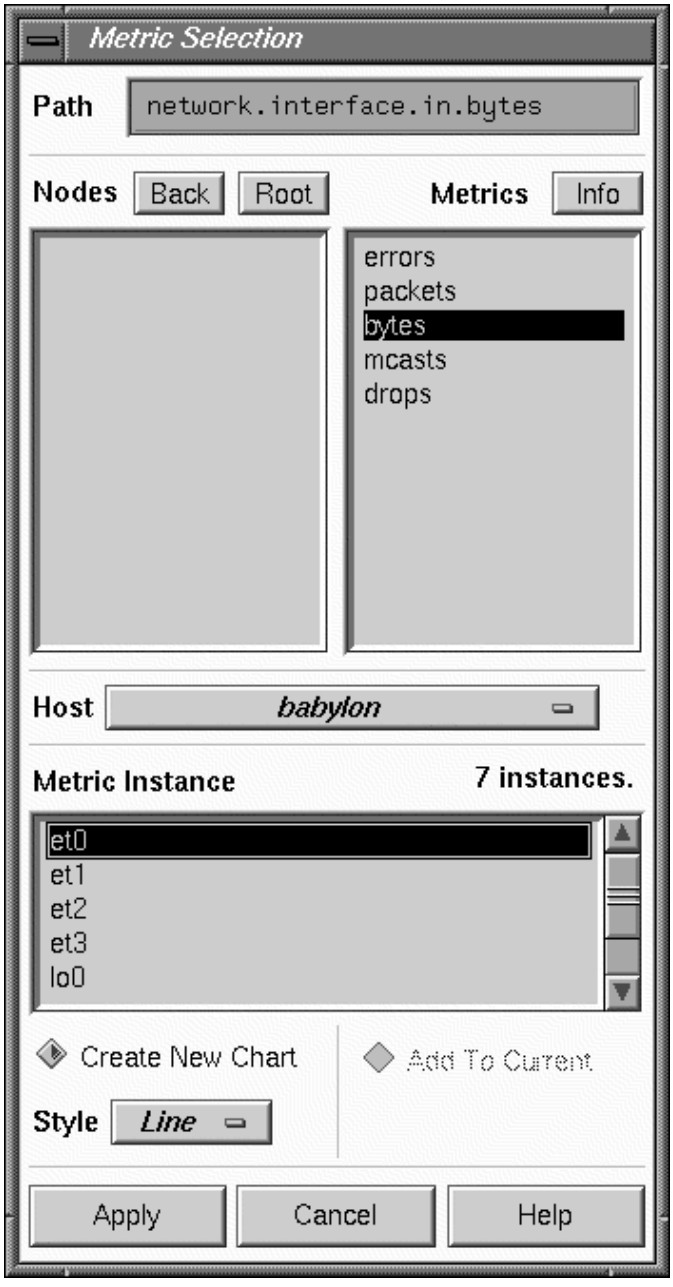

Figure 4-8 Selecting a Metric Instance

You can select multiple instances either by clicking and dragging up and down the list with the left mouse button, or by selecting the first instance and then using the `Shift` key (or `Ctrl` key) with the left mouse button to select one or more other instances.

4.1.5 Creating a PCP Archive from a `pmchart` Session

From the **File** menu of `pmchart` when running in live mode, the `Record` (`Stop Recording`) option may be used to start (or stop) the creation of a PCP archive log. The archive log is created using `pmlogger` and includes the update interval and all of the performance metrics in the current `pmchart` configuration when recording begins.

Note: Any changes made to the `pmchart` configuration after recording has been started will not be reflected in the archive log. For these to take effect, the recording must be stopped and restarted (thereby creating a second PCP archive log).

When recording is started, a **File Chooser** dialog is launched, and the user must provide the name of a new file to be used as the PCP archive folio for the new archive. The recording session produces multiple files in the same directory as the archive folio.

If necessary, `pmchart` creates directories on the path to the named archive folio.

It is often convenient to maintain one directory for each new folio, or else one directory for each group of folios related by collector host(s), service type, or chart selection.

When recording is active, a small red indicator appears in the time control button, as shown at the bottom left of Figure 4-9.

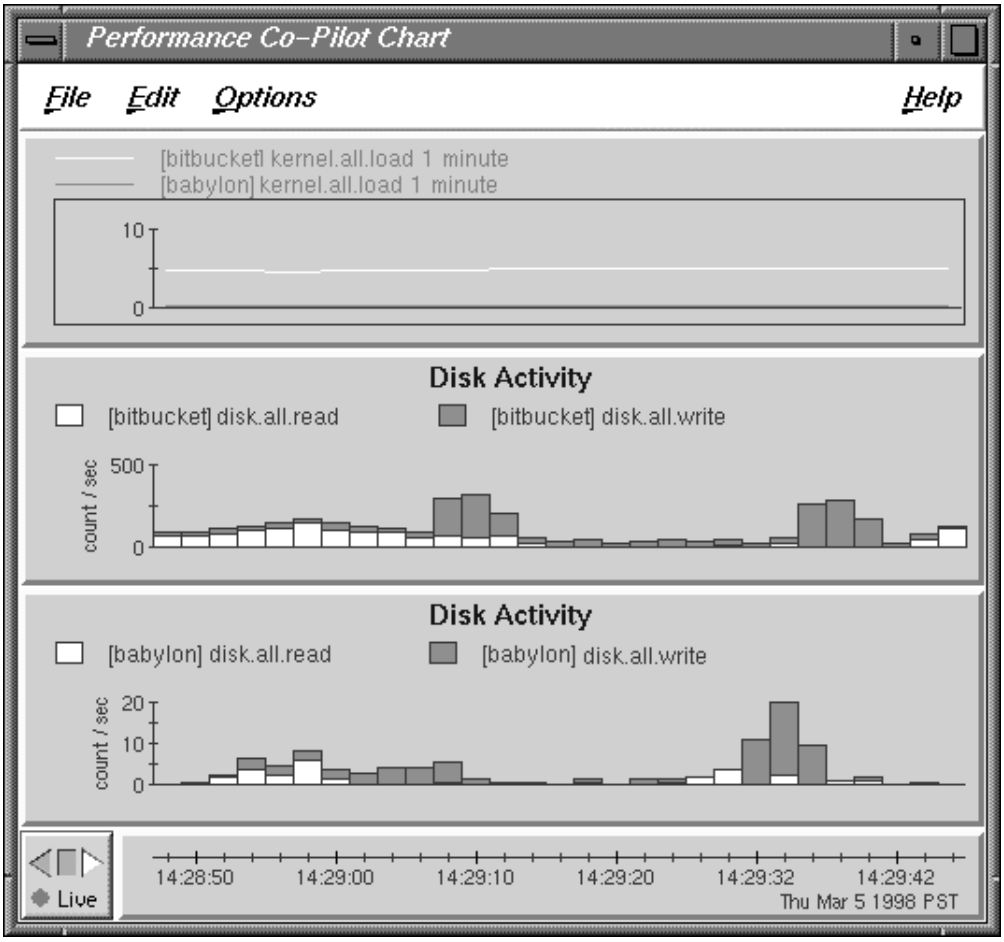

Figure 4-9 `pmchart` Display When Recording

If you choose **File > Stop Recording**, logging stops immediately. The red light in the lower left turns gray.

To start recording again, chose **File > Record** and specify a new archive folio name.

If you exit `pmchart` by choosing **File > Quit**, an **Archive Recording Session-pmchart** dialog similar to that shown in Figure 4-10 appears to remind you where the archive folio was created, and to confirm that recording should be terminated.

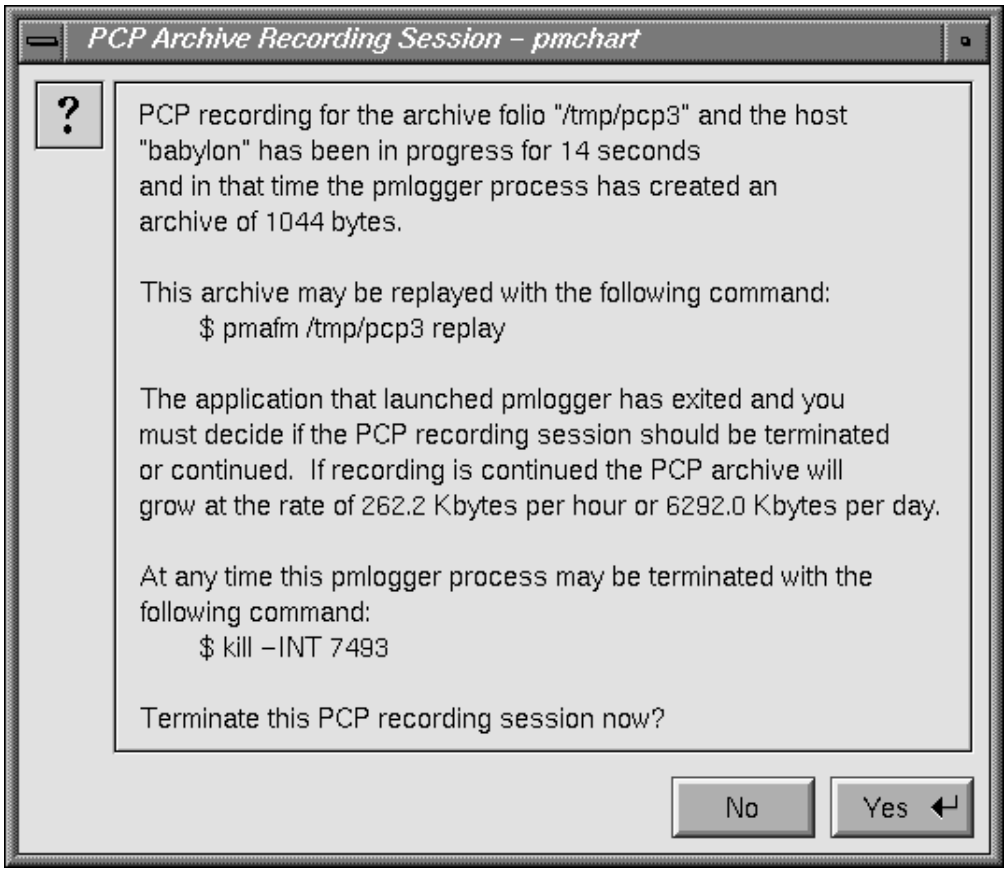

Figure 4-10 Archive Recording Session-pmchart Dialog

If you select **Yes**, recording stops immediately.

If you select **No**, recording continues. This is a useful way to continue archive logging without keeping `pmchart` active.

4.1.6 Changing `pmchart` Colors

When using a video projector, or when making presentations to a large group, or as a result of personal preference, the default pastel color scheme used by `pmchart` may be inappropriate.

The **Colors** option in the **Edit** menu allows arbitrary changes to the colors of individual charts. For more global changes, you can override the defaults using the X11 resources that `pmchart` honors.

For example, create or add the following entries in the `$HOME/.xrdb` file:

```
PmChart*xrtForegroundColor: "green"
PmChart*xrtBackgroundColor: "black"
PmChart*xrtGraphForegroundColor: "rgb:00/b0/00"
PmChart*xrtGraphBackgroundColor: "black"
PmChart*xrtHeaderForegroundColor: "green"
PmChart*xrtHeaderBackgroundColor: "black"
PmChart*pmDefaultColors: rgb:ff/ff/00 rgb:00/ff/00 rgb:00/00/ff \
                           rgb:ff/ff/00 rgb:00/ff/ff rgb:ff/00/ff
```

Use the following command to change the default color scheme for `pmchart` to one with bright primary colors on a black background:

```
xrdb -merge $HOME/.xrdb
```

4.1.7 Other Chart Customizations

The `pmchart` **Edit** menu provides options and a dialog that you may use to change and customize the display as follows:

Chart Style	Chooses from line, bar, stacked bar, area plot, and utilization.
Chart Title and Legend...	Changes the chart title, and enable or disable the legend annotation at the top of each chart.
Y-Axis Scaling...	Fixes the maximum and minimum values of the range on the Y-axis, or allow `pmchart` to adjust the range dynamically to reflect currently displayed values.
Colors...	Customizes plot colors.
Delete	Selects all charts, a complete chart, or individual plots from a chart.

The `pmchart` **Options** menu provides another option for customizing the display:

`Visible Points...`	Uses the slider to change the number of values along the time axis.

4.1.8 Time Control

The **Options** menu provides access to the **PCP Time Control** Dialog.

Show Time Control	Exposes the dialog for the controlling `pmtime` instance, thereby allowing users to change the sampling interval. Selecting the **Time Control** button in the lower left corner of the main `pmchart` window also exposes the **Time Control** dialog. If the current source of performance metrics is one or more PCP archive logs, this same dialog may be used for temporal navigation within the archive(s).
New Time Control	Detaches `pmchart` from the controlling `pmtime` instance and launches a new `pmtime` instance, initially dedicated to this `pmchart`.
Launch New Pmchart	Starts a new `pmchart`, with shared `pmtime` control.

4.1.9 Taking Snapshots of `pmchart` Displays and Value Dialogs

The **Print** option in the **File** menu enables the current `pmchart` display to be printed in a variety of PostScript styles. The output can be saved in a file or sent directly to a printer.

The `-o` option for `pmchart` also provides the facility to produce Graphics Interchange Format (GIF) image snapshots of the `pmchart` display.

It is often convenient to publish performance summary information for the users of a particular computing environment. The `pmchart` tool, in combination with the `pmsnap` script and its associated control files, can be used to produce high-quality performance summary snapshot images in GIF format. These images can be incorporated into Web pages, reports, e-mail, or presentation material.

The following files and utilities are included in support of this feature:

`/var/pcp/config/pmsnap/Summary`

This file contains a summary of the performance metrics used in the example snapshot.

`/var/pcp/config/pmsnap/Summary.html`

An example HTML page suitable for publishing images from the Summary `pmsnap` example via a Web server.

`/var/pcp/config/pmsnap/control`

This file controls the snapshot parameters.

`/var/pcp/config/pmlogger/config.Summary`

This configuration file specifies an archive log suitable for use with any `pmview`-type tool, and the example Summary snapshot configuration.

`/usr/pcp/bin/pmsnap`

The `pmsnap` script is designed to be periodically run by the `cron` command to process the control file `/var/pcp/config/pmsnap/control` and generate snapshot images according to the specifications therein. The `pmsnap`(1) man page describes the command line options for selecting the control lines to process, the default directory for the output files, the `X` display to use, and other parameters.

Instructions for configuring `pmsnap` are in the man page. There is also a verbose comment at the head of the control file. The `pmchart`(1) man page is also useful.

4.1.10 More Information

The annotated examples in the `pmchart` chapter of the *PCP Tutorial* provide a guided illustration to a typical user's interactions with `pmchart`. The *PCP Tutorial* can be optionally installed as the `pcp.man.tutorial` subsystem. To view the `pmchart` chapter of the tutorial, open the following URL with your Web browser:

`file:/var/pcp/Tutorial/pmchart.html`

4.2 The pmgadgets Command

The pmgadgets tool creates a small window containing a collection of graphical gadgets driven by performance metrics supplied by the PCP framework. Any numeric metric supported by PCP can be displayed.

Note: In the current PCP release, pmgadgets is constrained to process performance metrics from real-time sources (and not PCP archive logs), although metrics from several different hosts may be displayed simultaneously in the same window.

The layout of the gadgets and the performance metrics that lie behind them are specified in a configuration file, and pmgadgets is typically run on an existing configuration file or in conjunction with an application that automatically generates a configuration file. For example, pmgsys generates a configuration file for various IRIX performance metrics and feeds it to pmgadgets. The resulting display depends on the host configuration, but the display shown in Figure 4-11 is representative of a system with four CPUs, eleven disks on three controllers, and four network interfaces.

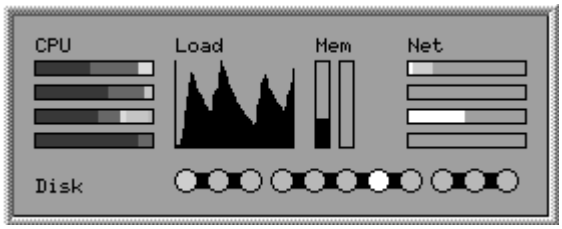

Figure 4-11 Representative pmgadgets Display Using pmgsys

Other pmgadget front end tools such as pmgcluster, pmgevctr, pmgcisco, and pmgshping are not described in this chapter. For information about these tools, see the pmgcluster(1), pmgevctr(1), pmgcisco(1), and pmgshping(1) man pages.

The pmgadgets tool displays much the same information as pmchart, but more compactly, and with less historical information.

The pmgadgets specification language provides the ability to define the following gadgets and components:

_bar — Displays a single performance metric value as a rectangle. This rectangle is filled from left to right or

	bottom to top in proportion to the ratio of the metric's value to some maximum.
`_multibar`	Is similar to the bar gadget, but displays several performance metrics at the same time (same as stacked bar). Each is allocated a color and the gadget's rectangle is filled with an amount of each color proportional to the ratio of the corresponding performance metric's contribution to a maximum value.
`_bargraph`	Displays a simple `xload` style strip chart of a performance metric's values over time.
`_led`	Is a circular gadget whose color is modulated using the value of a single performance metric.
`_line`	Is a solid rectangle, not modulated by any performance metric, useful for highlighting connectivity between gadgets.
`_label`	Provides textual annotation in the display.
`_actions`	Provides customized menus of "drill-down" actions that may be associated with any gadget. Using the right mouse button over a visible gadget causes any associated action menu to pop up.
`_colorlist`	Provides a list of X11 color specifications.
`_legend`	Provides the association between color and range of performance metric values for use in a `_led` gadget.

Each visible gadget must be assigned a Cartesian position in the `pmgadgets` display.

By way of an example, the `pmgadgets` specification shown in Example 4-1 includes CPU, disk, and load average information from two hosts, and produces a customized `pmgadgets` display like the one shown in Figure 4-12.

Example 4-1 Specification File for `pmgadgets`

```
_colourlist cpuColours (blue3 red3 yellow3 cyan3 green3)
_legend diskLegend (
       _default  green3
       15        yellow
       40        orange
       75        red
```

```
)
# host moomba
_label 70 12 "moomba"
_multibar 5 5 30 6
       _metrics (
           moomba:kernel.all.cpu.user
           moomba:kernel.all.cpu.sys
           moomba:kernel.all.cpu.intr
           moomba:kernel.all.cpu.wait.total
           moomba:kernel.all.cpu.idle
       )
       _maximum 0.0
       _colourlist cpuColours
_bargraph 40 5 25 20
       _metric moomba:kernel.all.load["1 minute"]
       _max 1.0
_led 12 16 6 6
       _metric moomba:disk.all.read _legend diskLegend
_led 25 16 6 6
       _metric moomba:disk.all.write _legend diskLegend
# host gonzo
_label 70 39 "gonzo"
_multibar 5 32 30 6
       _metrics (
           gonzo:kernel.all.cpu.user
           gonzo:kernel.all.cpu.sys
           gonzo:kernel.all.cpu.intr
           gonzo:kernel.all.cpu.wait.total
           gonzo:kernel.all.cpu.idle
       )
       _maximum 0.0
       _colourlist cpuColours
_bargraph 40 32 25 20
       _metric gonzo:kernel.all.load["1 minute"]
       _max 1.0
_led 12 43 6 6
       _metric gonzo:disk.all.read _legend diskLegend
_led 25 43 6 6
       _metric gonzo:disk.all.write _legend diskLegend
```

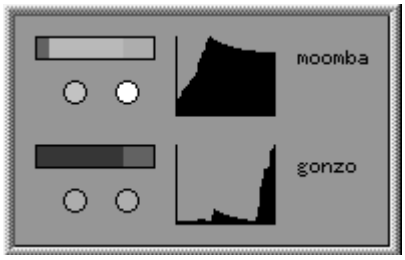

Figure 4-12 Customized `pmgadgets` Display

In addition to the drill-down capabilities of `pmgadgets`, positioning the cursor over a gadget and entering a space character causes an information dialog to be exposed. This dialog tracks the current values of the performance metrics that are associated with the gadget as illustrated by the `pmgadgets` information dialog in Figure 4-13.

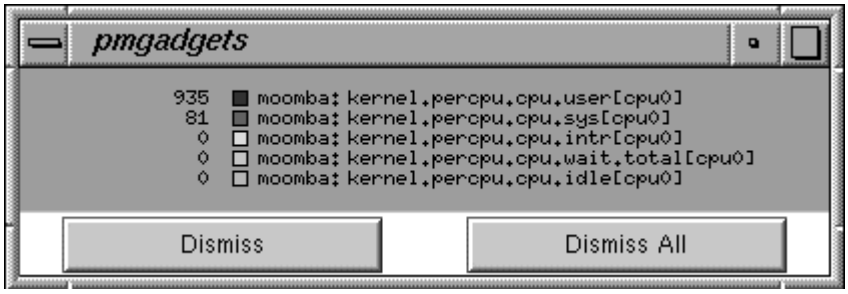

Figure 4-13 `pmgadgets` Dialog

The `pmgadgets`(1) man page provides a complete description of the gadget specification language and the user interface controls of `pmgadgets`.

4.3 The pmdumptext Command

The pmdumptext command displays performance metrics in ASCII tables, suitable for export into databases or report generators. It is a flexible command. For example, the following command provides continuous memory statistics on a host named serv:

```
pmdumptext -imu -h serv -f '%H:%M:%S' mem.util
Metric        kernel  fs_ctl  _dirty  _clean    free    user
      Units             b       b       b       b       b       b
20:14:28        99.14M   6.03M   0.85M  98.42M   0.17G   0.16G
```

See the pmdumptext(1) man page for more information.

System Performance Visualization Tools

Several three-dimensional (3D) graphical tools are provided with Performance Co-Pilot (PCP) to assist you in visualizing performance on monitored systems. These tools are implemented with and require Open Inventor, a 3D graphics facility. Each tool is completely described by its own man page, accessible through the `man` command. For example, the man page for the `pmview` tool can be viewed by giving the following command:

```
man pmview
```

The following major sections are covered in this chapter:

- Section 5.1, page 47, describes the graphical disk activity visualization tool, `dkvis`.
- Section 5.3, page 51, describes the graphical multiprocessor visualization and comparison tool, `mpvis`.
- Section 5.4, page 53, describes the graphical NFS activity visualization tool, `nfsvis`.
- Section 5.5, page 55, describes the Message Passing Interface (MPI) visualization tool, `mpivis`.
- Section 5.6, page 57, describes the tool that displays data extracted from Web-server log files, `weblogvis`.
- Section 5.7, page 61, describes the graphical performance visualization tool, `pmview`, on which the other visualization tools are based.

Other PCP visualization tools, such as `arrayvis`, `clustervis`, `nodevis`, `procvis`, `routervis`, `txmonvis`, `weblogvis`, `webpingvis`, `webvis`, and `xbowvis`, are not described in this chapter. See the `arrayvis`(1), `clustervis`(1), `nodevis`(1), `procvis`(1), `routervis`(1), `txmonvis`(1), `weblogvis`(1), `webpingvis`(1), `webvis`(1), and `xbowvis`(1) man pages for information about these `pmview` based tools.

5.1 Overview of Visualization Tools

For the most interesting and complex problems in performance management, the volume of available information is daunting. One approach to dealing with the

volume and complexity of the information is to employ automated reasoning. Refer to Chapter 6 for a complete description of the `pmie` tool that provides this capability. Another approach is to harness the considerable potential for human visual processing to absorb, analyze, and classify large amounts of information.

PCP has been developed with an assumption that being able to draw three-dimension pictures of system performance is a critical monitoring requirement, and one that offers vast potential for increased insight and understanding for the person charged with some aspect of performance monitoring and management.

Building on SGI technologies of high-performance 3D graphics at the workstation, OpenGL and Open Inventor, PCP delivers a range of utilities, services, and toolkits that are designed both to provide basic visualization tools and to foster the local customization of value-added tools to meet the needs of end-user application and operational environments.

Key components to this performance visualization strategy are as follows:

- Time-series strip charts with `pmchart` (described in Chapter 4) that allow performance metrics from multiple hosts and multiple Performance Metric Domains (PMDs) to be concurrently displayed on a single correlated time axis.

  Predefined chart configurations for common performance scenarios are provided.

- Basic three-dimension models for the following:

  – Per-processor CPU utilization with `mpvis`

  – Per-disk spindle activity with `dkvis`

  – NFS request traffic with `nfsvis`

  – MPI function activity with `mpivis`

- A generalized, three-dimension performance model viewer, `pmview`, that can easily be configured to draw scenes animated by the values of arbitrarily selected performance metrics. Tools like `dkvis`, `mpivis`, `mpvis`, and `nfsvis` are front-ends that create scene descriptions to be displayed and animated with `pmview`.

- Icon-sized stripcharts, meters, and indicator LEDs that may be combined into a desktop control and indicator panel using `pmgadgets`; see Section 4.2.

The `pmgsys` command provides a standard layout for important IRIX performance metrics using `pmgadgets`. The color or height of each gadget is modulated by user-selected performance metrics from one or more PCP sources.

When combined with the VCR and PCP archive services, these visualization tools provide both real-time and retrospective analysis of system performance at many different levels of detail.

5.2 The `dkvis` Disk Visualization Tool

The `dkvis` tool is a graphical disk device utilization viewer, displaying a bar chart showing disk activity. When you give the `dkvis` command, you see a bar chart displaying activity on each disk on the monitored system. You see a **Total Disk I/O Rate for Host *host*** window similar to the one shown in Figure 5-1.

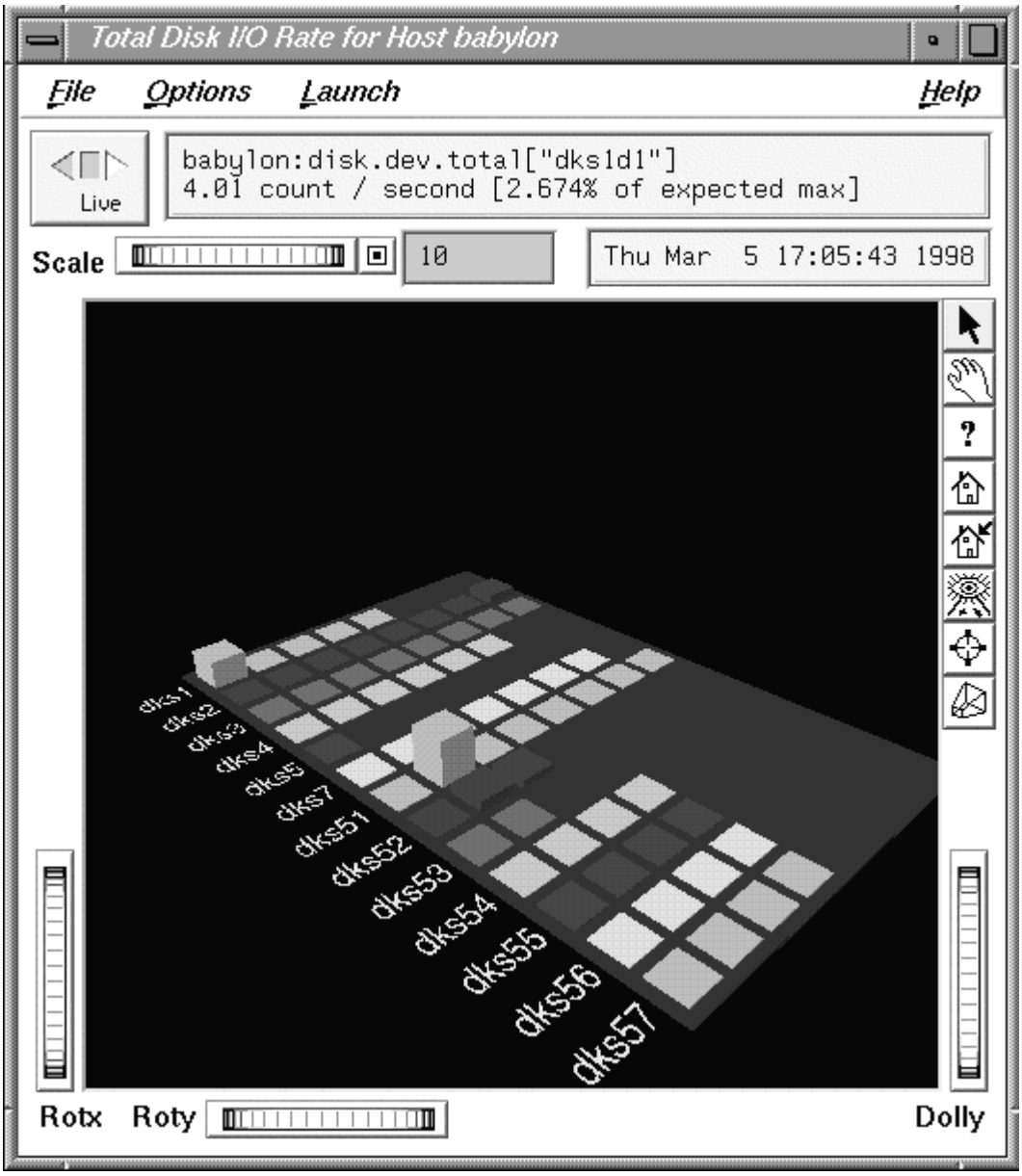

Figure 5-1 `dkvis` **Total Disk I/O Rate** Window

Each row of blocks on the base plane represents the group of disks connected to a single disk controller (or host adaptor or SCSI bus). The label for each row is generated from the characters common to the names of all of the disks on the controller. For example, in Figure 5-1, the disks in the row labeled `dks56` (the same row as the selected block for `dks56d2`) are `dks56d1`, `dks56d2`, `dks56d3`, and `dks56d4`.

The `dkvis` implementation uses the generalized 3D performance viewer `pmview` as described in Section 5.7. Hence, the command line options for `dkvis` include the common ones for `pmview`.

`dkvis` normally displays the total number of I/O operations per second (IOPS). The `-r` option may be used to restrict the display to just the read operations or `-w` may be specified for just the writes.

The `dkvis` command expresses the utilizations in the information window as percentages of some maximum expected rate (clipped to 100%). The `-m` flag allows you to override the default maximum value. This is useful if all of the utilizations are small compared to the maximum. In such a situation, specifying a smaller maximum has the effect of magnifying the differences between the blocks. Similarly, if some of the blocks are almost always at full height, there is a good chance that they are being clipped.

A suitable value for the `-m` option can be determined by clicking the blocks in question, observing the values displayed in the information window for a while, and adding about 10% to the highest value observed. Interactive adjustment of the block height is available via the scale thumb wheel in the `pmview` viewer.

Complete information on the `dkvis` command is available in the `dkvis`(1) man page. The *PCP Tutorial* contains additional examples on the use of `dkvis`.

5.3 The `mpvis` Processor Visualization Tool

The `mpvis` tool is a graphical multiprocessor activity viewer, displaying a bar chart that shows processor activity. When you enter the `mpvis` command, you see a bar chart displaying activity on each processor on the monitored system. You see a **CPU Utilization for Host *host*** window similar to the one shown in Figure 5-2.

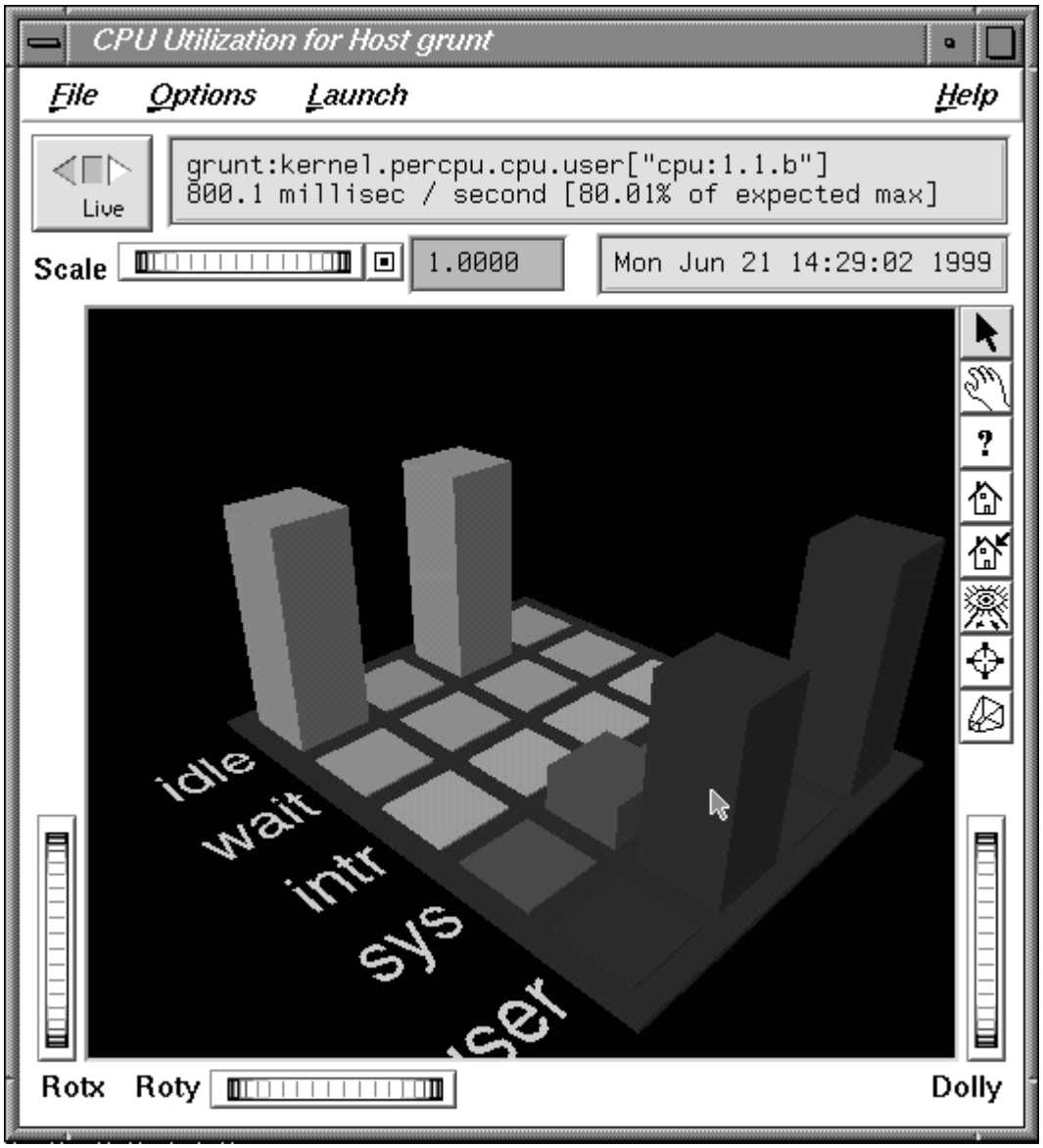

Figure 5-2 `mpvis` **CPU Utilization** Window

This figure shows `mpvis` monitoring a machine with four CPUs. CPU is spending 80% of its time processing user code and about 20% of its time executing system code. Another CPU is executing user code for close to 90% of the time. The remaining two CPUs are idle.

The display contains five labeled rows of blocks, which represent the breakdown of the activity of a single CPU into five states. There is one column of five blocks for each CPU on the system being monitored. These five states are as follows:

idle	No activity
wait	Like idle but waiting for I/O
intr	Processing an interrupt
sys	Executing in the IRIX kernel
user	Executing user code

The `mpvis` implementation uses the generalized 3D performance viewer `pmview` as described in Section 5.7. Hence, the command line options for `mpvis` include the common ones for `pmview`.

Complete information on the `mpvis` command is available in the `mpvis`(1) man page. The *PCP Tutorial* contains additional examples on the use of `mpvis`.

5.4 The `nfsvis` NFS Activity Visualization Tool

The `nfsvis` tool is a graphical NFS (Network File System) activity viewer, displaying a bar chart that shows NFS request activity on the monitored system. NFS is optional software, and may not be present on all systems or at all sites.

When you run the `nfsvis` command, you see a bar chart displaying NFS load on the monitored system. You see a **NFS Client V2 & Server V2 Request Traffic for host *host*** window similar to the one shown in Figure 5-3.

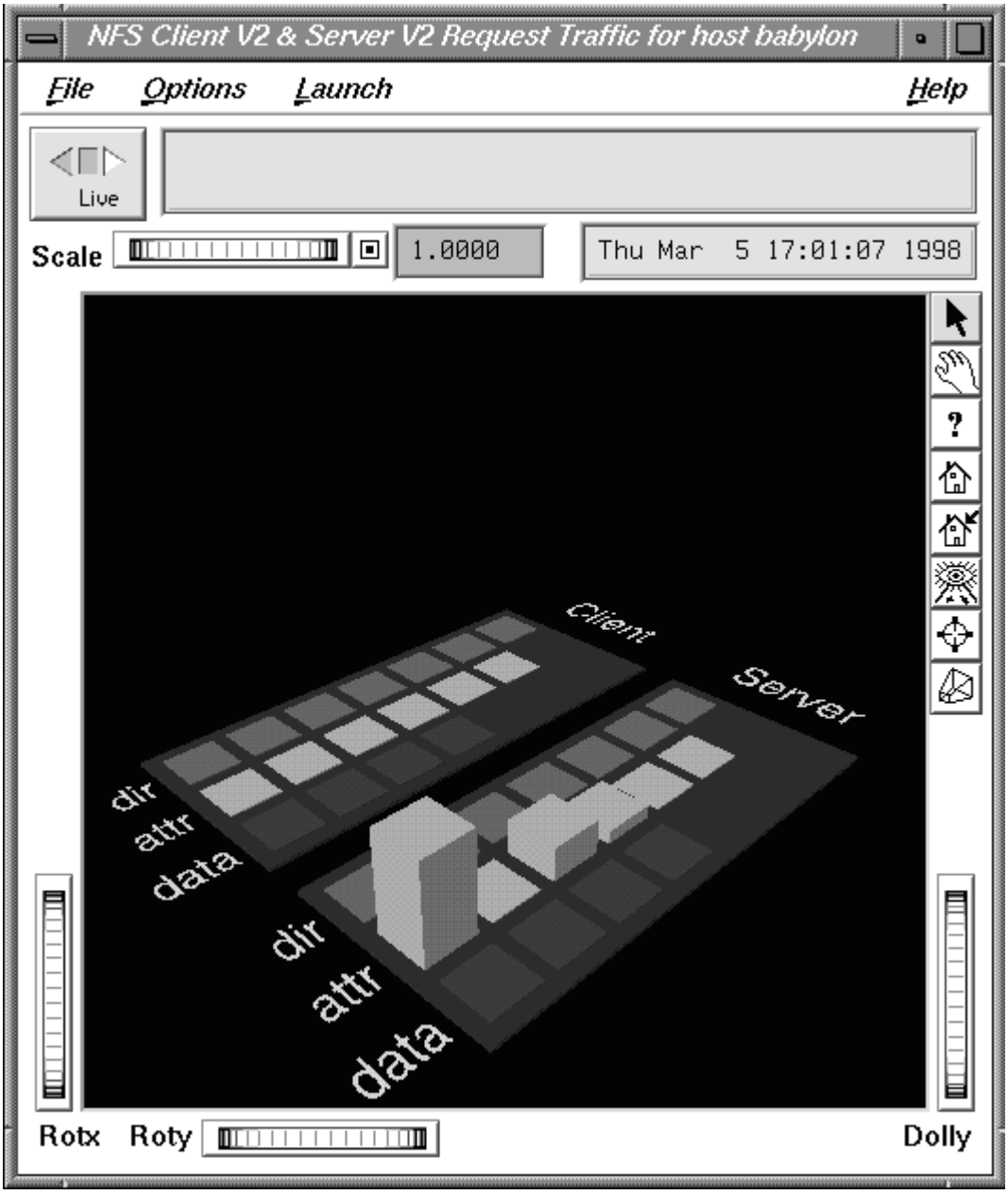

Figure 5-3 `nfsvis` **NFS Client V2 & Server V2 Request Traffic** Window

The statistics are broken into two groups:

Client	Requests made by the monitored machine to NFS servers on other machines
Server	Requests from other machines for the NFS server on the machine being monitored

The statistics in each of these two groups are the same, except that the client group is for outgoing requests and the server group is for incoming requests. Within each group, the requests are further broken down into three categories:

- Requests relating to data within files
- Requests for directory operations (for example, to rename a file)
- Requests involving other attributes of files

The `nfsvis` implementation uses the generalized 3D performance viewer `pmview` as described in Section 5.7. Hence, the command line options for `nfsvis` include the common ones for `pmview`.

Complete information on the `nfsvis` command is available in the `nfsvis`(1) man page. The *PCP Tutorial* contains additional examples on the use of `nfsvis`.

5.5 The `mpivis` MPI Function Activity Visualization Tool

The `mpivis` tool displays a 3D bar chart for the rate of elapsed time used by a particular set of MPI functions for an MPI application. Figure 5-4 shows a sample `mpivis` MPI Activity for ASH *ash* on Host *host* display.

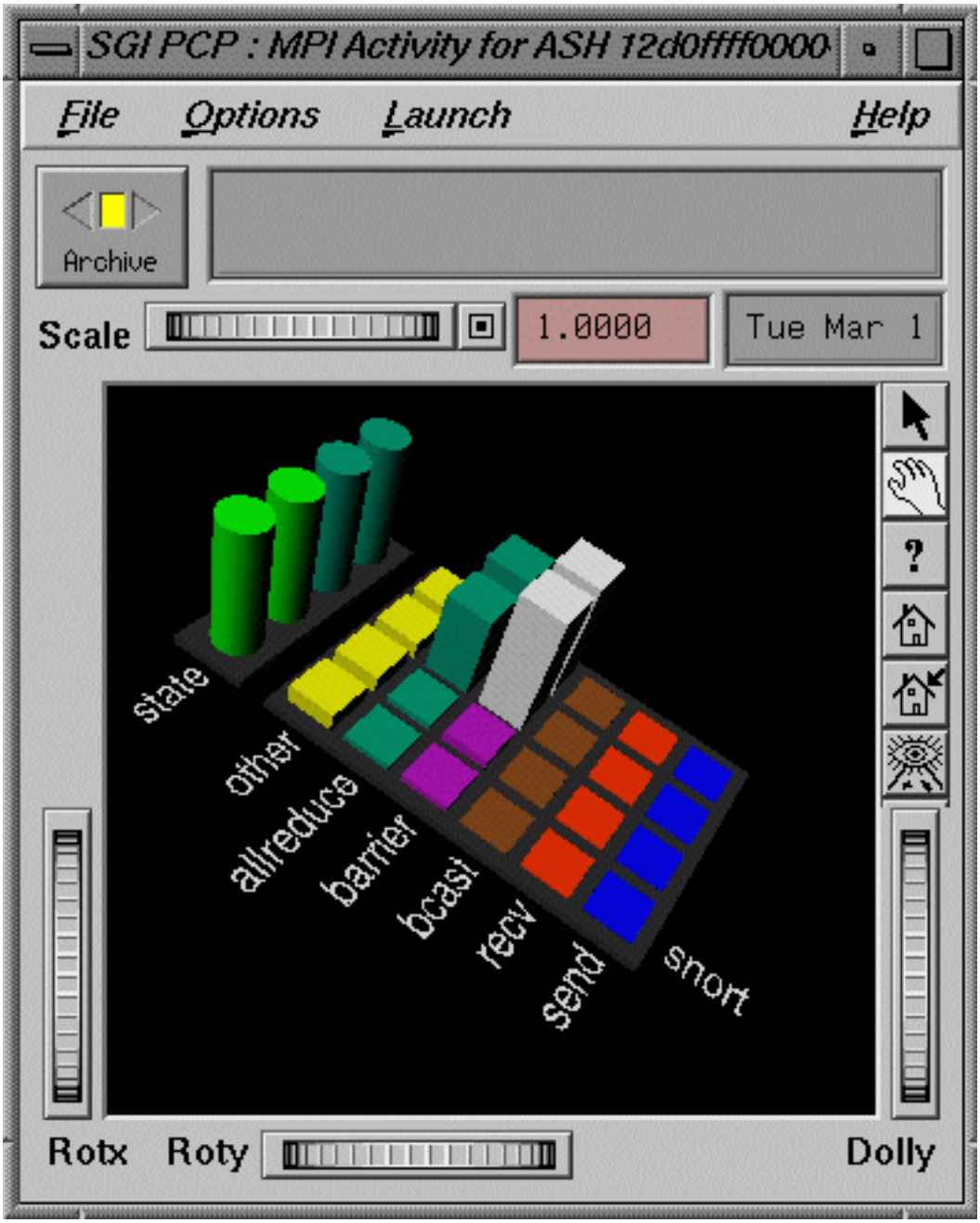

Figure 5-4 mpivis MPI Activity Window

This `mpivis` display was produced with the following control file:

```
# label     MPI function  color
#
send        MPI_Send      blue
recv        MPI_Recv      orange
bcast       MPI_Bcast     burlywood
barrier     MPI_Barrier   violet
allreduce   MPI_Allreduce turquoise
```

From Figure 5-4, you can see that the MPI application has 4 ranks and that the **MPI_Allreduce** and **MPI_Barrier** functions have recently consumed some time, as opposed to **MPI_Bcast**, **MPI_Recv**, and **MPI_Send**. The **other** category, which is always yellow, shows that some other MPI functions are also active. Ranks 0 and 1 have consumed about half as much time in **MPI_Allreduce** and **MPI_Barrier** than ranks 2 and 3. The chart prisms of ranks 2 and 3 of **MPI_Barrier** have turned white. This indicates that they have surpassed the maximum default value of 300 milliseconds per second.

The **state** row of cylinders at the back does not grow; its color changes to indicate the currently executing MPI function at the last update of the metrics. In Figure 5-4, the state is green, which is a reserved color for applications.

5.6 The `weblogvis` Visualization Tool

The `weblogvis` tool displays a 3D bar chart of data extracted from Web-server log files.

For each server, `weblogvis` displays idle time, total activity, and the activity classified by the size of the resultant request. The activity is, by default, the request rate (requests per second) as shown in Figure 5-5, but optionally a data rate (bytes per second) can be requested as shown in Figure 5-6. If the server logs report caching statistics, the bar displays are shown as a stack of values representing cached and non-cached results. Two examples of suitable caching log files are the Netscape common extended log format and the Squid default log file format.

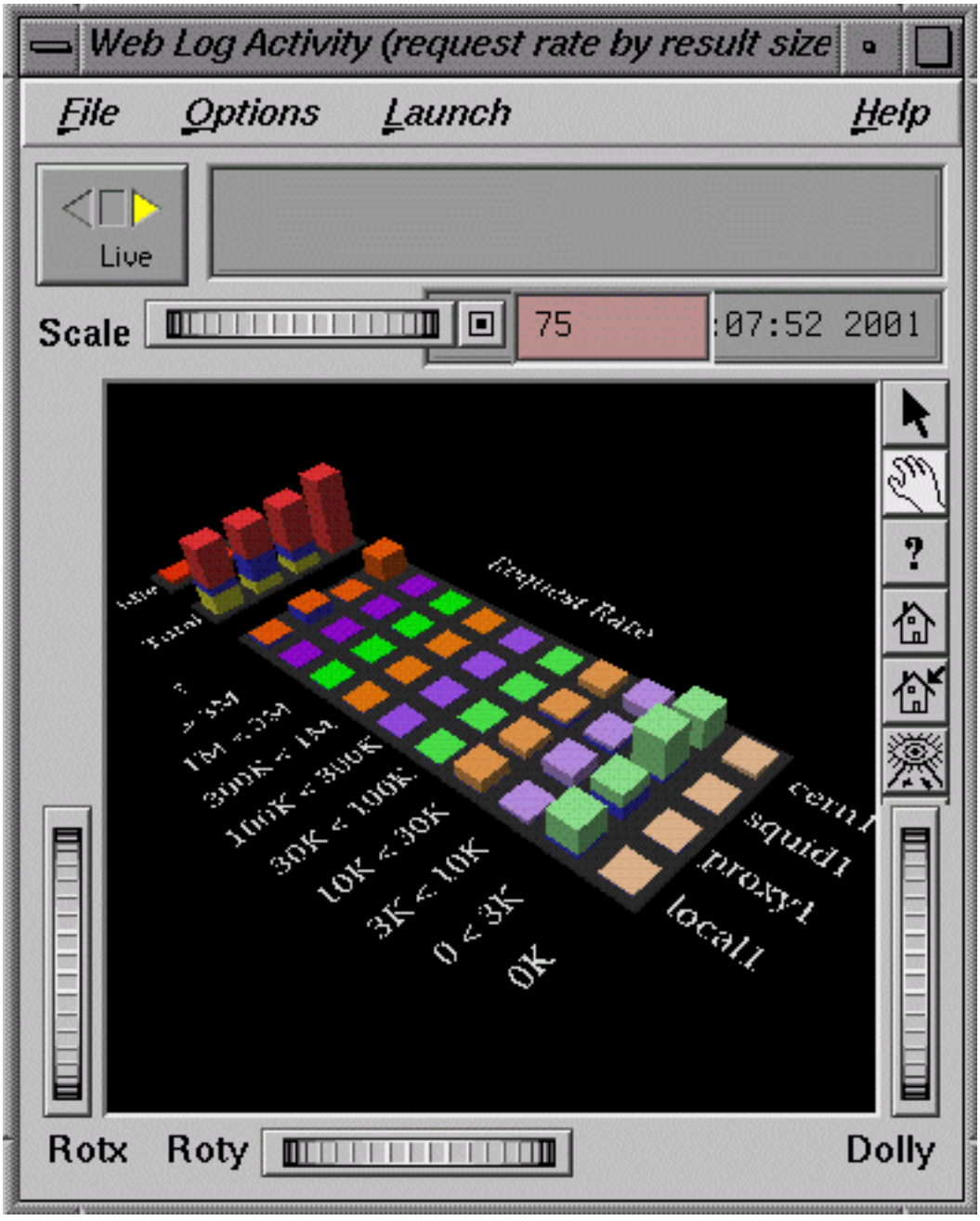

Figure 5-5 `weblogvis` Display of Request Rate Classified by Request Size

Figure 5-5 shows a host with four servers, one without cache statistics, two using the common extended log format, and one Squid server. The scene is comprised of three objects:

Idle time per server	This is a single-value bar representing the number of seconds that the log file has been unaltered.
Total activity rates per server	For servers that report extended statistics (**local1**, **proxy1**, and **squid1**), the total is composed of three stacked values representing the activity rates satisfied by the browser cache (yellow), by the proxy cache (blue) or by a remote server (red). For server logs without the extended data (**cern1**), the single value (red) represents the total activity rate.
Activity rates per server classified by request size	For servers that report extended statistics, each bar is composed of two stacked values, representing the activity rates for that request size, that are satisfied by the proxy cache (blue) or by a remote server (color coded based on the request size). For server logs without the extended data, the single color–coded value represents the activity rate for that request size.

Figure 5-6 shows the same host with an alternate type of display. It shows the activity rates classified by the request type instead of the request size.

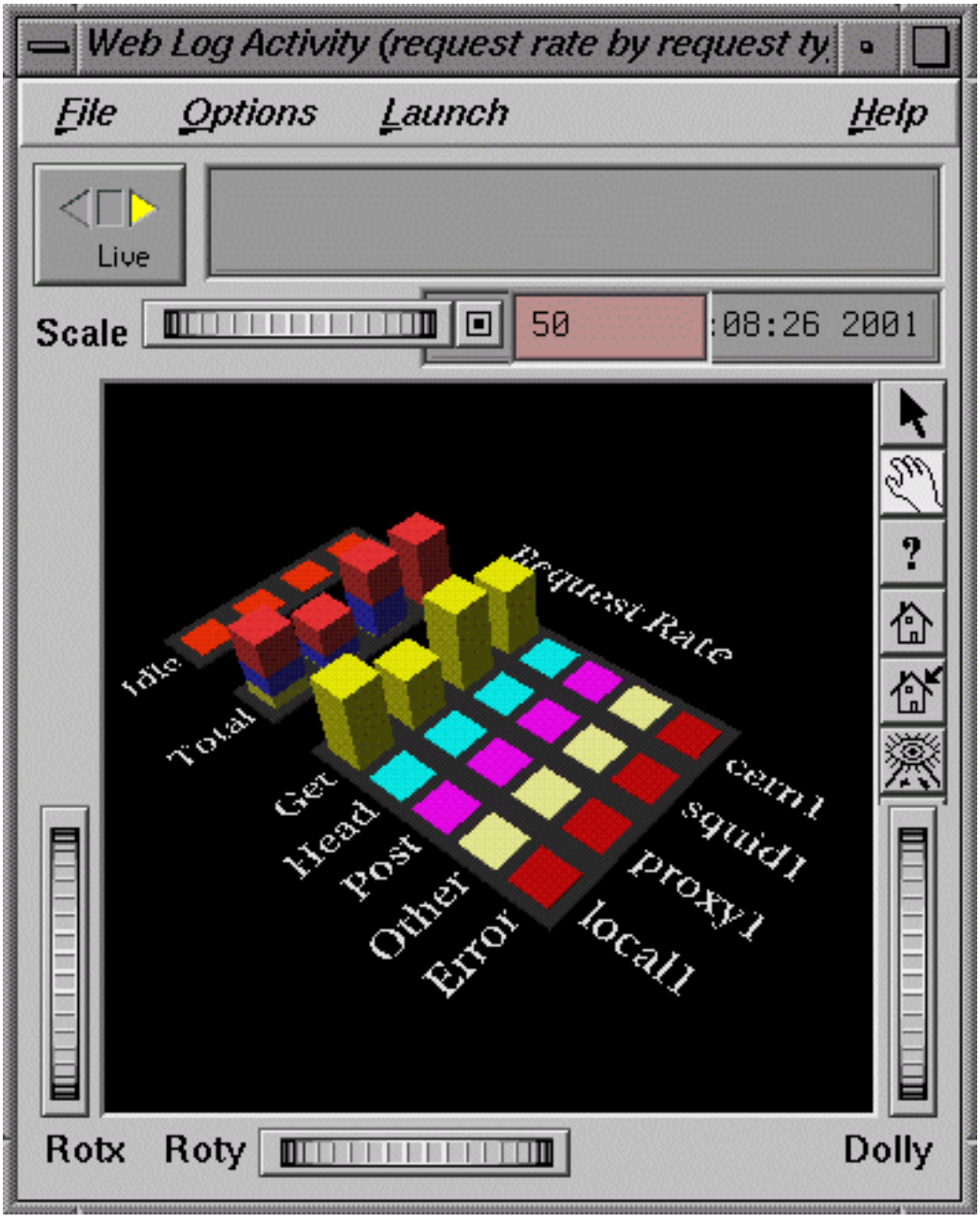

Figure 5-6 `weblogvis` Display Request Rate Classified by Request Type

5.7 The pmview Tool

The pmview tool is a generalized 3D Open Inventor application that supports dynamic displays of clusters of related performance metrics as utilization blocks (or towers) on a common base plane. The pmview tool is the basis for dkvis, mpvis, osvis, nfsvis, mpivis, and weblogvis all discussed in this chapter, as well as arrayvis, clustervis, nodevis, procvis, routervis, txmonvis, webpingvis, webvis, and xbowvis. The pmview tool may also be used to construct customized 3D performance displays.

A closely related tool to pmview is pmview+. It has the same capabilities as pmview, as well as the support for showing interconnects between the bases of pmview grids. It was written to support oview in its display of SGI 3000 servers.

Open Inventor is an object-oriented toolkit that simplifies and abstracts the task of writing graphics applications into a set of easy-to-use objects. Its run-time support is distributed with IRIX system software in the inventor_eoe.sw product image.

The pmview command displays performance metrics as colored blocks arranged in a grid on a grey base plane. The height of each block changes as the value of its corresponding metric (or metric instance) changes. Labels may be added to the scene to help identify groups of metrics, as shown in Figure 5-1, Figure 5-2, and Figure 5-3.

A configuration file is used to specify the position, color, and scale of metrics and metric instances in the scene. Metric values that exceed the associated scaling factor are displayed at the maximum height and change color to white. If a metric is unavailable, the bar height is minimized, and the bar color changes to grey.

Normally, pmview operates in live mode where performance metrics are fetched in real time. The user can view metrics from other accessible Internet hosts that are running the PCP collector daemon, pmcd. The pmview tool can also replay archives of performance metrics collected by pmlogger.

All metrics in the PMNS with numeric value semantics from multiple hosts or archives may be visualized. The pmview tool examines the semantics of the metrics and, where sensible, converts the fetched metric values to a rate before scaling.

The pmview tool window contains a menu bar, time and scale controls, metric and time values, and an *examiner* window; see the ivview command, which displays the 3D scene.

The left, right, and bottom edges of the examiner viewer window contain a variety of thumb wheels and buttons that allow the user to adjust the visualization of the 3D scene. The **Rotx** and **Roty** thumb wheels allow the user to rotate the scene about the

X and Y axes, respectively. The **Dolly** thumb wheel moves the virtual camera closer to or further from the scene, allowing the user to examine specific parts in detail or view the entire scene.

On the right edge of the viewer are eight buttons that affect the way the user can interact with the scene:

- The **pointer** button changes the cursor to a pointer that allows blocks in the scene to be selected. The `Esc` key can also be used to toggle between the pointer and hand cursors.
- The **hand** button changes the cursor to a hand that can be used to rotate, translate, and examine the scene via **Dolly**, using a combination of mouse buttons and movement.

  The left mouse button can be used to rotate the scene in the direction of the mouse. Releasing the mouse button before the mouse has stopped moving causes the scene to continue rotating until a mouse button is pressed again.

  The middle mouse button can be used to pan the scene. By pressing both left and middle buttons, the mouse can be used as a virtual camera.
- The **question mark** button displays **SGI Help** for the examiner viewer. To install online help, use `inst` to install the `inventor_eoe.sw.help` package from your IRIX system software distribution. See the Performance Co-Pilot release notes for more information on prerequisite subsystems.
- The **home** button changes the scene back to its original position, or the position set by the `home pointer` button.
- The **home pointer** button sets the new home position of the scene to be the scene currently in view.
- The **eye** button resizes the scene so that it completely fits into the 3D viewing area.
- The **cross-hairs** button moves the scene so that the object under the cursor is in the center of the viewing area. Change the hand cursor and press the **cross-hairs** button. The cursor changes to a target. Select the block to be centered and the scene rotates and translates appropriately.
- The **perspective box** button switches between perspective and orthogonal projections.

Pressing the right mouse button within the scene displays a menu of options that affect how the 3D scene is drawn. The options include drawing the blocks as wireframes and turning on stereo viewing.

When the pointer cursor is active, more information about the 3D scene can be obtained. Text describing the metric represented by the block beneath the cursor displays in the top text box of the `pmview` window. This text displays the source, name, and instance of the performance metric, and the value, units, and percentage of the expected minimum the value represents.

Clicking the left mouse button on a block highlights the block with a red wireframe, as shown in Figure 5-7. The metric description text box is now fixed on that metric and the values continue to be updated as new metrics are fetched. This allows other actions to be performed with the mouse while examining a single metric in detail at the same time. Click the left mouse button on the space surrounding the scene to remove the selection.

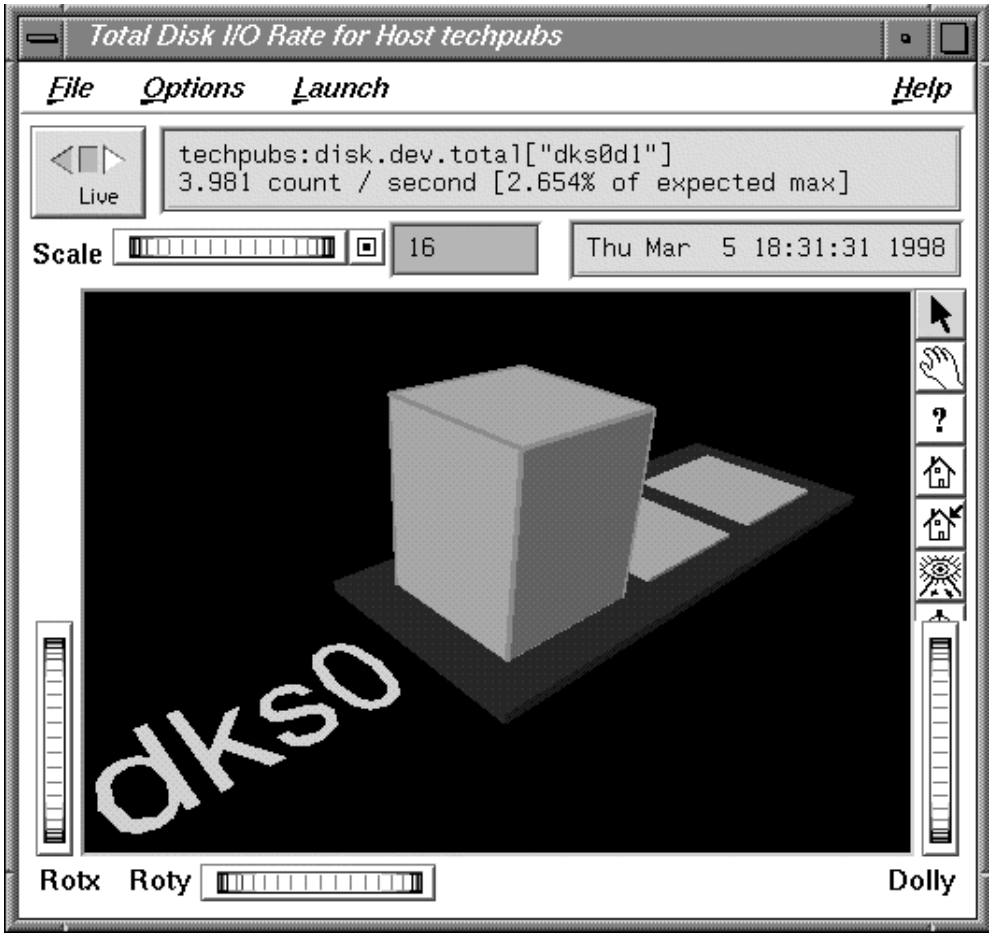

Figure 5-7 `pmview` Window with a Block Selected

Multiple blocks may also be selected by either Shift clicking with the left mouse button or by clicking on a base plane. Shift clicking toggles the selection status of a particular block and leaves the selection status of other blocks unaltered. Clicking on the base plane selects all blocks belonging to the base plane. Whenever multiple blocks are selected, no accompanying text is displayed in the text box. However, multiple block selection affects the launching of other tools because all metrics on the base plane are considered to be selected as a group.

5.7.1 pmview Menus

There are four menus in pmview tools:

File	Records, saves, and prints scenes
Options	Accesses the time controls
Launch	Starts other tools
Help	Obtains online help

The **Launch** menu consists of a list of tools that operate on the current selection of metrics. How the tool is invoked depends on the type of tool. Tools that operate on any metric (such as pmchart, pmval, and pmdumptext) use the metrics selected directly as input. Thus pmchart displays all the selected metrics in a chart, pmval is invoked within winterm for each metric, and pmdumptext displays multiple metrics in one winterm. Other tools use what metrics are pertinent. If no metric is pertinent or selected, only the source of the metrics is used, that is, the monitored host or archive. For each **Launch** menu item there is an associated launch script. The launch scripts generally know the relationship between routers, nodes, and CPUs. Thus if CPUs are selected in mpvis and if nodevis is launched, only the nodes that have the selected CPUs attached are displayed.

Some launchable tools are listed below:

dkvis, mpvis, nfsvis, and osvis	pmview-based tools for visualizing disk activity, CPUs, NFS, and the OS (operating system).
pcp	Brings up a window that summarizes the PCP installation.
pmdumptext	Brings up a window that shows the performance metrics as text.
pmchart	A tool for graphically displaying and correlating time-series trends of performance metrics. See Section 4.1, page 21, for details.
pmgsys	A miniature IRIX performance metrics viewer, available only in live mode, not in archive mode.
pmkstat	A text-based tool that displays, at intervals, a high-level summary of system performance.

`pmval` — A tool that displays the values of performance metrics textually. Only one metric (with one or more instances) may be selected to successfully launch this tool.

In addition to the menu options for time controls, the current direction and mode of the time controls is shown in a button in the top-left corner of the `pmview` window. Pressing this button displays the time control dialog.

Below this button is a thumb wheel and an editable text box to specify a scale multiplier that is applied to all values in the scene. Spinning the thumb wheel to the right, or incrementing the value in the text field, increases the scaling and raises the height of the bars. Conversely, spinning the thumb wheel to the left or decrementing the text field decreases the scaling and lowers the height of the bars.

The button beside the thumb wheel resets the scale to one. This is especially useful when the scale specified in the configuration file reduces the usefulness of the visualization as a consequence of the bars being either too low or beyond the maximum scale height.

5.7.2 Creating Custom Visualization Tools with `pmview`

At startup time, a configuration file is read that specifies the following:

- Geometry for the scene to be displayed by `pmview`
- Associations between the visual appearance of "blocks" and performance metrics

The scene is based on a grid that can contain a variety of objects and can resize itself to accommodate objects of varying sizes. To distinguish this configuration file format from an earlier (still supported) format, configuration files must begin with the following line:

```
pmview Version 2.1
```

All lines beginning with a # character are treated as comments and ignored. Spaces, tabs, and newlines are treated as white space to allow multiple statements on the same line. The simplest configuration file consists of a single object that may represent one or more metrics and metric instances.

The configuration file consists of two sections: global parameters and color lists, and the object definitions. The global parameters control the size of the objects in the scene. For example, a scaling factor of 1.2 can be applied to all objects with the following line:

```
_scale 1.2
```

Groups of colors may be associated with a name and referenced later in the file. Colors may be `X(1)` color names, `X(1)` numerical colors, or three real values representing the saturation of red, green, and blue, respectively. The following color list contains three identical colors:

```
_colorlist cpu ( red rgbi:1.0/0.0/0.0 1.0 0.0 0.0 )
```

The `mpvis` configuration file (which can be generated with the `-V` option) looks like Example 5-1:

Example 5-1 `mpvis` Configuration File

```
pmview Version 2.1 "mpvis" "-V" "-C"
#
# mpvis
#
# ncpus = 1
# nrows = 1
# ncols = 1
#
# List:
# cpu0
_gridSpace 120

_colorlist cpu ( green2 cyan2 yellow2 red2 blue2 )
_grid 0 0 _hide ( # outer grid
        _baseLabel "SGI PCP : CPU Utilization for Host ha1.melbourne.sgi.com\ncpu0 only"
        _bar _groupByInst (
            _metrics (
                kernel.percpu.cpu.idle[cpu0]        1000 "idle"
                kernel.percpu.cpu.wait.total[cpu0]  1000 "wait"
                kernel.percpu.cpu.intr[cpu0]        1000 "intr"
                kernel.percpu.cpu.sys[cpu0]         1000 "sys"
                kernel.percpu.cpu.user[cpu0]        1000 "user"
            )
            _colorlist cpu
            _baseLabel "SGI PCP : CPU Utilization for Host ha1.melbourne.sgi.com\ncpu0 only"
        )
)
```

Multiple objects can be visualized using a _grid object, which may contain multiple objects (including more _grid objects). The _grid object resizes columns and rows to accommodate the largest contained object. Objects can occupy multiple grid squares and can be aligned with a particular edge or corner of a grid square. The _bar object has a single bar for each metric instance and labels for each metric. The scale height for each metric instance is 1000 in the units of the metric (milliseconds utilization per second).

The specification file shown in Example 5-2 produces a scene like the one shown in Figure 5-8. The file has a grid, labels, bars, and a stack utilization object.

Example 5-2 Specification File for pmview

```
pmview Version 2.1
_colorlist cpu_colors ( blue2 red2 yellow2 cyan2 green2 )
_colorlist disk_colors ( purple2 yellow2 )
_colorlist memory_colors ( rgbi:1.0/1.0/0.0 rgbi:0.0/1.0/1.0 rgbi:1.0/0.0/0.0
                           rgbi:1.0/0.0/1.0 rgbi:0.0/0.0/1.0 rgbi:0.0/1.0/0.0 )
_grid hide (
_label 3 1 _west _down _large ‘‘CPU’’
       _stack 4 1 _west _utilmod (
             _metrics (
               kernel.all.cpu.user        1000
               kernel.all.cpu.sys         1000
               kernel.all.cpu.intr        1000
               kernel.all.cpu.wait.total  1000
               kernel.all.cpu.idle        1000
           )
           _colorlist cpu_colors
           _baseLabel ‘‘CPU Utilization’’
       )
_label 3 3 _west _down _large ‘‘Load’’
       _bar 4 3 2 1 _west (
           _metrics (
               kernel.all.load[15]    2
               kernel.all.load[5]     2
               kernel.all.load[1]     2
           )
           _metriclabels _away ( ‘‘15’’ ‘‘5’’ ‘‘1’’ )
           _colorlist ( blue2 blue2 blue2 )
_baseLabel ‘‘Average System Load over last 1, 5 and 15 minutes\nNormalized to 2’’
       )
```

```
_label 0 1 _west _down _large ‘‘Mem’’
         _stack 1 1 _west _utilmod (
           _metrics (
               mem.util.kernel        1
               mem.util.fs_ctl        1
               mem.util.fs_dirty      1
               mem.util.fs_clean      1
               mem.util.user          1
           )
           _colorlist memory_colors
           _baseLabel ‘‘Physical Memory Utilization’’
       )
       _label 0 3 _down _large ‘‘Disk’’
       _stack 1 3 _west _cylinder (
           _metrics (
               disk.all.read          100
               disk.all.write         100
           )
           _colorlist disk_colors
           _baseLabel ‘‘Disk Operations\nNormalized to 100 I/Os per second’’
        )
)
```

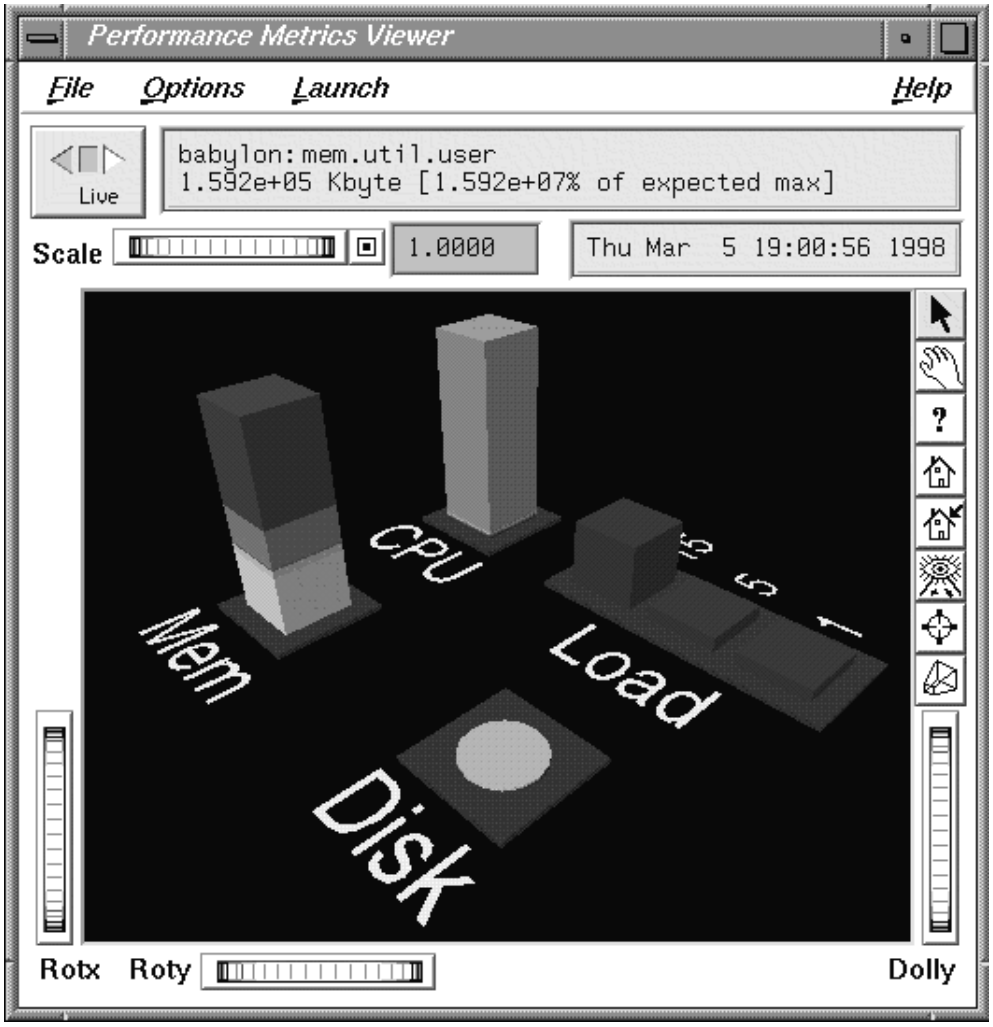

Figure 5-8 Custom `pmview` Scene

To assist in the creation of front-end tools, a file containing shell procedures for generating usage information and parsing `pmview` command line options is located at `/usr/pcp/lib/pmview-args`. The `dkvis`, `mpvis`, `nfsvis`, and `osvis` tools are all shell scripts that use these shell procedures to generate a configuration file for `pmview`.

Performance Metrics Inference Engine

The Performance Metrics Inference Engine (`pmie`) is a tool that provides automated monitoring of, and reasoning about, system performance within the Performance Co-Pilot (PCP) framework.

The following major sections in this chapter are as follows:

- Section 6.1, page 71, provides a brief description of how to use the `pmrules` GUI for creating `pmie` rules from parameterized templates.

This chapter only provides descriptions about tools and services which are not provided in the foundation components of PCP. For an introduction to `pmie` and its features and details of other pmie tools, see Chapter 5, "Performance Metrics Inference Engine" in the *Performance Co-Pilot for IRIX User's and Administrator's Guide*.

6.1 Creating `pmie` Rules with `pmrules`

The GUI tool `pmrules` may be used to generate `pmie` rules from templates that are shipped with PCP as shown in Procedure 6-1. These templates are parameterized versions of rules describing common performance scenarios suited for `pmie` monitoring.

Procedure 6-1 Creating `pmie` Rules

1. Start `pmrules`, and choose Import... from the Template menu.
2. Click the **Choose File...** button in the "Import template(s) from file" dialog.

   Sample templates are installed in the directory `/var/pcp/config/pmrules`.
3. Double-click the `pcp` directory in the `pmrules` directory browser window.

   An **Import template(s) from file** dialog appears, as shown in Figure 6-1.

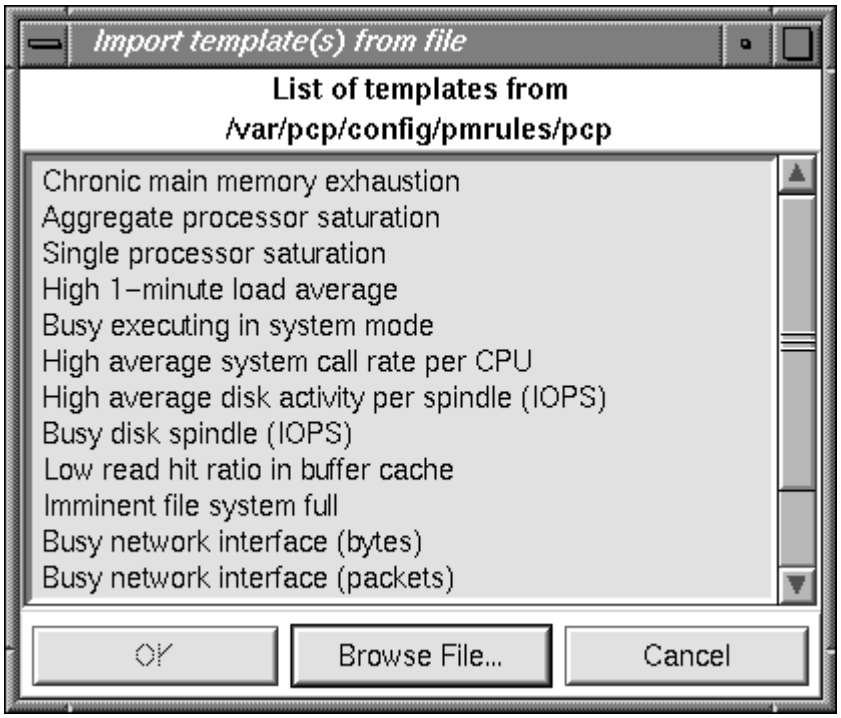

Figure 6-1 `pmrules` **Import template(s) from file** Dialog

4. Select the desired templates, click OK, and return to the `pmrules` main window, which appears similar to the one shown in Figure 6-2.

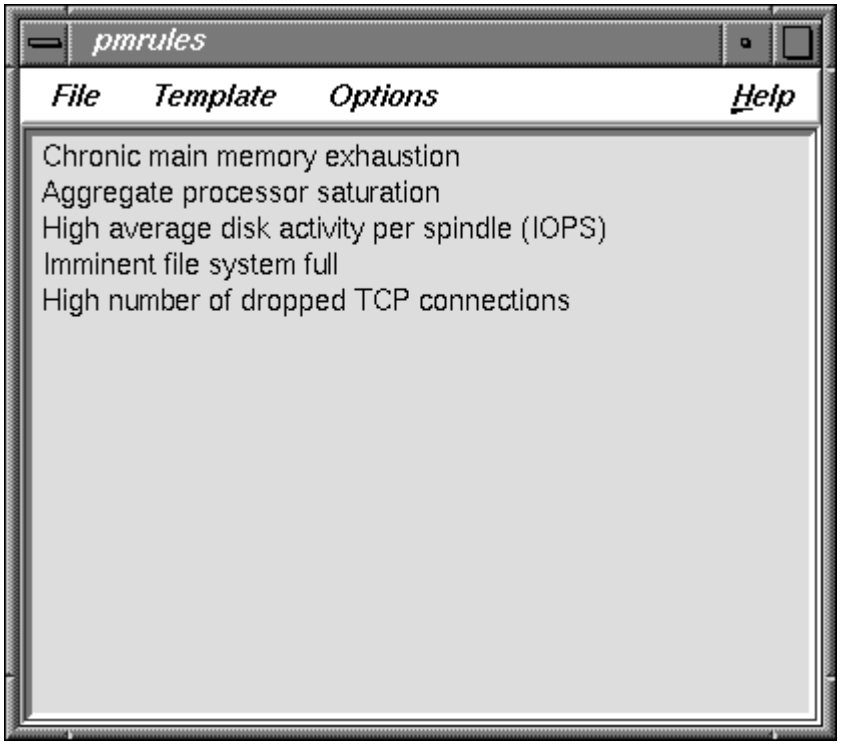

Figure 6-2 `pmrules` Main Dialog after Template Selection

5. Double-click the desired template, and the `pmrules` **Edit template** dialog displays, similar to the one shown in Figure 6-3.

   At this point, you can customize the template by assigning values to the **threshold**, **delta**, and **holdoff Parameters** text boxes, then either selecting one of the predefined **Actions**, or specifying your own custom user action.

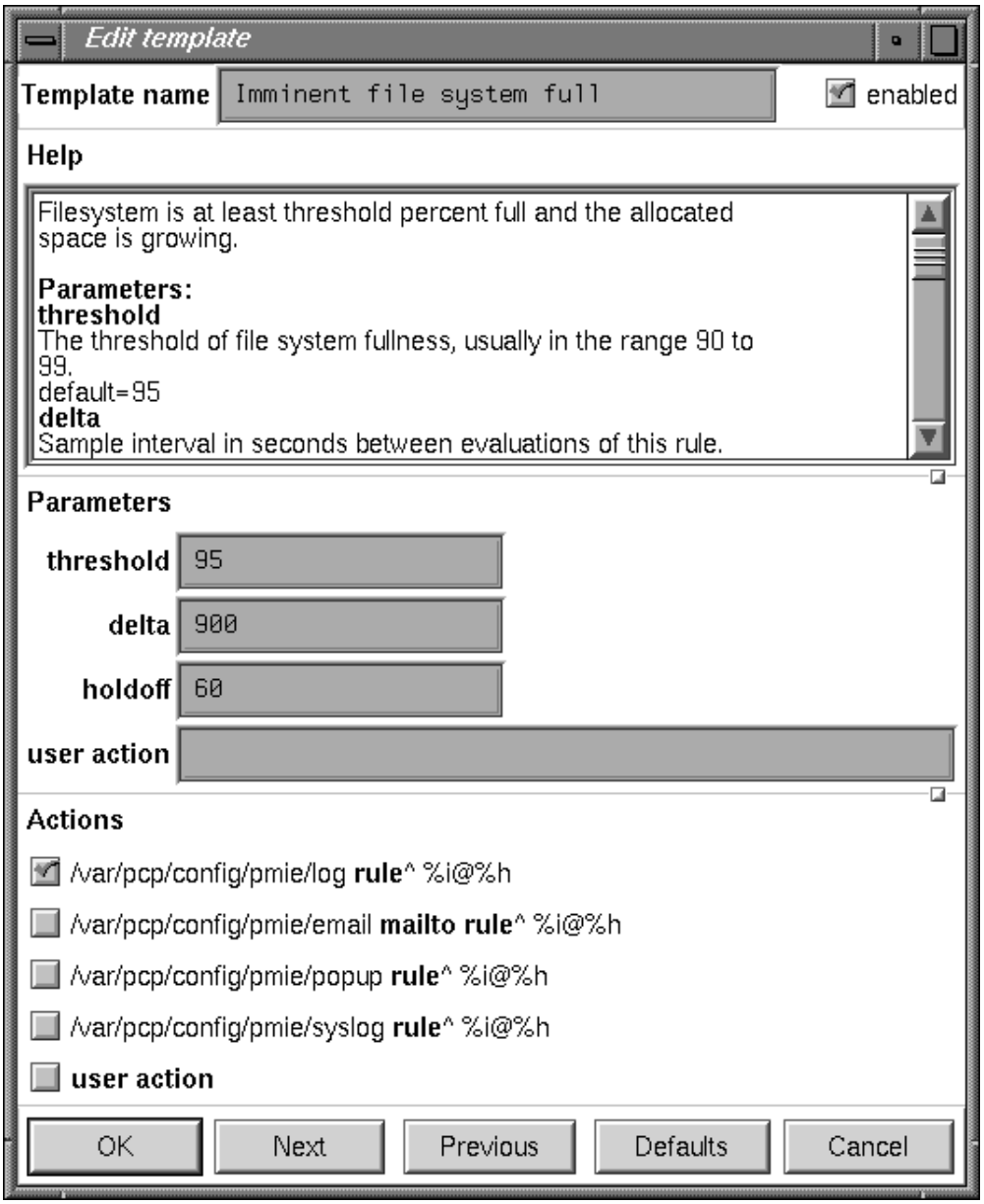

Figure 6-3 `pmrules` **Edit template** Dialog

6. When you are finished customizing the template, click OK and return to the main `pmrules` window.
7. Choose **Save As** from the **File** menu, and provide a new name for your private copy of the `pmrules` template file.

   Two files are saved. The first one takes the given filename and is your private copy of the `pmrules` template file. The second file takes the given filename with the suffix `.pmie` appended and contains the `pmie` rules—this second file should be given as an argument to `pmie`.

You can also create new templates for other performance problems. These can then be included in the template collection available to `pmrules`, and then used to customize instances of the `pmie` rules for particular hosts.

See the `pmrules`(1) man page for a complete description of the capabilities of the `pmrules` tool.

Archive Logging

Performance monitoring and management in complex systems demands the ability to accurately capture performance characteristics for subsequent review, analysis, and comparison. Performance Co-Pilot (PCP) provides extensive support for the creation and management of archive logs that capture a user-specified profile of performance information to support retrospective performance analysis.

The following major sections are included in this chapter:

- Section 7.1, page 77, presents the concepts and issues involved with creating and using archive logs.
- Section 7.2, page 78, describes the interaction of the PCP tools with archive logs.

This chapter only provides descriptions about tools and services which are not provided in the foundation components of PCP. For an introduction to `pmlogger` and its features and details about other `pmlogger` tools, see Chapter 6, "Archive Logging", in the *Performance Co–Pilot for IRIX User's and Administrator's Guide*.

7.1 Introduction to Archive Logging

Within the PCP, the `pmlogger` utility may be configured to collect archives of performance metrics. PCP adds the following features to those already provided by the standard PCP components in IRIX:

- Record mode in various GUI monitoring tools to create archives as needed from the current visualization.
- `cron`-based scripts to expedite the operational management, for example, log rotation, consolidation, and culling.

7.2 Using Archive Logs with Performance Visualization Tools

7.2.1 Administering PCP Archive Logs Using `cron` Scripts

The IRIX operating system supports the standard `cron` process scheduling system. Complete information on the `cron` command is available in the appropriate man page and in *IRIX Admin: System Configuration and Operation*.

PCP supplies shell scripts to use the `cron` functionality to help manage your archive logs. The following scripts are supplied:

Script	Description
`pmlogger_daily`	Performs a daily housecleaning of archive logs and notices.
`pmlogger_merge`	Merges archive logs and is called by `pmlogger_daily`.
`pmlogger_check`	Checks to see that all desired `pmlogger` processes are running on your system, and invokes any that are missing for any reason.
`pmsnap`	Generates graphic image snapshots of `pmchart` performance charts at regular intervals.

The configuration files used by these scripts can be edited to suit your particular needs, and are generally controlled by the `/var/pcp/config/pmlogger/control` file (`pmsnap` has an additional control file). Complete information on these scripts is available in the `pmlogger_daily`(1) and `pmsnap`(1) man pages.

7.2.2 Snapshots from PCP Archive Logs

Periodic snapshot images of recent performance, activity levels, and resource utilization can be extracted from the PCP archive logs and published via a World Wide Web (WWW) server. These are high-quality images generated from `pmchart` that provide an excellent vehicle for publishing performance summary information for users, system and network administrators, or managers. The `pmsnap` services may be used to automate snapshots. For additional information, see the `pmsnap`(1) man page.

7.2.3 Making Snapshot Images from Archive Logs

You may also choose to enable periodic snapshot images of performance data to be produced from the archive logs using the facilities of `pmsnap`; instructions for this operation can be found in Section 4.1.9, page 40, and in the `pmsnap`(1) man page.

Assume the local host has been set up to create archive logs of performance metrics collected from the host `oscar` (which may be either the local host or a remote host). Execute all of the following tasks while logged into the local host as the superuser (`root`).

1. Make sure the optional subsystem `pcp.sw.monitor` has been installed.
2. Use the `/var/pcp/config/pmsnap/Summary` snapshot as an example (you may wish to customize this later).
3. Ensure that the `pmlogger` that is collecting performance metrics from the host `oscar` includes all of the metrics named in the `/var/pcp/config/pmlogger/config.Summary` configuration file (you may wish to simply use this as the configuration file for this `pmlogger` instance). If necessary, reconfigure this `pmlogger` instance as follows:

   `kill -INT` *PID-of-pmlogger-instance*

   Edit the configuration file as required. Restart `pmlogger` with this command:

   `/usr/pcp/bin/pmlogger_check`

4. Check the two Summary lines in the `/var/pcp/config/pmsnap/control` file. You must replace `LOCALHOSTNAME` with `oscar` in both lines (unless `oscar` is the local host, in which case the change is optional), and you may wish to change the directory for the output files (the default is `/var/www/htdocs/snapshots`).
5. Augment the `crontab` file for `root` to allow `pmsnap` to be run periodically. For example:

   `crontab -l >/tmp/foo`

6. Edit `/tmp/foo`, adding lines similar to those from `/var/pcp/config/pmlogger/crontab` for `pmsnap`; for example:

   ```
   # every 30 minutes, generate performance snapshot images
   30,0   *     *     *     *       /usr/pcp/bin/pmsnap -d :0
   ```

   The snapshots are produced using `pmchart`, and this tool requires connection to an X server. If the local host is not running an X server, then you must locate a

system with an active X server, and ensure that this X server will accept connections from remote X clients; see the `xhost`(1) man page for details. If this host is `grover`, then replace `-d :0` in the line above with `-d grover:0`

Other options for gaining access to an active X server are discussed in the `pmsnap`(1) man page.

7. Make these changes permanent with this command:

   ```
   crontab </tmp/foo
   ```

8. After 30 minutes or so (time enough for the `cron` command to complete), check that the GIF files have been created:

   ```
   ls -l /var/www/htdocs/snapshots
   ```

9. Create a Web page that includes the images. A sample file of HTML source is provided in `/var/pcp/config/pmsnap/Summary.html`.

Customizing and Extending PCP Services

Performance Co-Pilot (PCP) has been developed to be fully extensible. The following sections summarize the various facilities provided to allow you to extend and customize PCP for your site:

- Section 8.1, page 81, describes the procedure for customizing the summary PMDA to export derived metrics formed by aggregation of base PCP metrics from one or more collector hosts.
- Section 8.2, page 85, describes the various options available for customizing and extending the basic PCP tools.
- Section 8.3, page 87, details where to find further information to assist in the development of new PMDAs to extend the range of performance metrics available through the PCP infrastructure.
- Section 8.4, page 87, outlines how new tools may be developed to process performance data from the PCP infrastructure.

This chapter describes ways of extending PCP services using tools and services made available by the `pcp` product. For information on how to extend and customize PCP, see Chapter 8, "Customizing and Extending PCP Services", in *Performance Co-Pilot User's and Administrator's Guide*.

8.1 PMDA Customization

The summary PMDA is a special case that warrants further discussion.

8.1.1 Customizing the Summary PMDA

The summary PMDA exports performance metrics derived from performance metrics made available by other PMDAs. It is described completely in the `pmdasummary`(1) man page.

The summary PMDA consists of two processes:

`pmie` process — Periodically samples the base metrics and compute values for the derived metrics. This dedicated instance

	of the PCP `pmie` inference engine is launched with special command line arguments by the main process.
`main` process	Reads and buffers the values computed by the `pmie` process and makes them available to the Performance Metrics Collection Daemon (PMCD).

All of the metrics exported by the summary PMDA have a singular instance and the values are instantaneous; the exported value is the correct value as of the last time the corresponding expression was evaluated by the `pmie` process.

The summary PMDA resides in the `/usr/pcp/pmdas/summary` directory and may be installed with a default configuration by following the steps described in Chapter 2, page 11.

Alternatively, you may customize the summary PMDA to export your own derived performance metrics by following the steps in Procedure 8-1:

Procedure 8-1 Customizing the Summary PMDA

1. Check that the symbolic constant `SYSSUMMARY` is defined in the `/var/pcp/pmns/stdpmid` file. If it is not, perform the postinstall update of this file, as superuser:

   ```
   # cd /var/pcp/pmns
           # ./Make.stdpmid
   ```

2. Choose Performance Metric Name Space (PMNS) names for the new metrics. These must begin with `summary` and follow the rules described in the `pmns`(4) man page. For example, you might use `summary.fs.cache_write` and `summary.fs.cache_hit`.

3. Edit the `pmns` file in the `/usr/pcp/pmdas/summary` directory to add the new metric names in the format described in the `pmns`(4) man page. You must choose a unique performance metric identifier (PMID) for each metric. In the `pmns` file, these appear as `SYSSUMMARY:0:x`. The value of *x* is arbitrary in the range 0 to 1023 and unique in this file.

   For example:

   ```
   summary {
             cpu
             disk
             netif
             fs              /*new*/
   ```

```
}
summary.fs {
          cache_write     SYSSUMMARY:0:10
          cache_hit       SYSSUMMARY:0:11
}
```

4. Use the local test PMNS `root` and validate that the PMNS changes are correct.

   For example, enter this command:

   `pminfo -n root -m summary.fs`

   You see output similar to the following:

   ```
   summary.fs.cache_write PMID: 27.0.10
   summary.fs.cache_hit PMID: 27.0.11
   ```

5. Edit the `/usr/pcp/pmdas/summary/expr.pmie` file to add new `pmie` expressions. If the name to the left of the assignment operator (=) is one of the PMNS names, then the `pmie` expression to the right will be evaluated and returned by the summary PMDA. The expression must return a numeric value.

   For example, consider this expression:

   ```
   // filesystem buffer cache hit percentages
   prefix = "kernel.all.io";          // macro, not exported
   summary.fs.cache_write =
                  100 - 100 * $prefix.bwrite / $prefix.lwrite;
   summary.fs.cache_hit =
                  100 - 100 * $prefix.bread / $prefix.lread;
   ```

6. Run `pmie` in debug mode to verify that the expressions are being evaluated correctly, and the values make sense.

   For example, enter this command:

   `pmie -t 2sec -v expr.pmie`

You see output similar to the following:

```
summary.fs.cache_write:      ?
summary.fs.cache_hit:      ?
summary.fs.cache_write:  45.83
summary.fs.cache_hit:   83.2
summary.fs.cache_write:  39.22
summary.fs.cache_hit:  84.51
```

7. Install the new PMDA.

   From the `/usr/pcp/pmdas/summary` directory, use this command:

   `./Install`

   You see the following output:

```
You need to choose an appropriate configuration for installation of
the ''summary'' Performance Metrics Domain Agent (PMDA).

collector collect performance statistics on this system
monitor   allow this system to monitor local and/or remote systems
both      collector and monitor configuration for this system

Please enter c(ollector) or m(onitor) or b(oth) [b] both
Interval between summary expression evaluation (seconds)? [10] 10
Updating the Performance Metrics Name Space...
Installing pmchart view(s) ...
Terminate PMDA if already installed ...
Installing files ..
Updating the PMCD control file, and notifying PMCD ...
Wait 15 seconds for the agent to initialize ...
Check summary metrics have appeared ... 8 metrics and 8 values
```

8. Check the metrics.

   For example, enter this command:

   `pmval -t 5sec -s 8 summary.fs.cache_write`

   You see a response similar to the following:

```
metric:    summary.fs.cache_write
host:      localhost
semantics: instantaneous value
```

```
units:     none
samples:   8
interval:  5.00 sec
63.60132158590308
62.71878646441073
62.71878646441073
58.73968492123031
58.73968492123031
65.33822758259046
65.33822758259046
72.6099706744868
```

Note that the values are being sampled here by `pmval` every 5 seconds, but `pmie` is passing only new values to the summary PMDA every 10 seconds. Both rates could be changed to suit the dynamics of your new metrics.

9. You may now create `pmchart` views, `pmview` scenes, and `pmlogger` configurations to monitor and archive your new performance metrics.

8.2 PCP Tool Customization

8.2.1 Stripchart Customization

The PCP tool `pmchart` produces stripchart displays of performance metrics. Refer to Section 4.1, page 21, for an extensive description of the capabilities of `pmchart`.

Customization is centered on PCP views that may be created interactively and saved via the **Save View** option in the **File** menu.

When `pmchart` is loading a view, the following directories are searched:

`.`	The current directory.
`$HOME/.pcp`	Views for each user.
`/var/pcp/config/pmchart`	The system-wide catalog of views. Any view installed here becomes visible to every `pmchart` user.

The X11 application resources for `pmchart` are in `/usr/lib/X11/app-defaults/PmChart`, and these may be edited to customize

the appearance of the display. The default update interval and other attributes are described in the pmchart(1) man page.

8.2.2 Inference Engine Customization

The PCP inference engine is presented in Chapter 6, page 71, and documented in the pmie(1) man page.

The following resources are available to aid customizing pmie:

/var/pcp/demos/pmie/*

Example pmie rules that may be used as a basis for developing local rules.

8.2.3 Snapshot Customization

The PCP snapshot production facility is presented in Section 7.2.3, page 79, and documented in the pmsnap(1) man page.

The following global files and directories influence the behavior of pmsnap:

/var/pcp/config/pmsnap/control

Defines how to produce a snapshot, including the output filename, the PCP archive folio name to be used as input, the pmchart configuration file, and command line arguments to pmchart.

/var/pcp/config/pmsnap/Summary

A pmchart configuration file to produce a sample summary snapshot in conjunction with pmsnap.

/var/pcp/config/pmlogger/config.Summary

A pmlogger configuration file that can produce an archive containing performance metrics required by the sample summary snapshot.

/var/pcp/config/pmlogger/crontab

Prototype crontab entries that may be merged with the crontab entries for root schedule the periodic execution of the archive log management scripts, for example, pmsnap.

`/var/pcp/config/pmsnap/Summary.html`

An example HTML page suitable for publishing images from the `pmsnap` examples via a Web server.

8.2.4 Icon Control Panel Customization

The gadget specification language of `pmgadgets` supports the creation of arbitrary gadget layouts and bindings to hosts and performance metrics. See Section 4.2, page 42, and the `pmgadgets`(1) man page.

8.2.5 3D Visualization Customization

The 3D scene specification language of `pmview` supports the creation of block layouts and bindings to hosts and performance metrics. See Chapter 5 and the `pmview`(1) man page.

8.3 PMDA Development

Performance Co-Pilot (PCP) is designed to be extensible at the collector site.

Application developers are encouraged to create new PMDAs to export performance metrics from the applications and service layers that are particularly relevant to a specific site, application suite, or processing environment.

These PMDAs use the routines of the `libpcp_pmda` library, which is discussed in detail by the *Performance Co-Pilot Programmer's Guide*.

Source code for several PMDAs (simple, trivial, and txmon) is provided in the `pcp.sw.demo` subsystem. When it is installed, all of the relevant files reside in directories (one per PMDA) below the `/var/pcp/pmdas` directory.

8.4 PCP Tool Development

Performance Co-Pilot (PCP) is designed to be extensible at the monitor site.

Application developers are encouraged to create new PCP client applications to monitor or display performance metrics in a manner that is particularly relevant to a specific site, application suite, or processing environment.

Client applications use the routines of the PMAPI (performance metrics application programming interface) described in the *Performance Co-Pilot Programmer's Guide*.

Source code for a sample PMAPI client (`pmclient`) is provided in the `pcp.sw.demo` subsystem, and when installed all of the relevant files reside in `/var/pcp/demos/pmclient`.

Acronyms

Table A-1 provides a list of the acronyms used in the Performance Co-Pilot (PCP) documentation, help cards, man pages, and user interface.

Table A-1 Performance Co-Pilot Acronyms and Their Meanings

Acronym	Meaning
API	Application Programming Interface
DBMS	Database Management System
DNS	Domain Name Service
DSO	Dynamic Shared Object
I/O	Input/Output
IPC	Interprocess Communication
PCP	Performance Co-Pilot
PDU	Protocol data unit
PMAPI	Performance Metrics Application Programming Interface
PMCD	Performance Metrics Collection Daemon
PMCS	Performance Metrics Collection Subsystem
PMD	Performance Metrics Domain
PMDA	Performance Metrics Domain Agent
PMID	Performance Metric Identifier
PMNS	Performance Metrics Name Space
TCP/IP	Transmission Control Protocol/Internet Protocol

Index

N

O

P

R