Performance Co-Pilot™ for ORACLE® Administrator's Guide

Document Number 007-3005-002

CONTRIBUTORS

Written by Jonathan Knispel and Bill Tuthill
Production by Allen Clardy
Engineering review by Ken McDonnell and Nathan Scott
St. Peter's Basilica image courtesy of ENEL SpA and InfoByte SpA. Disk Thrower image courtesy of Xavier Berenguer, Animatica.

Performance Co-Pilot™ for ORACLE® User's Guide
Document Number 007-3005-002

Contents

List of Figures

List of Tables

About This Guide

This guide explains how to use the Silicon Graphics Performance Co-Pilot (PCP) tools to monitor ORACLE database performance. The software described here is packaged as the Performance Co-Pilot for ORACLE product. You need both PCP (version 2.0 or later) and ORACLE (version 7 or 8) installed on your system(s) to take advantage of this software. If you have already installed this product, you may refer to the abbreviated installation instructions in Appendix A.

This section includes brief descriptions of the contents of this guide and an explanation of typographical conventions used.

What This Guide Contains

This guide contains the following chapters:

- Chapter 1, "Monitoring ORACLE Performance With PCP," provides an overview of how to monitor ORACLE databases with PCP.
- Chapter 2, "Installation Instructions," gives instructions for installing the necessary subsystems to support local databases and remote databases, tells how to upgrade existing installations, and shows how to remove PCP for ORACLE.
- Chapter 3, "Managing Multiple ORACLE Databases," describes strategies for monitoring multiple ORACLE databases on remote servers or local servers.
- Chapter 4, "ORACLE Response-Time Agent," provides information on using the *pmdaoraping* agent for measuring response time from ORACLE databases.
- Chapter 5, "ORACLE Performance Visualization Tools," provides information on several commands for visualizing ORACLE database performance.
- Appendix A, "Abbreviated Installation Instructions," gives instructions for getting PCP for ORACLE set up rapidly.
- Appendix B, "Troubleshooting PCP for ORACLE," provides troubleshooting help for this product.

Conventions Used in This Guide

These type conventions and symbols are used in this guide:

Bold Function and subroutine names, language keywords and data types, literal command-line arguments (options), nonalphabetic data types, and operators.

Italics BNF entries, executable names, filenames, glossary entries (online, these show up underlined), IRIX commands, manual/book titles, new terms, onscreen button names, program variables, tools, utilities, variable command-line arguments, variable coordinates, and variables to be supplied by the user in examples, code, and syntax statements.

`Fixed-width` Error messages, prompts, and onscreen text.

`Bold Fixed-width`
User input, including keyboard keys (printing and nonprinting); literals supplied by the user in examples, code, and syntax statements.

ALL CAPS Environment variables, operator names, directives, defined constants, macros in C programs.

Helvetica Bold Hardware labels.

"" (Double quotation marks) Onscreen menu items and references in text to document section titles.

() (Parentheses) Following function names, surround function arguments or are empty if the function has no arguments. Following commands, surround the reference page (manual page) section number.

[] (Brackets) Surrounding optional syntax statement arguments.

`#` Shell prompt for the superuser (root).

`%` C shell prompt for users other than root.

This guide uses the standard UNIX convention for referring to reference pages. The entry name is followed by the section number in parentheses. For example, pmchart(1) designates the online manual page for the *pmchart* command. If this reference page is installed, you can click on this entry to display the information.

Additional Resources

The following sections describe resources for obtaining further information.

PCP Documentation

The *Performance Co-Pilot User's and Administrator's Guide* describes your PCP software in detail. This is your main resource for using the base PCP software.

The *Performance Co-Pilot Programmer's Guide* provides information on the application programming interface (API) for extending PCP services, and the steps necessary to add new collection agents into the PCP framework.

Reference Pages

Table i shows some important reference pages for PCP commands and libraries.

Table i Selected PCP Commands and Libraries

Reference Page	Summary Information
cron.pmdaily(1)	Administration of Performance Co-Pilot archive log files
dkvis(1)	Visualize disk I/O rates and performance
mpvis(1)	Visualize CPU utilization/performance
nfsvis(1)	Visualize performance of NFS statistics
orachart(1)	Plot ORACLE database performance metrics against time
PMAPI(3)	Introduction to the Performance Metrics API
pmcd(1)	Performance metrics collector daemon
pmchart(1)	Plot performance metrics against time
pmclient(1)	A simple performance metrics client
PMDA(3)	Introduction to the Performance Metrics Domain Agent support library
pmdacisco(1)	Cisco router performance metrics domain agent (PMDA)
pmdasummary(1)	Summary performance metrics domain agent (PMDA)

Table i (continued) Selected PCP Commands and Libraries

Reference Page	Summary Information
pmgadgets(1)	A miniature graphical performance metrics viewer
pmgirix(1)	A miniature IRIX performance metrics viewer
pmgora(1)	A miniature ORACLE database performance metrics viewer
pmie(1)	Inference engine for performance metrics
pminfo(1)	Display information about performance metrics
pmkstat(1)	High-level system performance overview
pmns(4)	The performance metrics namespace

Release Notes

Release notes provide specific information about the current release. Each optional product has its own set of release notes. Release Notes are available online through the *relnotes* command, or graphically through the *grelnotes* command.

Web Resources

If you have access to the World Wide Web you can find up-to-date information about Performance Co-Pilot at this location:

- http://www.sgi.com/software/co-pilot/

At a related Web site, you can find additional Silicon Graphics information, including the Technical Publications Library:

- http://techpubs.sgi.com

Chapter 1

Monitoring ORACLE Performance With PCP

Within the collection infrastructure of Performance Co-Pilot, the *pmcd* process (the Performance Metrics Collector Daemon) is responsible for gathering performance metrics relating to the systems on which it is run. The *pmcd* process manages a group of Performance Metrics Domain Agents (PMDAs, or simply agents), each of which is responsible for gathering data within a specified domain.

ORACLE database performance metrics are collected by an agent called *pmdaoracle*; see pmdaoracle(1) for more information.

As of PCP release 2.0, the Performance Metric Name Space (PMNS) is distributed. Because PMDAs now transmit their namespace as part of the protocol, it is not necessary to install a special namespace on remote PCP monitoring systems.

See the pmcd(1) reference page for more detailed information about *pmcd*, agents, and domains. Refer to the *Performance Co-Pilot User's and Administrator's Guide* for more information on PCP in general, especially if you do not understand the terms, concepts, or instructions presented in this document.

PCP for ORACLE Requirements

If you wish to use metrics from an ORACLE database (for example, using the *orachart* command to plot database performance; see orachart(1) or Chapter 5), there are certain requirements on the system where the database resides and the system from which the database is to be monitored:

- On the system where the database(s) reside, there must be a *pmcd* process running. Furthermore, the *pmcd* process must be configured to run *pmdaoracle* for the database(s) being monitored. This is the pcp_ora.sw.collector subsystem.
- On the monitoring system, PCP monitoring tools must be installed, along with some ORACLE performance monitoring tools. This is the pcp_ora.sw.monitor subsystem.

Figure 1-1 shows a typical use of the Performance Co-Pilot to monitor an ORACLE database running on a server system. The server system often has no graphics capability, so a graphical workstation is used to monitor the database.

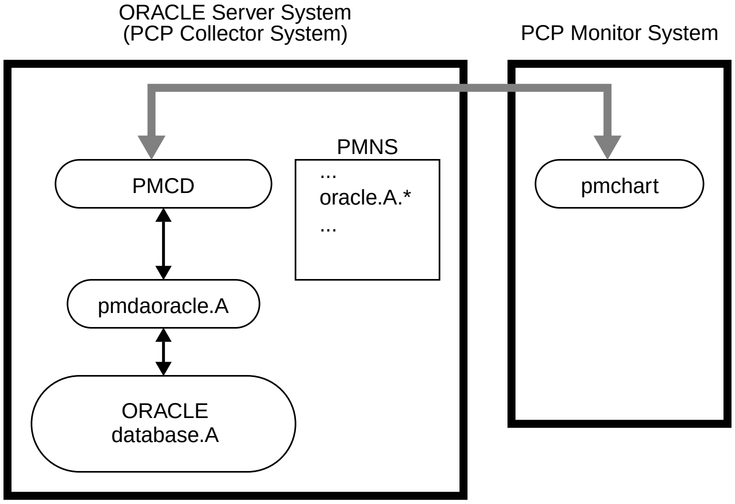


Figure 1-1 Monitoring an ORACLE Instance on a Remote Server

Figure 1-2 shows a more complicated arrangement, with multiple ORACLE servers to be monitored. The server system contains namespace entries for each monitored database. (Although server systems do not need to monitor themselves, PCP requires export of namespaces for each ORACLE database to enable fully distributed operation.)

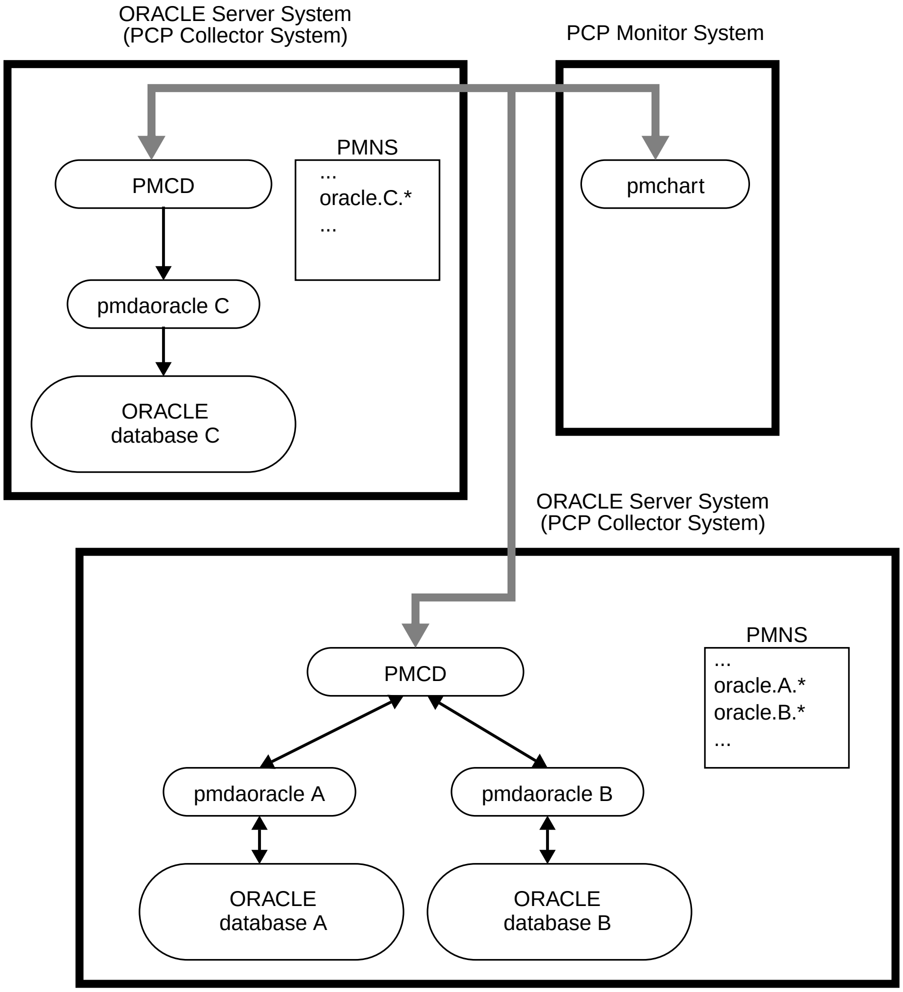


Figure 1-2 Monitoring Several ORACLE Instances on Multiple Servers

Overview of Installation Scenarios

Installation of PCP for ORACLE involves at least one PCP collector system installation, and typically at least one PCP monitor system installation. These two installations are performed with the *Install* shell script located in the */var/pcp/pmdas/oracle* directory.

Run the *Install* script once per database instance, after the PCP software and PCP for ORACLE packages have been installed.

PCP Collector Systems

On a system with an ORACLE database running:

1. Make and install the ORACLE agent, *pmdaoracle*.
2. Alter the PMCD configuration file to run *pmdaoracle*.
3. Augment the PCP namespace to include the ORACLE metrics.
4. Create an ORACLE database user for *pmdaoracle* to use.

This series of operations, largely automated by the *Install* script, must be repeated for each ORACLE database instance running on a system.

PCP Monitor Systems

On a system used to monitor the ORACLE metrics on a remote system, install only the PCP monitor components.

Chapter 2

Installation Instructions

This chapter provides you with specific installation instructions for the standard scenarios. For more complex installations, use the instructions in this chapter together with the information in Chapter 3, “Managing Multiple ORACLE Databases.”

In the remainder of this guide, “the database system” means the remote system that is running the ORACLE database instance you wish to monitor, “the database instance” refers to the database instance on the database system, and “the monitoring system” refers to the system where the PCP monitoring tools are running.

PCP Collector Installation

On a PCP collector system, follow the procedure described in “Collector Installation Procedure” on page 7 once for each ORACLE database instance that you wish to monitor. This procedure installs one PCP agent for each ORACLE database, and ensures that the associated ORACLE performance metrics can be exported into the PCP framework.

Collector Prerequisites

Prerequisites for the collector system are as follows:

- The *pcp.sw*, *pcp_eoe.sw*, and *pcp_ora.sw.collector* product images must have been installed, using either *swmgr* or *inst*; see the inst(1M) reference page.
- The *pmcd* daemon must be running on the PCP collector system.
- You must have superuser (root) permission to run *Install*.
- You must know the *ORACLE_HOME* and *ORACLE_SID* environment variables for the desired database instance.
- The ORACLE database instance must be up and running.
- You must have DBA access to the database instance as the SYS database user.

The *pmdaoracle* agent needs access to the database, so an ORACLE user must be created on your system for this. The *Install* script automatically generates SQL statements to create the ORACLE database user and grant that user suitable privileges for the database (that is, "select" access to the dynamic V$* views).

Before running *Install* you must decide on a name for this ORACLE user. You have two options: create a conventional user with a password (such as `pcp/pcp`), or use a default login (such as `ops$pcp`). If you are upgrading an existing PCP installation, the old *pmcd* configuration file (*/etc/pmcd.conf*) shows the previously specified user. See "Upgrading an Existing Installation" on page 10 or the pmcd(1) and pmdaoracle(1) reference pages.

If a system contains multiple ORACLE database instances, it is strongly recommended that you use the same ORACLE database user for all of the ORACLE database instances and corresponding instances of *pmdaoracle* on that system. If you elect to use a default login for *pmdaoracle* (an `ops$...` user), then it is essential that the same user be specified for all database instances. However, note that the ORACLE database user may differ between systems, if necessary.

Collector Security Issues

There are some security issues. Under the "user with a password" approach, the user's name and password appear on the command line of the ORACLE agent run by *pmcd*. Disclosure of this information may allow unauthorized users to connect to the database as this PCP for ORACLE user, and have read access to the performance data made available by ORACLE. However, note that no access is granted to any other data in the database.

If possible, it is better to create an `ops$...` login instead, so that no password appears on the command line. If you choose this option, you must create an IRIX user account with the appropriate name prior to running the *Install* script (for example, an account named `pcpora` for the `ops$pcpora` ORACLE user). If this account is used only for the ORACLE agent, it is strongly recommended that you disable standard logins for it. Refer to *IRIX Admin: System Configuration and Operation* for information on the procedure to disable logins.

Caution: If an ORACLE database user with the same name already exists, part of the installation process grants that user *select* (only) access on several of the ORACLE dynamic performance views.

Collector Installation Procedure

Run *Install* by performing the tasks outlined below. The ORACLE database must be up and running before you can perform the following procedure.

To install a collector for one ORACLE database instance, follow these steps:

1. Become superuser using the *su* command or by logging in as root.
2. Set the environment variables *ORACLE_HOME* and *ORACLE_SID* for the ORACLE database you want.

   Note: If the *ORACLE_SID* is not a valid PCP namespace identifier, run *Install* with the **-r** option to specify an alternative name for this database's ORACLE metrics in the namespace. Valid identifiers consist of an initial letter followed by zero or more letters, digits, or underscores. For example, if your *ORACLE_SID* is 7.1.6 you might specify **-r inst_7_1_6**.
3. Enter the following command, replacing *XXX* with the ORACLE database user identification (and optional password) you chose:

   ```
   # ./Install XXX
   ```

   For example, if you elected to use a default login of `ops$pcp`, you would enter:

   ```
   # ./Install 'ops$pcp'
   ```

   The single quotes in the command above stop the shell from trying to evaluate `$pcp` as a shell variable.

   If instead you decided to create a user named `pcpora` with a password of `secret9`, you would enter:

   ```
   # ./Install pcpora/secret9
   ```

   If there are any errors, your PCP configuration is restored to its previous state before you ran *Install*.

4. When finished, *Install* prints a few post-installation tasks that you must perform. In the example text below, *DBNAME* is the name of your ORACLE database.[1] You see a message similar to the following:

   ```
   You must run setup.DBNAME.sql as SYS on the ORACLE_SID database
   to grant the Oracle PMDA access to the dynamic performance tables.
   ```

   For example:

   ```
   # $ORACLE_HOME/bin/sqldba lmode=y
   SQLDBA> connect sys
   Password:
   SQLDBA> @setup.DBNAME.sql
   ```

   Once this has been done, start the *pmdaoracle* PMDA to verify that the installation has completed correctly:

   ```
   # killall -HUP pmcd
   # ./Verify-DBNAME
   ```

   When performing these tasks, make sure that you use the *ORACLE_HOME* and *ORACLE_SID* that were in the environment when you ran *Install*.

The *pmcd* daemon is started automatically when the system reboots. ORACLE databases too are often started automatically when the system reboots. Because they take some time to initialize and become available, they are often started in the background so as not to hold up the reboot sequence. If PMCD is started before the ORACLE database is available, the ORACLE PMDA will not be able to connect to the database.

If it has no connection, *pmdaoracle* periodically attempts to (re)connect to the database.

Note that the *pmcd* process does this only when ORACLE performance metrics are requested. In addition, there is a cooling-off period between connection attempts, to prevent a flood of requests for ORACLE metrics from generating a corresponding flood of ORACLE database connection attempts. See the pmdaoracle(1) reference page for more details.

If you do not want the common prefix for the names of PCP ORACLE metrics in the namespace to correspond with the ORACLE database instance name, the *Install* command's **-r** option allows you to specify an alternative. For example, the following commands cause *pmdaoracle* to run on the *demo* database instance:

```
# ORACLE_SID=demo
# ./Install -r pqo_demo 'ops$pcp'
```

[1] *DBNAME* should be the same as the *ORACLE_SID* of your database unless you gave it a different name by using the **-r** option when you ran *Install*.

However, the metrics appear in the namespace under *oracle.pqo_demo*, as opposed to *oracle.demo*. This can be useful when several databases on different systems have the same *ORACLE_SID*. Once started, *pmdaoracle* in this case creates an activity record in */var/adm/pcplog/oracle-pqo_demo.log* to report unusual or error conditions that occur while the PMDA is running.

Customized Configuration Files

The base PCP product and PCP for ORACLE contain a number of configuration files that can be modified to reflect the local configuration or product customizations. When you install a new version of PCP and the installation software discovers that you have customized a file, it performs one of two possible actions:

- It may save the customized file (by appending *.O* to its name) then put the new version of the file in its place.
- It may leave the customized version of the file in place and save the new one to a temporary name (by appending *.N* to the default name).

Use the following command to find conflicts, then reconcile differences as needed, using *xdiff* to compare files (see xdiff(1L) for details).

```
$ showfiles -cCH pcp\*
```

PCP Monitor Installation

On a PCP monitor system, the installation procedure is quite simple, involving just the installation of graphical monitoring tools.

Monitor Prerequisites

Prerequisites for the monitoring system are as follows:

- The *pcp.sw*, *pcp_eoe.sw*, and *pcp_ora.sw.monitor* product images must have been installed, using either *swmgr* or *inst*; see the inst(1M) reference page.
- You must have superuser (root) permission to run *Install*.
- *Install* must have been run on the database system to install an ORACLE agent for the database instance.

In the remainder of this guide, “the database system” means the remote system that is running the ORACLE database instance you wish to monitor, “the database instance” refers to the database instance on the database system, and “the monitoring system” refers to the system where the PCP monitoring tools are running.

Upgrading an Existing Installation

Given the variation possible in ORACLE installations, it is not possible to automatically upgrade an existing PCP for ORACLE installation by typing a single upgrade command. Fortunately, upgrading is a straightforward process.

Note: The PCP for ORACLE directory moved from */usr/demos/PerfCoPilot/pmdas/oracle7* in PCP for ORACLE 1.0, then to */usr/pcp/pmdas/oracle7* in PCP for ORACLE 1.1, then to */var/pcp/pmdas/oracle7* in PCP for ORACLE 1.2, and finally to */var/pcp/pmdas/oracle* in this release, PCP for ORACLE 2.0. The *Install* and *Remove* scripts described in this document should be run from this new directory.

Follow this procedure:

1. Make a copy of the current */etc/pmcd.conf*. Use this command:

   ```
   # cp /etc/pmcd.conf /etc/pmcd.conf.old
   ```

   If you have customized the flags used for *pmdaoracle*, you can use the saved version as a reference for replacing your customizations. For more information on flags, see the pmdaoracle(1) reference page. Even if you did not customize flags, the *pmcd.conf.old* file contains all the previous domain assignments if you need to reconstruct them.

2. Find the username and password of each *pmdaoracle*.

   To upgrade each *pmdaoracle* you need to know the username and password it uses to connect to its database. Use the *grep* command to search for the string *pmdaoracle* in */etc/pmcd.conf*. The *ORACLE_HOME* and *ORACLE_SID* appear as the last two entries on each line. If there is a **-c** flag, it is followed by the username/password pair that the agent uses for database connections. If there is no **-c** flag, an `ops$...` login is being used; the *pmdaoracle* executable has the **setuid** mode bit set, and the IRIX user who owns the file determines the `ops$...` user for ORACLE.

For example, if *pmdaoracle* is **setuid** and owned by *pcpora*, as in the example below, the corresponding ORACLE username is `ops$pcpora`.

```
# cd /var/pcp/pmdas/oracle
# ls -l pmdaoracle
-rwsr-xr-x    1 pcpora   sys     3119516 Nov 11 12:38 pmdaoracle
```

3. Use the instructions earlier in this chapter to install the new version of PCP for ORACLE. Use the same ORACLE username (and optional password) that the existing *pmdaoracle* uses. Remember that there are two kinds of installation: one for systems configured as PCP collectors, and one for PCP monitor systems.

   The ORACLE username and password are necessary only for PCP collector installations. PCP collector systems must be upgraded because the agent has been enhanced to extract new metrics. PCP monitor systems must also be upgraded so the new metrics appear in the PCP namespace.

4. If you have trouble upgrading, you may use the *Remove* script to delete the *pmdaoracle*, then install it from scratch by following the instructions in “PCP Collector Installation” on page 5 and in “PCP Monitor Installation” on page 9.

5. Reapply any customizations you might have made to ORACLE agents in the */etc/pmcd.conf* file on each system running an ORACLE database. Use the saved version from step 1 (*/etc/pmcd.conf.old*) as a guide.

6. Check that metrics are available from each system. For example:

   ```
   $ pminfo -f oracle
   ```

7. Remove any old PCP for ORACLE directories and their contents. For example:

   ```
   # rm -rf /usr/demos/PerfCoPilot/pmdas/oracle7
   # rm -rf /usr/pcp/pmdas/oracle7.0
   # rm -rf /var/pcp/pmdas/oracle7
   ```

Removing an Installation

The *Remove* script is the reverse of *Install*. The *Remove* script is run in the same way on both the PCP collector systems and on the PCP monitor systems. *Remove* must have the name of the ORACLE metrics in the PCP namespace (without the *oracle* part of the name) as a parameter.

For example, the following sequence of Bourne shell commands installs a PCP collector for the *demo* database instance:

```
# ORACLE_SID=demo
# export ORACLE_SID
# ORACLE_HOME=/oracle/home
# export ORACLE_HOME
# ./Install ’ops$pcp’
```

To undo the actions of the above commands, use the following command:

```
# ./Remove demo
```

The same command works on PCP monitor systems, provided you specify the correct name for the ORACLE metrics to be removed.

If you are not sure what ORACLE metrics are available in the namespace, the following command produces a list of names suitable for use as parameters to *Remove*:

```
$ pminfo oracle | cut -f2 -d. | sort -u
```

Remember that the name of the ORACLE metrics in the namespace should be used as the parameter. If the **-r** option was used to override *ORACLE_SID* when *Install* added the ORACLE metrics to the namespace, the name of the ORACLE metrics in the namespace may not correspond to *ORACLE_SID*. Consider the following sequence of commands on a PCP collector system:

```
# ORACLE_SID=demo
# export ORACLE_SID
# ORACLE_HOME=/oracle/home
# export ORACLE_HOME
# ./Install -r pqo_demo ’ops$pcp’
```

To undo the effects of this installation, enter the following command:

```
# ./Remove pqo_demo
```

Remove checks that the corresponding entry in */etc/pmda.conf* is removed, notifies *pmcd* to ensure termination of the PMDA, and culls all of the associated ORACLE metrics from the PCP namespace. If this is the only existing ORACLE PMDA, *Remove* also deletes the *pmdaoracle* executable.

Remove does not remove any users added to the ORACLE database for monitoring the ORACLE performance statistics—this must be done separately.

Chapter 3

Managing Multiple ORACLE Databases

This section contains information for organizations with multiple ORACLE databases.

Planning for Multiple Databases

Before planning a multiple database installation, it is a good idea to be thoroughly familiar with the contents of this guide. In addition to reading this guide, try actually installing and using PCP for ORACLE on a single database instance. Try monitoring a database from one or more other systems. Try using **-r** to specify a name other than the *ORACLE_SID* on one of the remote systems. Try removing PCP for ORACLE.

Each PCP performance metric has a name (for example, *irix.swap.pagesout*), which is mapped to a more compact internal representation for the PCP namespace. This internal representation is known as a PMID (performance metric identifier). The *pmcd* process on each system manages a collection of Performance Metrics Domain Agents (PMDAs, or simply agents). All requests for performance metrics on a system go to the *pmcd*.

Each of a *pmcd*'s agents is responsible for gathering performance data from a domain, such as IRIX, ORACLE, Cisco routers, and so on. Each agent that *pmcd* manages is assigned a unique numeric domain identifier. Every PMID contains a domain identifier that tells its *pmcd* which agent is responsible for supplying the metric.

For example, there is only one IRIX agent on any given system, because only one copy of IRIX can run on a system. Domain number 1 is reserved for IRIX on all PCP collector systems. So the namespace on every system maps all IRIX performance metrics to a set of PMIDs that all have the domain number 1. Similarly, every PMCD knows that the IRIX agent has domain 1, so any request for a PMID with a domain of 1 is for the IRIX agent.

This works well provided that there is only one copy of a particular agent on a system. Now consider a system with more than one ORACLE database. Running a single "super" agent able to supply information from all of the ORACLE databases on a system is not feasible for a number of reasons (performance, fault resilience, complexity, and so on). Instead, a separate agent is created for each database instance on the system.

Each PCP for ORACLE agent supplies performance metrics from a single database instance. If you have multiple databases, a separate copy of the same agent program is running for each, but with different command-line arguments. The metrics supplied by the agents are distinguished by giving them different names in the namespace, and different domain numbers.

Consider a system with two databases named *production* and *test*. There is one copy of *pmdaoracle* (the PCP for ORACLE agent) running using the *production* database and another using the *test* database. The latch miss statistics for the two databases are available as *oracle.production.latch.misses* and *oracle.test.latch.misses*, respectively. The domains for these two metrics in the namespace are different, but they must agree with the domains used by *pmcd* for the two ORACLE agents. This is also the case for remote systems wishing to monitor the *production* and *test* databases.

Multiple Databases on One Database Server

In the following example, there is a single database server system with two databases: *production* and *test*.

On the database server system, an agent is required for each database and the namespace must be updated to include ORACLE metrics for each database. Using the Bourne shell, the superuser enters the following commands:

```
# cd /var/pcp/pmdas/oracle
# ORACLE_HOME=/oracle/home
# export ORACLE_HOME
# ORACLE_SID=production
# export ORACLE_SID
# ./Install pcpora/pcpora
```

The *Install* command above prints instructions describing how to add the necessary database user, and generates a *Verify* script for the *production* database, *Verify-production* in the */var/pcp/pmdas/oracle* directory. Continuing:

```
# ORACLE_SID=test
# export ORACLE_SID
# ./Install pcpora/pcpora
```

Install prints instructions describing how to add the necessary database user, and generates a *Verify* script for the *test* database, *Verify-test* also in */var/pcp/pmdas/oracle*.

Now *pminfo* displays the following output:

```
$ pminfo -m oracle | grep reqdist
oracle.production.reqdist PMID: 32.5.0
oracle.test.reqdist PMID: 33.5.0
```

The above command sequence prints out the PMID of one metric from each of the databases. The first integer in the dotted triple is the domain assigned to that database by the *Install* script. In this example, the *production* database is domain 32 and the *test* database is domain 33.

Note: A common error when fetching metrics in a distributed environment is forgetting to specify the host from which the metrics are to be fetched. If you forget to specify a database server system when fetching metrics, *localhost* gets used by default, and no meaningful values are returned. The following example shows a typical sequence, corrected:

```
$ pmval -t 10 oracle.orders.all.table.scanrows
pmval: pmLookupDesc: Unknown or illegal metric identifier

$ pmval -h bigboy -t 10 oracle.orders.all.table.scanrows

metric:     oracle.orders.all.table.scanrows
host:       bigboy
semantics: cumulative counter (converting to rate)
units:      count (converting to count / sec)
samples:    all
interval:   10.00 sec

    53.6
   103.1
    87.4
    97.8
```

Multiple Databases and Multiple Database Servers

If your organization has several systems, each with one or more ORACLE database instance, then you may wish to assign unique domain numbers to each database instance. This is usually less confusing than having two ORACLE agents for the same PCP domain on different database servers. To accomplish this, you must assign unique domains to the databases manually. The *Install* script is not sophisticated enough to search through an entire network looking for ORACLE databases.

Create and maintain a central registry of domain numbers. The registry specifies the system's hostname, the database instance name, and the corresponding PCP ORACLE agent's domain number. Keep the registry up to date and periodically check it against all of the */etc/pmcd.conf* files in your organization.

Assign a unique domain identifier to each database you intend to monitor. The range of domain identifiers reserved for ORACLE databases instances is 32 to 39, inclusive. Additional numbers in the range 128 to 254 may be used if they are not already taken by other agents.

The following example illustrates the process of configuring PCP for ORACLE support for multiple databases on multiple systems. Assume that there are three systems:

- *sys1*, with two ORACLE databases: *production* and *test*
- *sys2*, with three ORACLE databases: *dev*, *proto*, and *test*
- *sys3*, with one ORACLE database: *demo*

First, allocate six unique domain identifiers, as shown in Table 3-1. Although the *test* databases on systems *sys1* and *sys2* have the same name, they are different database instances.

Table 3-1 Unique Domain Identifiers

System	Database	Domain
sys1	production	32
sys1	test	33
sys2	dev	34
sys2	proto	35
sys2	test	36
sys3	demo	37

There are two databases named *test*. You must give them different names in the PCP namespace to distinguish them, such as *test_sys1* and *test_sys2*. You need only rename them in the PCP namespace; there is no need to re-create the database with a new name on the system.

Next, install an agent for each database running on a database system.

The **-d** option manually specifies the domain. For the *test* databases, **-r** specifies an alternate name for the databases in the namespace. For example, using the Bourne shell on system *sys1*, the superuser enters the following commands:

```
# cd /var/pcp/pmdas/oracle
# ORACLE_HOME=/oracle/home
# export ORACLE_HOME
# export ORACLE_SID=production
# export ORACLE_SID
# ./Install -d 32 pcpora/pcpora
```

Add any necessary database users and run the *Verify* script for the *production* database, */var/pcp/pmdas/oracle/Verify-production*. Continuing:

```
# ORACLE_SID=test
# export ORACLE_SID
# ./Install -d 33 -r test_sys1 pcpora/pcpora
```

Add any necessary database users and run the *test* database *Verify* script, *Verify-test*.

The same process takes place on the systems *sys2* and *sys3*. For these, only the *Install* commands are shown. After setting up the environment for the *dev* database on *sys2*:

```
# ./Install -d 34 pcpora/pcpora
```

After setting up the environment for the *proto* database on *sys2*:

```
# ./Install -d 35 pcpora/pcpora
```

After setting up the environment for the *test* database on *sys2*:

```
# ./Install -d 36 -r test_sys2 pcpora/pcpora
```

And after setting up the environment for the *demo* database on *sys3*, enter this command:

```
# ./Install -d 37 pcpora/pcpora
```

You do not have to add all the databases to every system's namespace. For example if a system named *dbspy* is used to monitor only the *production* database on *sys1* plus the *dev* and *test* databases on *sys2*, the following commands suffice. As superuser on *dbspy*, enter the following commands after installing PCP for ORACLE:

```
# cd /var/pcp/pmdas/oracle
# ./Install -N -d 32 -r production
# ./Install -N -d 34 -r dev
# ./Install -N -d 36 -r test_sys2
```

In the example above, the hostname was appended to the database name for the *test* databases. For complex ORACLE database deployments, it is a good idea to do this for all the database names, as an aid to remembering which database resides on which system. Remember that you must specify the correct host from which to fetch the metrics. This is easier if the metric name actually contains the correct hostname.

If you have a collection of workstations potentially used to monitor any database, augmentation is not necessary because of distributed namespaces.

Multiple ORACLE Homes on One System

Because *pmdaoracle* is statically linked to the ORACLE libraries in *ORACLE_HOME*, different ORACLE homes (for different versions of ORACLE) produce different code in the agent. Although it is unusual to have more than one version of ORACLE running on a system, the *Install* script supports this situation by appending the name of the ORACLE instance to *pmdaoracle*.

Chapter 4

ORACLE Response-Time Agent

The PCP for ORACLE package includes a second ORACLE-specific agent, *oraping*. The *oraping* agent periodically executes an SQL query against some ORACLE database, measures the response time from the ORACLE server, and exports that response time as a performance metric into the PCP infrastructure.

Installing oraping

Because *oraping* is a PCP PMDA, it must be installed on a PCP collector system, where a *pmcd* process is running. Prerequisites are as follows:

- The *pcp.sw* and *pcp_ora.sw.collector* product images must have been installed, using either *swmgr* or *inst*; see Chapter 2.
- The *pmcd* daemon must be running on the PCP collector system.
- You must have superuser (root) permission to run *Install*.
- You must know the *ORACLE_HOME* and *ORACLE_SID* environment variables for the desired database instance.
- The ORACLE database instance must be up and running.
- You must have DBA access to the database instance as the SYS database user.
- ORACLE Pro*C must be installed (*oraping* is supplied as Pro*C source code).
- If the database instance to be probed by *oraping* is remote, ORACLE SQL*Net must be installed (*oraping* executes as a client of the ORACLE server instance).

The installation procedure is as follows:

1. Become superuser using the *su* command or by logging in as root.
2. Enter the following commands:

```
# cd /var/pcp/pmdas/oraping
# ./Install
```

3. Answer questions from the *Install* script as follows:

```
Use ORACLE_HOME=/oracle/home? y
ORACLE_SID is not set in the environment. Please enter it now: demo
Do you want to customize the ping interval? [n] y
How many seconds between pings? 30
The oraping agent requires an ORACLE username (and password) to
connect to the database, create a table, and add some rows to it.
Please enter the username and password for a suitable user.
ORACLE user/password: username/password
Making the oraping agent...
done
You need to choose an appropriate configuration for installation
of the “oraping” 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

Updating the Performance Metrics Name Space (PMNS)...
Installing pmchart view(s)...
Terminate PMDA if already installed...
Installing files...
        make -f Makefile.ora install
Updating the PMCD control file, and notifying PMCD...
******************************
* You must run SQL script /var/pcp/pmdas/oraping/setup.sql as ORACLE
* user SYS on the demo database to grant the oraping PMDA access.
*
* For example as the ORACLE DBA:
*    $ $ORACLE_HOME/bin/svrmgrl
*    SVRMGR> connect sys
*    Password: ....
*    SVRMGR> @/var/pcp/pmdas/oraping/setup.sql
*
* Once this has been done, start the oraping PMDA:
*    # /etc/init.d/pcp start
******************************
```

4. Follow the commands shown at the end of the installation messages, which provide the necessary database access privileges for the *oraping* PMDA.

Once started, the *oraping* agent creates an activity log in */var/adm/pcplog/oraping.log* to record unusual conditions or errors that might arise, such as failed SQL commands.

Customizing oraping

The *oraping* agent may be deployed differently to quantify the following ORACLE quality of service measures:

- Local ORACLE server performance (*oraping* running on the same system as the ORACLE server).
- ORACLE server performance as seen from remote client applications (use *oraping* where the client applications run; for example, on a workstation connected to the LAN, and on a system connected to the ORACLE server by means of the WAN).
- ORACLE server performance on platforms other than Silicon Graphics systems (use *oraping* on Silicon Graphics systems to probe ORACLE on another platform).

The *oraping* agent is distributed in source code format to encourage the following customizations:

- Replace the generic SQL query by something that is more representative of the workload in the target environment.
- Add further query types and export response times for each query type.

Refer to the *Performance Co-Pilot Programmer's Guide* for a complete description of how to create and integrate new PMDAs into the PCP infrastructure.

Chapter 5

ORACLE Performance Visualization Tools

Performance Co-Pilot for ORACLE offers several visualization tools, including *orachart* to display various performance graphs for ORACLE instances, *pmgora* to continuously monitor disk and ORACLE file performance, and *pmgoraping* to monitor ORACLE server response time (quality of service).

Graphing Performance With orachart

The *orachart* command (see orachart(1)) displays a number of two-dimensional graphs showing performance of a given ORACLE database instance that is being (or has been) monitored with *pmdaoracle* (see pmdaoracle(1)).

For *orachart* you must specify a configuration file (see orachart(1) for the syntax), which *orachart* uses to generate a *pmchart* configuration file (see pmchart(1) for the syntax) for a specific ORACLE instance.The default configuration directory is */var/pcp/config/orachart*, and configurations supplied with the product are shown in Table 5-1.

Table 5-1 Default orachart Configuration Files

Filename	Information Supplied
Activity	Dirty queue length, database gets and changes, disk reads and writes
BufferLatch	Gets and immediate gets (from V$LATCH view), most recent and average
DBWR	Important statistics (from V$SYSSTAT) for database writer, DBWR
IPC	Number of messages sent and received between ORACLE processes
Parallel	Number of SELECT, DDL, and DML statements parallelized
SQL	Memory and disk sorts, parses, executions, and cache hits by SQL commands
Scans	Table scans (short and long), table rows scanned, and table blocks scanned
Transact	User commits, user rollbacks, redo writes by LGWR, and transaction rollbacks

To monitor database activity for database instance *CTC2* on host *thebeast*, the following command produces a window similar to that shown in Figure 5-1:

```
$ orachart -i CTC2 -h thebeast Activity
```

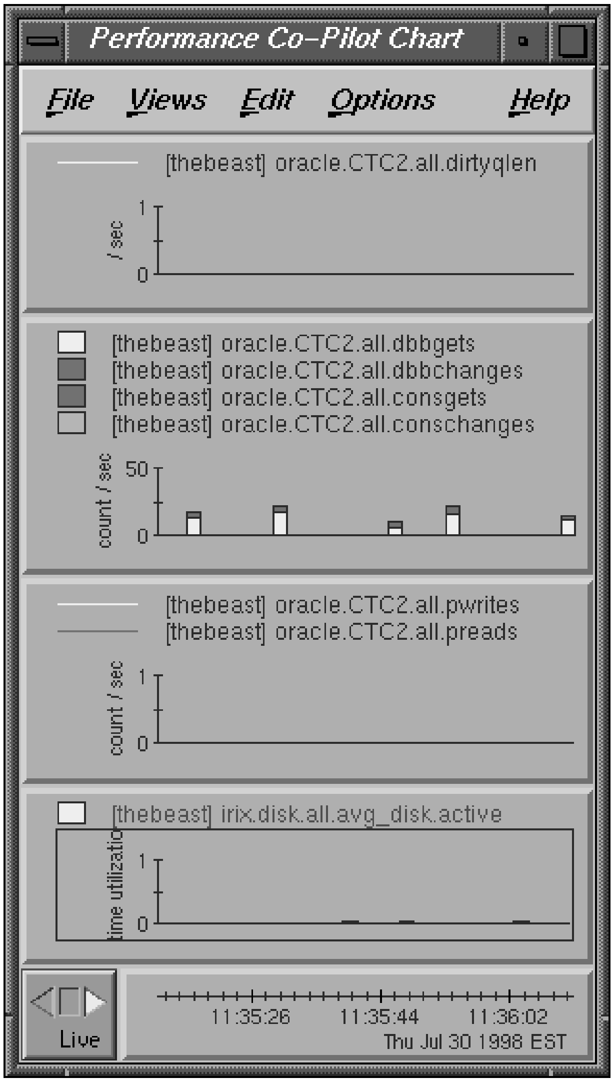

Performance Co-Pilot Chart
File
Views
Edit
Options
Help
[thebeast] oracle.CTC2.all.dirtyqlen
/ sec
1
0
[thebeast] oracle.CTC2.all.dbbgets
[thebeast] oracle.CTC2.all.dbbchanges
[thebeast] oracle.CTC2.all.consgets
[thebeast] oracle.CTC2.all.conschanges
count / sec
50
0
[thebeast] oracle.CTC2.all.pwrites
[thebeast] oracle.CTC2.all.preads
count / sec
1
0
[thebeast] irix.disk.all.avg_disk.active
time utilizatio
1
0
Live
11:35:26
11:35:44
11:36:02
Thu Jul 30 1998 EST

Figure 5-1 orachart Window With Activity

The **-i** option is needed only for systems running multiple database instances, and then only if the environment variable *ORACLE_SID* is not set. The **-h** option is required if the database instance resides on a remote host.

If you want to create a custom database performance view, use the standard *pmchart* facilities: choose Edit > New Chart and then use the Metric Selection window to select the exact combination of monitored data that you want graphed.

Monitoring Disk Performance With pmgora

Like *pmgirix* (see pmgirix(1)), *pmgora* displays a window showing continuously updated performance symbols for a given ORACLE database instance that is being monitored with *pmdaoracle* (see pmdaoracle(1)).

Actually, *pmgora* generates a *pmgadgets* configuration file for monitoring throughput in ORACLE database files, then calls *pmgadgets* with that configuration; see pmgadgets(1). The *pmgora* window displays two LEDs for each ORACLE file showing read/write activity by the monitored ORACLE instance on that file. Beneath the file list, a single LED shows the actual I/Os per second (IOPs) for each disk identified in the system. The connecting lines between disk LEDs represent disk controllers, giving an indication of I/O parallelization across both disks and controllers.

To monitor file activity for database instance *Two* on host *moomba*, the following command produces a window similar to that shown in Figure 5-2:

```
$ pmgora -i Two -h moomba
```

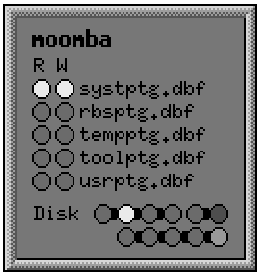


Figure 5-2 Default pmgora Window

For complete information about *pmgora*, including how to customize gadget options, see the pmgora(1) and pmgadgets(1) reference pages.

Monitoring Response Time With pmgoraping

Like *pmgora*, *pmgoraping* generates a *pmgadgets* configuration file for monitoring ORACLE response time. This is a useful tool for monitoring performance as perceived by end users, and provides insight into the quality of service provided by a particular ORACLE database instance.

Note: *pmgoraping* requires the *oraping* PMDA (described in Chapter 4) to be installed and running.

To monitor response time for database instance *Two* on host *moomba*, the following command produces a window similar to that shown in Figure 5-3:

```
$ pmgoraping -i Two -h moomba
```

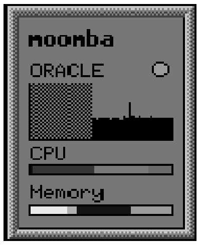
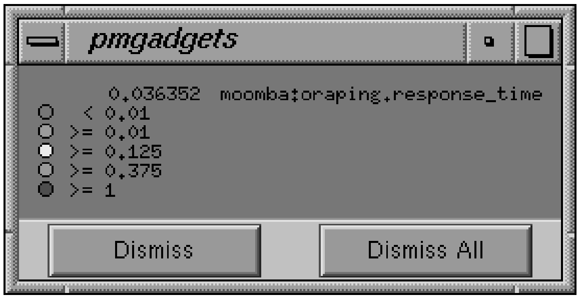


Figure 5-3 Default pmgoraping Window

The moving ORACLE graph displays response latency over time, in proportion to the maximum response time recorded, or the maximum value if you specified the **-m** option. The LED light shows green for good performance, yellow for borderline conditions, and red for poor performance. “Good, borderline, and poor” are ratios derived from the expected response time specified by the **-m** option.

Beneath the response-time graph, *pmgoraping* shows CPU and Memory meters for the overall system, to help you judge whether problems might be caused by shortage of system resources. The default *ping* update time is every 10 seconds, but this can be changed with the **-t** option. Note that this update time is independent of the update interval for the *oraping* PMDA.

If you position the mouse pointer over the LED, click the Right mouse button, and choose Info, *pmgoraping* displays a window similar to that shown in Figure 5-4.



```
     0.036352  moomba:oraping.response_time
○  < 0.01
○ >= 0.01
● >= 0.125
○ >= 0.375
● >= 1
```

Figure 5-4 Explanatory pmgoraping Window

This shows the response time thresholds at which the LED changes state and color.

Appendix A

Abbreviated Installation Instructions

This chapter contains abbreviated installation instructions for system administrators who have experience with PCP for ORACLE. No information is presented here that is not presented in Chapter 2, "Installation Instructions."

The example in this appendix presumes you have a single ORACLE database to monitor. It also assumes that the *ORACLE_SID* is a valid namespace identifier (it must start with a letter, followed by zero or more letters, digits, or underscores).

PCP Collector System

On the database server:

1. Make sure the *pcp* and *pcp_ora.sw.collector* images are installed and *pmcd* is running.
2. Know the password for the ORACLE database's DBA *SYS* user, or be able to "connect internal" to ORACLE.
3. Switch user to root (superuser).
4. Set *ORACLE_HOME* and *ORACLE_SID* in the environment.
5. Enter these commands:

   ```
   # cd /var/pcp/pmdas/oracle
   # ./Install pcp/pcp
   ```

6. Follow the instructions printed by the *Install* command.

PCP Monitor System

On any system where you want to monitor ORACLE database metrics from the server where *pmdaoracle* was just installed, make sure the *pcp* and *pcp_ora.sw.monitor* images are installed.

Appendix B

Troubleshooting PCP for ORACLE

Before installing an ORACLE PMDA, please read this book or the *README* file in */var/pcp/pmdas/oracle*. To fix problems you might have to run the *Install* and *Remove* scripts in that directory, so it would be a good idea to review the documentation to refresh your memory before continuing.

There is one complete subtree of ORACLE metrics in the PCP namespace for each ORACLE database instance for which you have installed an ORACLE PMDA.

The log files for ORACLE PMDAs are */var/adm/pcplog/oracle-*.log,* where the star is replaced by the ORACLE instance name assigned to the PMDA at install time. Thus, the log file for the ORACLE PMDA instance named “xyz” is */var/adm/pcplog/oracle-xyz.log*. Take care to check the date and time in the log files to ensure that you are not using an old log file to diagnose problems.

For more information, see the pmdaoracle(1) and pmdaoraping(1) reference pages.

PMDA Cannot Connect to ORACLE

Symptom: The ORACLE PMDA’s log file contains this entry:

```
PMDA unable to connect to ORACLE
(invalid username/password; logon denied)
```

Cause: The ORACLE database user account has not been created for the PMDA or the user.

Resolution: When the script */var/pcp/pmdas/oracle/Install* was run to configure the PMDA, a file was created containing the SQL statements that allow the PMDA to connect to the database and access the performance data. The file is in the */var/pcp/pmdas/oracle* directory. For example, if the ORACLE database instance is called *xyz*, the file is named *setup-xyz.sql*.

Connect to the ORACLE database instance associated with the PMDA using *svrmgrl* or *sqldba*, and run the SQL commands in the file.

If the file creates an `ops$...` user, make sure that a corresponding UNIX user exists and that */var/pcp/pmdas/32/pmdaoracle-xyz* exists, is owned by that UNIX user, and has the **setuid** bit set. Become that UNIX user (using *su* or equivalent command), set the *ORACLE_HOME* and *ORACLE_SID* environment variables, and attempt a default login to the database:

```
sqlplus /
```

On the other hand, if the file created a normal ORACLE user with a password (for example, user *pcp* with password *meter*), log in to the database by typing this command:

```
sqlplus pcp/meter
```

In either case, ensure that you use the same *ORACLE_HOME* and *ORACLE_SID* as those specified for the PMDA in the log file. If either is wrong in the log file, alter them in the command line for that PMDA in the */etc/pmcd.conf* file.

Finally, send PMCD a reconfiguration request:

```
# killall -HUP pmcd
```

Check the log file for the PMDA again.

ORACLE Connection Errors

Symptom: The ORACLE PMDA's log has errors of the form:

```
Error connecting to ORACLE. ORACLE not available.
smsgsg: shmget() failed...
```

Cause: This can be caused by a number of things, ranging from ORACLE being unavailable or misconfigured to incorrect parameters being specified when using the *Install* script to configure the PMDA.

Resolution: Make sure that the ORACLE database instance that the problematic PMDA utilizes is available. Connect to it using the command:

```
sqlplus
```

and try to fetch some data using a database user other than that of the PMDA. A demonstration user such as *scott/tiger* is ideal here.

If that works, connect to the database using the same ORACLE user that the PMDA does. Before doing so, make absolutely certain that the values of *ORACLE_HOME* and *ORACLE_SID* that appear in the PMDA's log file are correct, then set both these variables in the environment. The ORACLE user and password for the PMDA appear on the command line for the PMDA in */etc/pmcd.conf* as the argument to the **-c** option. For example, if the flag **-c pcp/pcp** appears on the command line, enter this command:

```
sqlplus pcp/pcp
```

If there is no **-c** option, the PMDA is using an `ops$...` login name and */var/pcp/pmdas/pmdaoracle* will be **setuid**. In this case, become the user who owns the *pmdaoracle* binary (using *su* or equivalent command) and use a default login to *sqlplus*:

```
sqlplus /
```

Before doing this, remember to check that *ORACLE_HOME* and *ORACLE_SID* are correct. If *sqlplus* indicates that the login is not permitted because the username or password is invalid, see the section titled "PMDA Cannot Connect to ORACLE" on page 32.

If *sqlplus* refuses to let you in, there may be a problem with your ORACLE database configuration. In particular, ORACLE does not handle NFS-mounted *ORACLE_HOME* directories well, because write permission is required to update control and log files located in *$ORACLE_HOME/dbs*.

Once you are in *sqlplus*, enter this command:

```
describe v$sysstat
```

A description of the *v$sysstat* view should appear. If it does not, rerun the script to grant the PMDA access to the performance data from the database (see the section titled "PMDA Cannot Connect to ORACLE" on page 32).

Restart PMCD by becoming superuser or root and entering:

```
# /etc/init.d/pcp start
```

Inspect the log file for the PMDA. If it started successfully this time, you may need to alter your database startup routine. See the section "Cannot Find ORACLE Metrics" on page 34 for details.

Cannot Find ORACLE Metrics

Symptom: After a system is rebooted, or after the ORACLE database is shut down and then restarted, it is not possible to get any ORACLE metrics.

Cause: If a PMDA is unable to connect to its ORACLE database instance, it cannot retrieve any performance metrics but will try to connect to the database again later. If the PMDA has a connection to the database and the database is shut down, the PMDA does not notice until the next request for ORACLE metrics arrives. This is because a PMDA accesses the ORACLE database only when a request is made. Thus, if a PMDA is idle while its database is shut down and then brought back up again, the next request to it fails because it is trying to use an old connection to the database.

Resolution: The ORACLE PMDA periodically attempts to reconnect to the database in response to requests for metrics. The pmdaoracle(1) reference page describes how you can customize reconnection behavior.

PMDA Starts but Later Stops Running

Symptom: The ORACLE PMDA *pmdaoracle* is installed and started by *pmcd*, but later requests to retrieve ORACLE performance metrics produce errors such as these:

```
No PMCD agent for domain of request
```

or

```
No values available
```

Cause: Typically this means that the *pmdaoracle* agent has been terminated by *pmcd* when the agent did not respond quickly enough to a request for performance metrics. This can sometimes happen when either the system is very heavily loaded, or when the ORACLE DBMS is very busy (for example, during a checkpoint).

To verify that this is really the cause of the problem, scan the tail of the */var/adm/pcplog/pmcd.log* file, looking for lines of this form:

```
[Mon Nov 4 15:32:26] pmcd(5024) Warning: _pmRead: timeout (after
5.000 sec) on fd=26
```

```
Cleanup "ora_production" agent (dom 32): protocol failure for fd=26
```

In the PMDA's log file (*/var/adm/pcplog/oracle-*.log*) should be some messages indicating an orderly termination, such as these:

```
Log for /var/pcp/pmdas/oracle/32/pmdaoracle-xyz on moomba started...
Set ORACLE_HOME=/oracle/home
Set ORACLE_SID=xyz
Log finished Mon Nov 4 15:40:49 1996
```

Resolution: The short-term solution is to restart the ORACLE PMDA by sending *pmcd* a SIGHUP signal:

```
# killall -HUP pmcd
```

This tells *pmcd* to restart any deceased PMDAs, restarting *pmdaoracle* with a fresh connection to the database.

If the problem persists, consider changing the *pmcd* timeout to be longer than 5 seconds in */etc/config/pmcd.options*. For example, add this line to change the default reconnect timeout to 10 seconds:

```
-r 10
```

Index

Tell Us About This Manual

As a user of Silicon Graphics products, you can help us to better understand your needs and to improve the quality of our documentation.

Any information that you provide will be useful. Here is a list of suggested topics:

- General impression of the document
- Omission of material that you expected to find
- Technical errors
- Relevance of the material to the job you had to do
- Quality of the printing and binding

Please send the title and part number of the document with your comments. The part number for this document is 007-3005-002.

Thank you!

Three Ways to Reach Us

- To send your comments by **electronic mail**, use either of these addresses:
  - On the Internet: techpubs@sgi.com
  - For UUCP mail (through any backbone site): *[your_site]*!sgi!techpubs
- To **fax** your comments (or annotated copies of manual pages), use this fax number: 650-932-0801
- To send your comments by **traditional mail**, use this address:

  Technical Publications
  Silicon Graphics, Inc.
  2011 North Shoreline Boulevard, M/S 535
  Mountain View, California 94043-1389