HP-UX Workload Manager Toolkits User's Guide

Version A.01.08

Manufacturing Part Number: T1302-90028
May 2005

Publication history

Eight edition
May 2005
T1302-90028
HP-UX 11i v1, HP-UX 11i v2

Seventh edition
March 2004
T1302-90022
HP-UX 11.0, HP-UX 11i v1, HP-UX 11i v2

Sixth edition
March 2004
T1302-90019
HP-UX 11.0, HP-UX 11i v1, HP-UX 11i v2

Fifth edition
July 2003
T1302-90016
HP-UX 11i v2

Fourth edition
June 2003
T1302-90012
HP-UX 11.0, HP-UX 11i v1, HP-UX 11i v2

Third edition
February 2002
T1302-90007
HP-UX 11.0, HP-UX 11i

Second edition
October 2001
T1302-90004
HP-UX 11.0, HP-UX 11i

First edition
March 2001
T1302-90001
HP-UX 11.0, HP-UX 11i

Notice

The information contained in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Oracle is a registered trademark, and Oracle8 as well as Oracle9 are trademarks of Oracle Corporation.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

The SNMP Toolkit uses a library written by CMU:

Contents

1. HP-UX Workload Manager Overview

What is HP-UX Workload Manager? 1
What are these toolkits? 2
Where do I get WLMTK? 2
Feedback to the development team 3
Support and patch policies 3
Training 3

2. HP-UX WLM Oracle Database Toolkit: Providing Database Metrics to WLM

Supported installations 6
Unsupported combinations. 7
Required software 7
Audience for the ODBTK documentation 8
Why use Oracle database metrics with WLM? 8
Tools in WLM Oracle Database Toolkit (ODBTK) 9
What metrics are available? 10
Overview of how the metrics work with WLM 11
Where do I get ODBTK? 12
What comes with ODBTK? 12
How do I install ODBTK? 14
Giving ODBTK access to the database instances 14
Where can I find example files showing how to set up metrics? 16
How do I use the metrics? 17
Implementing metrics as a system administrator vs. as a DBA 21
How do I customize an example WLM configuration file? 24
How do I place instances in workload groups? 32
Enabling and disabling SLOs based on database metrics 33
Executing a query to get an SQL value 35
Executing a query to get response time. 36
Specifying an amount of CPU per metric (parametric) 38
Specifying a desired metric level (goal-based). 40
Command reference 43
wlmoradc 43
smooth 43

Contents

What about security issues? 44
Potential password visibility issue. 44
Potential password visibility/file tampering 44
Potential perl code issue in the configuration file 45
Potential SQL code issue using --sqlfile file. 45
Troubleshooting 46
Error and informational messages 46
Debugging suggestions 46
Common errors and their solutions 46
Related information 48

3. HP-UX WLM Pay Per Use Toolkit (PPUTK)

4. HP-UX WLM Apache Toolkit (ApacheTK)
Supported installations 54
Why use ApacheTK with WLM? 54
Tools in ApacheTK 55
Where do I get ApacheTK? 55
What comes with ApacheTK?. 56
How do I install ApacheTK?. 58
How do I use ApacheTK?. 58
Example WLM and Apache configurations 59
Command reference 60
time_url_fetch 60
wlm_watch.cgi 60
Related information 61

5. HP-UX WLM BEA WebLogic Server Toolkit (WebLogicTK)
Supported installations 63
Why use WLM with BEA WebLogic? 63
Tools in WebLogicTK 64
Where do I get WebLogicTK?. 65
What comes with WebLogicTK?. 65
How do I install WebLogicTK?. 67
How do I use WebLogicTK?. 68

Example WLM configurations and properties files 68
Command reference 71
wlmwlsdc 71
expsmooth 72
What about security issues? 72
Potential password visibility 72
posix_sh_setup_file issues 72
Related information 73

6. HP-UX WLM Duration Management Toolkit and HP-UX WLM Toolkit for Base SAS Software

Supported installations 76
Audience for the DMTK / SASTK documentation 76
Why use DMTK / SASTK with WLM? 77
Tools in DMTK / SASTK 80
Overview of how WLM's duration management works 81
Where do I get DMTK / SASTK? 84
What comes with DMTK / SASTK? 84
How do I install DMTK / SASTK? 86
Where can I find example files showing how to use DMTK / SASTK? 86
How do I use DMTK / SASTK? 87
Completing an application within a certain duration (duration management) 87
Instrumenting SAS jobs for better duration management. 100
Completing an urgent job as quickly as possible (express lanes). 101
Command and macro reference. 113
wlmdurdc. 113
hp_wlmtk_goals_report 114
Frequently asked questions. 115
I want a process to finish in five minutes. How do I set that up? 115
What's the shortest job that can be managed? 115
Troubleshooting 115
Related information 116

Contents

7. HP-UX WLM SNMP Toolkit

Supported installations 117
Audience for the SNMPTK documentation 117
Why use SNMPTK? 118
Tools in SNMPTK 118
Where do I get SNMPTK? 119
What comes with SNMPTK? 119
How do I install SNMPTK? 120
How do I use SNMPTK? 120
Example WLM configuration. 122
Command reference 124
snmpdc. 124
Related information 124

Contents

Contents

Tables

Table 2-1. ODBTK directories and files 12
Table 2-2. WLM example configuration files described 18
Table 2-3. wlmoradc example configuration files described 20
Table 4-1. ApacheTK directories and files 56
Table 5-1. WebLogicTK directories and files 65
Table 6-1. DMTK/SASTK directories and files 84
Table 7-1. PPUTK directories and files 119

Tables

Figures

Figure 2-1. Overview of how the database metrics are used 11
Figure 6-1. Unrelated applications without duration management 78
Figure 6-2. Unrelated applications with duration management 79
Figure 6-3. Overview of how applications can be managed 82
Figure 6-4. Overview of how SAS jobs can be managed 83

Figures

1 HP-UX Workload Manager Overview

This chapter provides an overview of the HP-UX Workload Manager product and its supporting toolkits.

What is HP-UX Workload Manager?

HP-UX Workload Manager, also known as WLM, is an automatic resource management tool used for goal-based workload management. A workload is a group of processes that are treated as a single unit for the purposes of resource management. For example, a database application that consists of multiple cooperating processes could be considered a workload.

HP-UX WLM provides automatic resource allocation and application performance management through the use of prioritized service-level objectives (SLOs). Multiple prioritized workloads can be managed on a single server based on their reported performance levels.

HP-UX WLM manages workloads as defined in a configuration file. You assign applications and users to workload groups. HP-UX WLM then manages each workload group's CPU, real memory, and disk bandwidth resources according to the current configuration. HP-UX WLM automatically allocates CPU resources in order to achieve the desired SLO. Real memory and disk bandwidth allocations are statically assigned in the configuration file.

HP-UX WLM automates many of the features of PRM (Process Resource Manager) and HP-UX Virtual Partitions (vPars), including automatic re-allocation of resources in response to SLO performance.

What are these toolkits?

The HP-UX Workload Manager Toolkits (WLMTK) are a collection of toolkits that simplify:

- Getting metrics on Oracle database instances into WLM
- Managing Apache-based workloads
- Managing WebLogic Server workloads
- Managing the duration of workloads

For information on these toolkits, see

- “HP-UX WLM Oracle Database Toolkit: Providing Database Metrics to WLM” on page 5
- “HP-UX WLM Apache Toolkit (ApacheTK)” on page 53
- “HP-UX WLM BEA WebLogic Server Toolkit (WebLogicTK)” on page 63
- “HP-UX WLM Duration Management Toolkit and HP-UX WLM Toolkit for Base SAS Software” on page 75

For more information on HP-UX WLM, see the *HP-UX Workload Manager User’s Guide* or the wlm(5) man page.

Where do I get WLMTK?

The WLM Toolkits are installed at /opt/wlm/toolkits/.

If WLMTK is not already installed on your system, you can download it free of charge from the web. It is available at http://www.hp.com/go/wlm.

Feedback to the development team

If you would like to comment on the current functionality or make suggestions for future releases, please send email to:

wlmfeedback@rsn.hp.com

Support and patch policies

The http://www.hp.com/go/wlm site provides information on WLMTK's support policy and patch policy. These policies indicate the time periods for which this version of WLMTK is supported and patched.

Training

HP offers a course in HP-UX resource management using WLM. For information, including a course outline, visit:

http://www.hp.com/education/courses/u5447s.html

2 HP-UX WLM Oracle Database Toolkit: Providing Database Metrics to WLM

The HP-UX Workload Manager Oracle® Database Toolkit is a collection of utilities that simplify getting metrics on Oracle database instances into HP-UX Workload Manager, also known as HP-UX WLM.

WLM enables you to place Oracle instances and other applications in their own WLM workload groups. With the instances and applications separated in this manner, WLM can then manage the performance of each instance and application through prioritized SLOs. These SLOs typically attempt to ensure:

- A consistent level of performance for the workload
- The needs of critical instances are met—even during peak demand
- Resources are allocated based on time of day, system events, or application and database metrics

For more information on WLM, see the *HP-UX Workload Manager User's Guide* or the wlm(5) man page.

The tools in the WLM Oracle Database Toolkit (ODBTK) make it easy to use metrics from Oracle database instances as goal values in service-level objectives. These metrics can also be used to enable or disable SLOs based on database conditions.

These tools were designed with the following goals in mind:

- Simplify getting the data to answer the question: “When does my Oracle instance need more CPU to meet business objectives?” Toward this end, the tools provide a simple method for getting the following metrics to WLM:
  - Wallclock time consumed by a piece of SQL code (the amount of time it takes for the SQL code to execute)
  - A single floating-point value from a piece of SQL code
- Not require modifications to the application or the database
- Be easy to use quickly by providing useful examples, instructions, and pointers for additional information

Supported installations

The tools discussed in this document work with:

- HP-UX WLM Version A.02.02 on HP-UX 11i v1 (B.11.11) and HP-UX 11i v2 (B.11.23)
- HP-UX WLM Version A.02.01.01 on HP-UX 11i v2 (B.11.23)
- HP-UX WLM Version A.02.01 on HP-UX 11i v1 (B.11.11)
- HP-UX WLM Version A.01.02 (which is the first version to include the `cpushares` keyword) on HP-UX 11i v1 (B.11.11)
- Oracle 8.0.x, Oracle 8.1.5, Oracle 8.1.6, Oracle 8.1.7, and Oracle 9.0.1

Unsupported combinations

Oracle8*i* includes Database Resource Manager. When used with WLM, Database Resource Manager can produce unexpected results in terms of resource allocation and user-access priorities—however, there are no data integrity issues. Nevertheless, it is recommended that you not use Database Resource Manager and HP-UX Workload Manager on the same system at the same time.

Required software

ODBTK uses /opt/perl/bin/perl.

If you already have perl version 4.0.1.8 (or newer) installed in a location other than /opt/perl/bin/ and you prefer to use your perl, either:

- Ensure there is a symbolic link from /opt/perl/bin/perl to your perl

or

- Invoke `wlmoradc` and `smooth` with your installed perl.

  The following examples illustrate this idea for perl installed in /usr/contrib/bin/ instead of /opt/perl/bin/, with the invocations on the command line:

```
# /usr/contrib/bin/perl /opt/wlm/lbin/coll/wlmoradc arguments
# /usr/contrib/bin/perl /opt/wlm/lbin/coll/smooth arguments
```

  The next examples perform the same functions as the above lines, but with the invocations inside the WLM configuration file using the `coll_argv` keyword:

```
coll_argv = /usr/contrib/bin/perl /opt/wlm/lbin/coll/wlmoradc arguments ;
coll_argv = /usr/contrib/bin/perl /opt/wlm/lbin/coll/smooth arguments ;
```

Use the `coll_stderr` keyword in your WLM configuration to catch perl incompatibilities, ODBTK errors, and errors in your data collectors.

Audience for the ODBTK documentation

This documentation is intended for database administrators and system administrators who are charged with consolidating or managing multiple applications, including database instances, onto a single server, while ensuring that performance levels are maintained.

Why use Oracle database metrics with WLM?

The key benefit of using Oracle database metrics with WLM is that you can use these metrics to manage the performance of your instances. You specify SLOs for the instances based on the metrics.

For example, with these metrics you can:

- Keep response times for your transactions below a given level by setting response-time SLOs
- Increase an instance's available CPU when a particular user connects to the instance
- Increase an instance's available CPU when more than *n* users are connected
- Increase an instance's available CPU when a particular job is active
- Give an instance *n* CPU shares for each process in the instance
- Give an instance *n* CPU shares for each user connection to the instance

For examples showing how to set up these types of SLOs, see the files with names ending in ".wlm" in the directory /opt/wlm/toolkits/oracle/config/.

Tools in WLM Oracle Database Toolkit (ODBTK)

This toolkit includes two tools:

`wlmoradc`

`wlmoradc` is a data collector for HP-UX Workload Manager and is designed to provide an easy building block for Oracle instance management with wlmd(1M).

It takes one or more SQL statements and uses the Oracle tool SQL*Plus to connect to an Oracle instance and execute the statements, returning either the raw value returned from the SQL or the elapsed execution time. The results are sent to stdout for human viewing, logging, or most often, for use by wlmd(1M) by means of the wlmrcvdc(1M) or wlmsend(1M) utilities.

`smooth`

NOTE

As of WLM A.02.02, it is recommended that you use the `cntl_smooth` WLM keyword instead of the `smooth` tool. For information on `cntl_smooth`, see the wlmconf(4) man page.

`smooth` takes a stream of newline-delimited numbers and outputs a stream of numbers that are a running average of the last *n* values, where *n* is a value that can be set on the command line. The principal use for the `smooth` utility is to remove short spikes or dips in data collector output used with WLM, but can be applied to any stream of floating-point numbers.

Although not part of ODBTK, the functionality provided by the `cpushares` keyword in the WLM configuration file fits quite nicely with the toolkit.

`cpushares` keyword in WLM configuration file

> The `cpushares` keyword allows shares-per-metric allocations. This enables you to create goal-based SLOs in WLM of the form “x CPU shares for each metric y.” (A CPU share is 1/100 of one CPU or of each CPU, depending on WLM’s mode of operation.) (This type of allocation can also be thought of as a parametric allocation. It is calculated directly from the metric and is thus a function of the metric. This type of function is a parametric equation.)

NOTE

The `cpushares` keyword is available starting with HP-UX WLM Version A.01.02.

What metrics are available?

The following types of database metrics are available:

- Time elapsed while a user-defined set of SQL code executes

  This provides a response-time measurement for the database
- Application-specific value returned by executed SQL code
- Information retrieved from Oracle V$ tables using SQL

  These tables provide dynamic performance data for Oracle instances and allow the Oracle DBA to see current performance information.

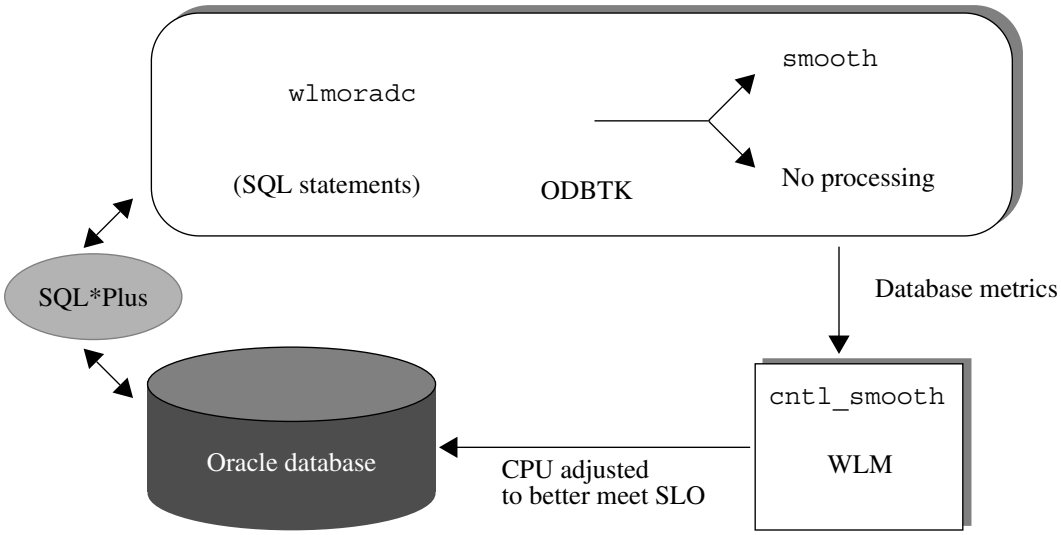
Overview of how the metrics work with WLM

Figure 2-1 illustrates how the ODBTK works with your databases to get metrics to WLM. First, the toolkit utility `wlmoradc` uses SQL statements to interact with the database through SQL*Plus to get the database metrics. Once `wlmoradc` has the resulting data, ODBTK may calculate a running average with the `smooth` toolkit utility or perform no action on the data. ODBTK then sends the metrics to WLM, which adjusts the CPU for the instance so that it better meets its SLOs.

NOTE As of WLM A.02.02, it is recommended that you use the `cntl_smooth` WLM keyword instead of the `smooth` tool. For information on `cntl_smooth`, see the wlmconf(4) man page.

This diagram ignores the fact that other database instances and applications may be on the system competing for resources. For a diagram showing the interaction of several applications, see the *HP-UX Workload Manager User's Guide*.

Figure 2-1 Overview of how the database metrics are used



Where do I get ODBTK?

ODBTK is installed at /opt/wlm/toolkits/oracle/.

If it is not already installed on your system, download it free of charge from the web. It is available at http://www.hp.com/go/wlm.

What comes with ODBTK?

ODBTK comes with two utilities (`wlmoradc` and `smooth`), their man pages, and a number of example configuration files. Table 2-1 lists many of the included files.

Table 2-1 ODBTK directories and files

Directory/file	Description
/opt/wlm/share/man/man5.Z/wlmtk.5	Man page that provides an overview of the HP-UX Workload Manager Toolkits
/opt/wlm/toolkits/oracle/	Directory for ODBTK and all its files
/opt/wlm/toolkits/doc/WLMTKug.pdf	WLMTK User's Guide (PDF)
/opt/wlm/toolkits/doc/WLMTKug.ps	WLMTK User's Guide (PostScript)
/opt/wlm/share/man/man1m.Z/wlmoradc.1m	Man page for the HP-UX WLM Oracle data collector
/opt/wlm/share/man/man1m.Z/smooth.1m	Man page for `smooth`
/opt/wlm/toolkits/README	Readme for all the WLM toolkits
/opt/wlm/toolkits/oracle/README	Readme for the HP-UX WLM Oracle Database Toolkit
/opt/wlm/toolkits/oracle/bin/wlmoradc	The HP-UX WLM Oracle data collector, which provides data to WLM

Table 2-1 ODBTK directories and files (Continued)

Directory/file	Description
/opt/wlm/toolkits/oracle/bin/smooth	Utility for computing a running average, which is useful for smoothing out extreme values in data
/opt/wlm/toolkits/oracle/config/alpha_shares_per_user.wlm	Example configuration file*
/opt/wlm/toolkits/oracle/config/batchuser_boost.wlm	Example configuration file*
/opt/wlm/toolkits/oracle/config/manual_payroll_boost.wlm	Example configuration file*
/opt/wlm/toolkits/oracle/config/process_cnt.oradc	Example configuration file*
/opt/wlm/toolkits/oracle/config/select_scott_resptime.oradc	Example configuration file*
/opt/wlm/toolkits/oracle/config/shares_per_process.wlm	Example configuration file*
/opt/wlm/toolkits/oracle/config/shares_per_user.wlm	Example configuration file*
/opt/wlm/toolkits/oracle/config/sys_table_resptime.oradc	Example configuration file*
/opt/wlm/toolkits/oracle/config/timed_select_scott.wlm	Example configuration file*
/opt/wlm/toolkits/oracle/config/timed_sys_table.wlm	Example configuration file*
/opt/wlm/toolkits/oracle/config/user_cnt.oradc	Example configuration file*
/opt/wlm/toolkits/oracle/config/user_cnt_boost.wlm	Example configuration file*
/opt/wlm/toolkits/oracle/install/create_hp_wlmuser.sql	SQL file for creating the default user and role that `wlmoradc` uses to connect to database instances
/opt/wlm/toolkits/oracle/install/drop_hp_wlmuser.sql	SQL file for removing the default user and role used by `wlmoradc`

* For descriptions of the example configuration files, see Table 2-2 on page 18 and Table 2-3 on page 20.

How do I install ODBTK?

Use the SD-UX `swinstall` command to install ODBTK, which is part of the WLM Toolkits product (product number T1302AA).

The toolkit is installed at /opt/wlm/toolkits/oracle/. The man pages are installed at /opt/wlm/share/man/.

For more information about installation procedures and related issues, refer to the following documentation:

- *Managing HP-UX Software with SD-UX*
- swinstall (1M) man page

Giving ODBTK access to the database instances

Before you use the ODBTK utility `wlmoradc`, you must provide it with access to the database instances. You do this in one of two ways:

- Using an existing username/password combination
- Using the default `wlmoradc` username/password combination hp_wlmuser/hp_wlmuser

ODBTK comes with a script that creates a user and role for hp_wlmuser, as explained in the next section.

After granting ODBTK access to your instances, see the section “How do I use the metrics?” on page 17 for information on using `wlmoradc` with WLM.

Creating a user and role for hp_wlmuser

The `wlmoradc` utility, by default, attempts to connect to database instances with the username/password combination hp_wlmuser/hp_wlmuser. For convenience, the DBA may wish to create a user and role for hp_wlmuser. Creating this user allows `wlmoradc` to connect and query the V$ tables for SLOs based on SQL values; however, hp_wlmuser is not allowed to create any tables. To create the user and role, connect to each instance with SQL*Plus to execute create_hp_wlmuser.sql, as shown below.

NOTE Be sure to run create_hp_wlmuser.sql against each instance for which you want to use hp_wlmuser, setting $ORACLE_HOME appropriately.

```
$ whoami
oracle
$ ORACLE_HOME=/oracle/app/oracle/product/8.1.5
$ ORACLE_SID=instance1
$ $ORACLE_HOME/bin/sqlplus system/manager

SQL*Plus: Release 8.1.5.0.0 - Production on Thu Feb 22 17:32:53 2001

(c) Copyright 1999 Oracle Corporation.  All rights reserved.

Connected to:
Oracle8i Enterprise Edition Release 8.1.5.0.0 - Production
With the Partitioning and Java options
PL/SQL Release 8.1.5.0.0 - Production

SQL> @ /opt/wlm/toolkits/oracle/install/create_hp_wlmuser.sql
DOC> * @(#) HP WLMTK A.01.01 (2001_10_05_15_05_23) hpux_11.00
DOC> *
DOC> * (c) Copyright 2000, Hewlett-Packard Company, all rights reserved.
DOC> *
DOC> * $RCSfile: create_hp_wlmuser.sql,v $
DOC> * $Date: 2001/03/23 18:43:21 $
DOC> * $Revision: 1.4 $
DOC> */

Role created.


Grant succeeded.


Grant succeeded.


User created.


Grant succeeded.
```

```
USERNAME                       ACCOUNT_STATUS                 CREATED
------------------------------ ------------------------------ ---------
HP_WLMUSER                     OPEN                           14-MAR-01

SQL> Disconnected from Oracle8i Enterprise Edition Release 8.1.5.0.0 -
Production
With the Partitioning and Java options
PL/SQL Release 8.1.5.0.0 - Production
```

Removing the user and role for hp_wlmuser

To remove the user and role for hp_wlmuser, use the script drop_hp_wlmuser.sql against each affected instance:

```
SQL> @ /opt/wlm/toolkits/oracle/install/drop_hp_wlmuser.sql
DOC> * @(#) HP WLMTK A.01.01 (2001_10_05_15_05_23) hpux_11.00
DOC> *
DOC> * (c) Copyright 2000, Hewlett-Packard Company, all rights reserved.
DOC> *
DOC> * $RCSfile: drop_hp_wlmuser.sql,v $
DOC> * $Date: 2001/03/23 18:43:21 $
DOC> * $Revision: 1.2 $
DOC> */

User dropped.


Role dropped.

SQL> Disconnected from Oracle8i Enterprise Edition Release 8.1.5.0.0 -
Production
With the Partitioning and Java options
PL/SQL Release 8.1.5.0.0 - Production
```

Where can I find example files showing how to set up metrics?

The directory /opt/wlm/toolkits/oracle/config/ is filled with example WLM configuration files as well as `wlmoradc` configuration files. Copy these files to a working directory and modify them to match your environment.

For an overview of the example WLM configuration files, see Table 2-2 on page 18.

For an overview of the example `wlmoradc` configuration files, see Table 2-3 on page 20.

How do I use the metrics?

To use the Oracle database metrics:

1. Determine a useful example WLM configuration file.
2. Customize the WLM configuration file.
3. (Optional) Customize an example `wlmoradc` configuration file.
4. Check the WLM configuration file's syntax with `wlmd -c`.
5. Activate the WLM configuration file with `wlmd -a`.

If none of the example WLM configuration files are useful to you, consider the following approach:

1. Define workload groups in a WLM configuration file for your Oracle instances.
2. Decide which metrics you want to collect.
3. Implement the SQL statements to retrieve the desired metrics. Place these statements in an `wlmoradc` configuration file or in the WLM configuration file as an option to the `wlmoradc` utility.
4. Specify `wlmoradc` in the WLM configuration file as a data collector for WLM.
5. Check the WLM configuration file's syntax with `wlmd -c`.
6. Activate the WLM configuration file with `wlmd -a`.

The following steps discuss the first approach, where you customize an example WLM configuration file.

Step 1. Determine a useful example WLM configuration file.

Example configurations are available in /opt/wlm/toolkits/oracle/config/. Choose a file that provides metrics matching the type you need. Table 2-2 describes the available example files.

Table 2-2 WLM example configuration files described

WLM configuration file	Purpose
alpha_shares_per_user.wlm*	Demonstrate how to customize the supplied file shares_per_user.wlm for a slightly different set of instances
batchuser_boost.wlm	Demonstrate a conditional allocation: Each instance has a fixed minimum allocation and gets a boost if a certain user connects to the instance
manual_payroll_boost.wlm	Demonstrate a conditional allocation, with the allocation enforced when a certain application becomes active; also, demonstrate how to feed “external” metrics to WLM using `wlmsend`
shares_per_process.wlm*	Demonstrate an allocation where each instance gets a certain number of CPU shares per process
shares_per_user.wlm*	Demonstrate an allocation where each instance gets a certain number of CPU shares per user connection
timed_select_scott.wlm	Demonstrate a response-time goal using a SCOTT/TIGER table
timed_sys_table.wlm	Demonstrate a response-time goal using V$ Oracle system tables
user_cnt_boost.wlm	Demonstrate a conditional allocation, with a new allocation enforced when more than a set number of users connect

* Requires WLM A.01.02 or later

Step 2. Customize the appropriate WLM configuration file.

a. Determine whether metrics will be collected internal or external to the configuration file. To decide this issue, see the section "Implementing metrics as a system administrator vs. as a DBA" on page 21.

b. Copy the WLM configuration files from the directory /opt/wlm/toolkits/oracle/config/ to the local directory. These files have a ".wlm" extension. In these files, you define SLOs, the metrics you want to track, and how to track them.

c. Identify which instances will be managed.

d. Customize the instance names and paths in the configuration file to your environment.

e. Create a workload group for each instance in the `prm` structure in the configuration file.

f. Create or modify the SLOs to better fit your goals. SLOs are specified in `slo` structures.

NOTE

If the system is going to have interactive users, you may want to reserve a portion of the CPU to ensure reasonable response times for these users. You can reserve this CPU by adding an SLO similar to the following in your configuration file:

```
slo keep_interactive_nice {
        pri = 1;
        cpushares = 10 total;
        entity = PRM group OTHERS;
}
```

This SLO reserves 10% of the system CPU for interactive users.

NOTE

For optimal performance, do not set the `wlm_interval` keyword in your WLM configuration to a value of less than 10 seconds when using the `wlmoradc` data collector.

---

g. Create an application record in the `prm` structure in the configuration file for each instance by modifying the existing application records. For example, consider this line from /opt/wlm/toolkits/oracle/config/manual_payroll_boost.wlm:

```
apps = Ora_grp_1:/oracle/app/oracle/product/8.1.5/bin/oracle "ora*instance1",
```

Here, you would change the `oracle` path to match the `$ORACLE_HOME` setting for the instance. Also, you would change "ora*instance1" to reflect the actual name of the instance. The processes from your instance would be placed in the workload `Ora_grp_1` unless you change your workload definitions.

For more information on placing instances in workload groups, see "How do I place instances in workload groups?" on page 32.

For an example illustrating how to customize a WLM configuration file, see the section "How do I customize an example WLM configuration file?" on page 24.

Step 3. (Optional) Customize an example `wlmoradc` configuration file.

Copy the `wlmoradc` configuration files from the /opt/wlm/toolkits/oracle/config/ directory and customize them as needed. This step is optional because you can place many of the items from the file on the command line; the file is mainly for convenience. These files have a ".oradc" extension and can be used to set database environment variables. Also, you can place SQL statements in these files. Table 2-3 describes the available files.

Table 2-3 `wlmoradc` example configuration files described

`wlmoradc` configuration file	Purpose
process_cnt.oradc	Report number of user processes using an instance
select_scott_resptime.oradc	Demonstrate a timed proxy transaction with the SCOTT/TIGER tables
sys_table_resptime.oradc	Demonstrate a timed proxy transaction against V$ Oracle system tables

Table 2-3 **`wlmoradc` example configuration files described (Continued)**

`wlmoradc` configuration file	Purpose
user_cnt.oradc	Report number of users connected to an instance

For information on the syntax of the `wlmoradc` configuration files, see the wlmoradc(1M) man page.

Use these configuration files from their current location via the `--configfile` option to `wlmoradc`.

For an example of how to use these files, see "Specifying an amount of CPU per metric (parametric)" on page 38.

Step 4. Check the WLM configuration file's syntax:

```
# wlmd -c configfile
```

Fix any errors that are found.

Step 5. Activate the WLM configuration file:

```
# wlmd -a configfile
```

Implementing metrics as a system administrator vs. as a DBA

ODBTK in combination with WLM provides a number of methods for implementing database metrics. The metrics can be collected through:

- Commands specified only in the WLM configuration file
- Commands external to the WLM configuration file

These methods are discussed below.

Commands specified only in the WLM configuration file

Specifying commands in the WLM configuration file is ideal for a DBA who also serves as a system administrator, as this method requires root privileges. With root privileges, the system administrator can conveniently edit the configuration file to collect the necessary metrics. In addition, this method requires the configuration file to be activated (with `wlmd -a`), which can only be done by a system administrator.

With this method, `wlmoradc`, is specified in the HP-UX WLM configuration file in a `coll_argv` line. Consider the following example from /opt/wlm/toolkits/oracle/config/shares_per_user.wlm:

```
tune oracle.instance1.user_cnt {
    coll_argv =
        wlmrcvdc
          wlmoradc
            --configfile /opt/wlm/toolkits/oracle/config/user_cnt.oradc
            --home /oracle/app/oracle/product/8.1.5
            --instance instance1
    ;
}
```

Here, the system administrator sets the version of Oracle to use and the instance to monitor. Any environment variables required by Oracle are set in the WLM configuration file. The administrator also chooses the metric to collect by using the `wlmoradc` configuration file user_cnt.oradc.

For more examples, see the files ending in ".wlm" in the directory /opt/wlm/toolkits/oracle/config/.

Commands external to the WLM configuration file

For environments with separate people in the system administrator and DBA roles, it is possible to have the system administrator set up the WLM configuration file with the data collector commands outside the file. This allows the DBA to easily change the metrics being collected without having to alter the configuration file. To accomplish this, the administrator sets up a `coll_argv` line to monitor a given metric using `wlmrcvdc`.

For example, from /opt/wlm/toolkits/oracle/config/manual_payroll_boost.wlm, the metric oracle.instance1.payroll_running is monitored because of the following `tune` structure:

```
tune oracle.instance1.payroll_running {
    coll_argv = wlmrcvdc ;
}
```

This structure sets up a mechanism using `wlmrcvdc` that takes metric values and feeds them into HP-UX WLM. These values are provided by the DBA using `wlmsend` in one of the manners described below. For information on these utilities, see the wlmrcvdc(1M) and wlmsend(1M) man pages.

The DBA provides metric values to `wlmrcvdc` in one of the following manners:

- Once, with a hard-coded value:

  Using `wlmsend`, send the value 1 to WLM to indicate the payroll application is running:

  ```
  # wlmsend oracle.instance1.payroll_running 1
  ```

  Similarly, use `wlmsend` to send the value 0 to indicate the payroll application is not running:

  ```
  # wlmsend oracle.instance1.payroll_running 0
  ```

- Continuous, with values from `wlmoradc`:

  Here, `wlmoradc` uses a configuration file and reports values every 60 seconds. These values are piped to `wlmsend`, which in turn, feeds the values to WLM:

  ```
  # wlmoradc --configfile file --interval 60 | wlmsend \
  oracle.instance1.payroll_running &
  ```

- Once, with a value from `wlmoradc`:

  Here, `wlmoradc` uses a configuration file again, but reports a value only once:

  ```
  # wlmsend oracle.instance1.payroll_running \
  `wlmoradc --configfile file --single`
  ```

You can set up these commands to be triggered:

- By `cron` on a regular interval
- Manually when the SLO needs to be met
- From inside SQL or via the `HOST` or '`!`' commands

How do I customize an example WLM configuration file?

To illustrate how to customize a WLM configuration file, consider the file /opt/wlm/toolkits/oracle/config/shares_per_user.wlm:

```
#
# Name:
#       shares_per_user.wlm
#
# Version information:
#
#       (C) Copyright 2000-2005 Hewlett-Packard Development Company, L.P.
#
#       $Revision: 1.35 $
#
# Caveats:
#       DO NOT MODIFY this script in its /opt/wlm/toolkits location!
#       Make modifications to a copy and place that copy outside the
#       /opt/wlm/ directory, as items below /opt/wlm will be replaced
#       or modified by future HP-UX WLM product updates.
#
# Purpose:
#       Demonstrate a parametric allocation: Each Oracle instance gets
#       three CPU shares per user connection. Also see the
#       shares_per_process.wlm example.
#
# Components:
#       Uses the user_cnt.oradc file with wlmoradc to count user connections.
#
# Dependencies:
#       This example was designed to run with HP-UX WLM version A.02.01
#       or later.  It uses the coll_stderr keyword introduced in A.02.01
#       and is, consequently, incompatible with earlier versions of
#       HP-UX WLM.
#
```

```
# prm structure
#       Define three workload groups.  The first two groups are for
#       Oracle instances.  The third group, called long_batch, is for
#       other work being done on the server.
#
#       The 'apps' definitions place the applications in workload
#       groups.  Note that the two Oracle instances each have a
#       Pro*C-based application:  /workload1/oltp1 and /workload2/oltp2.
#       These applications run in the same workload groups as their
#       respective instances.
#
#       Because Oracle processes rename themselves, we have HP-UX WLM
#       place processes in workload groups based on their alternate
#       names.  In this case, those alternate names will be
#       "ora*instance1" and "ora*instance2".
#
#       See wlmconf(4) for complete HP-UX WLM configuration information.
#
prm {
        groups = OTHERS : 1,
                 ora1_grp : 2,
                 ora2_grp : 3,
                 long_batch : 4;

        apps = ora1_grp:/oracle/app/oracle/product/8.1.5/bin/oracle
               "ora*instance1",
               ora1_grp:/workload1/oltp1,
               ora2_grp:/oracle/app/oracle/product/8.1.5/bin/oracle
               "ora*instance2",
               ora2_grp:/workload2/oltp2,
               long_batch:/oracle/demo/batch/long_batch;
}

#
# Have HP-UX WLM give workload ora1_grp three CPU shares for each user
# connected to the associated Oracle instance. See the tune structure below
# for details on how the user counts are collected.
#
slo ora1_slo {
        pri = 1;
        mincpu = 5;
        maxcpu = 90;
        entity = PRM group ora1_grp;
        cpushares = 3 total per metric oracle.instance1.user_cnt;
}
```

```
# Have HP-UX WLM give workload ora2_grp three CPU shares for each user
# connected to the associated Oracle instance. See the tune structure below
# for details on how the user counts are collected.
#
slo ora2_slo {
        pri = 2;
        mincpu = 5;
        maxcpu = 90;
        entity = PRM group ora2_grp;
        cpushares = 3 total per metric oracle.instance2.user_cnt;
}


#
# Have HP-UX WLM give workload long_batch a fixed allocation of 20 CPU
# shares. The priority of the workload is set very low, thus giving
# preference to the Oracle workloads over the long_batch jobs.
#
slo long_batch {
        pri = 10;
        cpushares = 20 total;
        entity = PRM group long_batch;
}

#
# Any CPU that remains after satisfying the above SLOs is given to the
# OTHERS group by default. You can change this default using the
# distribute_excess keyword. For more information on this keyword, see
# the wlmconf(4) man page.
#

#
# Set the basic interval on which wlmd allocation changes
# are made to 30 seconds. For optimal performance with Oracle, do not use
# an interval of less than 10 seconds.
#
tune {
        wlm_interval = 30;

        #
        # Collect any datacol stderr and send it to syslog.
        # This is handy for catching errors a missing
        # /opt/perl/bin/perl when using wlmoradc.
        #
        coll_stderr = syslog;
}
```

```
# Determine the number of number of connected users for instance1 and
# pass the metric on to wlmd.
#
tune oracle.instance1.user_cnt {
    coll_argv =
        wlmrcvdc
          wlmoradc
            --configfile /opt/wlm/toolkits/oracle/config/user_cnt.oradc
            --home /oracle/app/oracle/product/8.1.5
            --instance instance1
    ;
}
#
# Determine the number of number of connected users for instance2 and
# pass the metric on to wlmd.
#
tune oracle.instance2.user_cnt {
    coll_argv =
        wlmrcvdc
          wlmoradc
            --configfile /opt/wlm/toolkits/oracle/config/user_cnt.oradc
            --home /oracle/app/oracle/product/8.1.5
            --instance instance2
    ;
}
```

Next, we present the previous example modified. The modified file, /opt/wlm/toolkits/oracle/config/alpha_shares_per_user.wlm, has been altered as follows:

- There are three managed instances: alpha, beta, and gamma. Each instance gets a workload group: `alpha_grp`, `beta_grp`, and `gamma_grp`.
- There are three less important instances that are not actively managed: larry, moe, and curly. These instances share the `static_instances` workload group.
- `$ORACLE_HOME` for these instances does not match the `$ORACLE_HOME` in the above example file. As a result, the application record lines in the modified file were changed.
- Because there are more instances, the minimum allocations were changed slightly.
- Because there is no batch work done on the machine, the `long_batch` workload group was removed.

- A Pro*C-based application, zeus_app, is used with instance beta, so it has an application record that places it in the workload group beta_grp.

```
#
# Name:
#       alpha_shares_per_user.wlm
#
#
# Version information:
#
#       (C) Copyright 2000-2005, Hewlett-Packard Development Company, L.P.
#
#       $Revision: 1.21 $
#
# Caveats:
#       DO NOT MODIFY this script in its /opt/wlm/toolkits location!
#       Make modifications to a copy and place that copy outside the
#       /opt/wlm/ directory, as items below /opt/wlm will be replaced
#       or modified by future HP-UX WLM product updates.
#
# Purpose:
#       Example configuration file for fictional Oracle instances and
#       applications for the HP-UX Workload Manager Oracle Database Toolkit
#       User's Guide.  This configuration demonstrates how to copy and
#       modify the shares_per_user.wlm configuration for a set of Oracle
#       instances. See the User's Guide for an explanation of the process.
#
#       Independently control CPU allocation by number of user
#       connections for three database instances, alpha, beta, gamma.
#       Set aside remaining CPU resources for the common workload group
#       hosting the database instances larry, moe, and curly.
#
# Components:
#       Uses the user_cnt.oradc file with wlmoradc to count user connections.
#
# Dependencies:
#       This example was designed to run with HP-UX WLM version A.02.01
#       or later.  It uses the coll_stderr keyword introduced in A.02.01
#       and is, consequently, incompatible with earlier versions of
#       HP-UX WLM.
#
```

```
# prm structure
#       Create workload groups and designate which binaries and Oracle
#       instances will be placed in each. Alpha, beta, and gamma all
#       get their own group, while larry, moe, and curly are placed in
#       the group named static_instances.
#
#       The alpha, beta, and gamma instances are all Oracle 8.1.5 based,
#       and share an ORACLE_HOME setting, while larry, moe, and curly
#       are 8.0.6 based.
#
#       Because Oracle processes rename themselves (as in ora_dbw1_alpha
#       or ora_pmon_alpha), we also tell HP-UX WLM to watch for the
#       applications under the alternate names "ora*alpha", "ora*beta",
#       etc.
#
#       The application zeus_app runs with the instance beta, so it is
#       placed in the beta_grp workload group.
#
#       See wlmconf(4) for complete HP-UX WLM configuration information.
#
prm {
        groups = OTHERS : 1,
                 alpha_grp : 2,
                 beta_grp : 3,
                 gamma_grp : 4,
                 static_instances : 5;

        apps = alpha_grp:/u01/app/oracle/product/8.1.5/bin/oracle
               "ora*alpha",

               beta_grp:/u01/app/oracle/product/8.1.5/bin/oracle
               "ora*beta",
               beta_grp:/u01/app/olympus_bin/zeus_app,

               gamma_grp:/u01/app/oracle/product/8.1.5/bin/oracle
               "ora*gamma",

               static_instances:/u01/app/oracle/product/8.0.6/bin/oracle
               "ora*larry",
               static_instances:/u01/app/oracle/product/8.0.6/bin/oracle
               "ora*moe",
               static_instances:/u01/app/oracle/product/8.0.6/bin/oracle
               "ora*curly"
               ;

}
```

```
# Have HP-UX WLM give alpha_grp three CPU shares for each user.
#
slo alpha_slo {
        pri = 1;
        mincpu = 20;
        maxcpu = 70;
        entity = PRM group alpha_grp;
        cpushares = 3 total per metric oracle.alpha.user_cnt;
}

#
# Have HP-UX WLM give beta_grp three CPU shares for each user.
#
slo beta_slo {
        pri = 1;
        mincpu = 20;
        maxcpu = 70;
        entity = PRM group beta_grp;
        cpushares = 3 total per metric oracle.beta.user_cnt;
}

#
# Have HP-UX WLM give gamma_grp three CPU shares for each user.
#
slo gamma_slo {
        pri = 1;
        mincpu = 20;
        maxcpu = 70;
        entity = PRM group gamma_grp;
        cpushares = 3 total per metric oracle.gamma.user_cnt;
}

#
# Have HP-UX WLM give workload static_instances a fixed allocation of 30
# CPU shares. Instances larry, moe, and curly will collectively use this
# allocation. The priority of the workload is set very low, thus giving
# preference to the Oracle workloads over the long_batch jobs.
#
slo static_instances_fixed {
        pri = 10;
        cpushares = 30 total;
        entity = PRM group static_instances;
}
```

```
# Any CPU that remains after satisfying the above SLOs is given to the
# OTHERS group by default. You can change this default using the
# distribute_excess keyword. For more information on this keyword, see
# the wlmconf(4) man page.
#

#
# Set the basic interval on which wlmd allocation changes
# are made to 30 seconds. For optimal performance with Oracle, do not use
# an interval of less than 10 seconds.
#
#
tune {
        wlm_interval = 30;

        #
        # Collect any datacol stderr and send it to syslog.
        # This is handy for catching errors a missing
        # /opt/perl/bin/perl when using wlmoradc.
        #
        coll_stderr = syslog;
}

#
# Have wlmd start a copy of wlmoradc to collect the number of
# users connected to instance alpha.
#
tune oracle.alpha.user_cnt {
    coll_argv =
        wlmrcvdc
          wlmoradc
            --configfile /opt/wlm/toolkits/oracle/config/user_cnt.oradc
            --home /oracle/app/oracle/product/8.1.5
            --instance alpha
    ;
}
```

```
# Duplicate the setup for beta -- count beta instance user connections
# and send to wlmd via wlmrcvdc.
#
tune oracle.beta.user_cnt {
    coll_argv =
        wlmrcvdc
          wlmoradc
            --configfile /opt/wlm/toolkits/oracle/config/user_cnt.oradc
            --home /u01/app/oracle/product/8.1.5
            --instance beta
    ;
}

#
# Duplicate the setup for gamma -- count gamma instance user connections
# and send to wlmd via wlmrcvdc.
#
tune oracle.gamma.user_cnt {
    coll_argv =
        wlmrcvdc
          wlmoradc
            --configfile /opt/wlm/toolkits/oracle/config/user_cnt.oradc
            --home /u01/app/oracle/product/8.1.5
            --instance gamma
    ;
}
```

For more information on the WLM configuration file syntax, see wlmconf(4).

How do I place instances in workload groups?

When you start an Oracle instance, the process names have the instance name appended. For example, the pmon process for a database instance Inventory would appear in a process list as `ora_pmon_Inventory`.

WLM allows you to place all the processes for a given instance in a workload group of their own. WLM can then manage SLOs for the individual instances.

In the WLM configuration file, instances are separated using the `apps` keyword. In the example below, from /opt/wlm/toolkits/oracle/config/batchuser_boost.wlm, there are four workload groups. Processes for the instance named instance1, which are presumably of the form `ora*instance1`, are placed in the workload group `Ora_grp_1`. Similarly, instance2 processes are placed in `Ora_grp_2`.

```
prm {
        groups = OTHERS : 1,
                 Ora_grp_1 : 2,
                 Ora_grp_2 : 3,
                 long_batch : 4;

        apps = Ora_grp_1:/oracle/app/oracle/product/8.1.5/bin/oracle
                "ora*instance1",
                Ora_grp_1:/workload1/oltp1,
                Ora_grp_2:/oracle/app/oracle/product/8.1.5/bin/oracle
                "ora*instance2",
                Ora_grp_2:/workload2/oltp2,
                long_batch:/oracle/demo/batch/long_batch;
}
```

Enabling and disabling SLOs based on database metrics

This section explains how to enable and disable SLOs based on number of processes, number of connections, queue length, and other such metrics.

WLM provides a mechanism to cause an SLO to be active or inactive for a given period of time. This feature is accessed through the `condition` and `exception` keywords, as explained in the wlmconf(4) man page. You can use time of day, day of week, and so forth to drive a condition. Also, external metric values can drive the conditions.

For example, an SLO can be built to control the allocation for a particular application. However, there is no point to give the application any allocation if it is not running. The GlancePlus toolkit's APP_ACTIVE_PROC metric, explained in the glance_app(1M) man page, could be used as a condition to the application's SLO so that it is only active when glance Adviser sees an active process in the application's group.

The `wlmoradc` utility can be used in a similar fashion to enable or disable SLOs based on values available from inside an Oracle instance. Items from the instance V$ tables are particularly useful for this purpose, as they provide administrative information to the DBA in much the same way that GlancePlus or OpenView metrics provide information to an HP-UX system administrator. For more information on the V$ tables, see your Oracle documentation.

For instance, if we'd like WLM to give 20 shares to instance 'blue_suede_shoes' whenever we see the Oracle user ELVIS connected to the database instance, a SQL statement like

```
select count(*) from V$SESSION where username = 'ELVIS';
```

would count the matching connections, and an SLO/metric combination like

```
slo blue_slo {
   pri = 1;
   cpushares = 20 total;
   entity = PRM group blue_group;
   condition = metric elvis_user_cnt > 0;
}

tune elvis_user_cnt {
   coll_argv =
      wlmrcvdc
        wlmoradc
          --sqlstring "select count(*) from V$SESSION where username = 'ELVIS';"
   ;
}
```

would manage the allocation.

Example files in the toolkit with `conditions` or `exceptions` are:

```
/opt/wlm/toolkits/oracle/config/batchuser_boost.wlm
/opt/wlm/toolkits/oracle/config/manual_payroll_boost.wlm
/opt/wlm/toolkits/oracle/config/user_cnt_boost.wlm
```

Executing a query to get an SQL value

In this section, the configuration has an SLO that increases the workload's CPU as the number of active connections (or users) increases. To implement this configuration:

Step 1. Copy the shares_per_user.wlm file to your local directory and rename it:

```
# cp /opt/wlm/toolkits/oracle/config/shares_per_user.wlm \
./my_shares_per_user.wlm
```

Step 2. Edit the file my_shares_per_user.wlm:

a. To be specific to your instance by changing the instance strings

b. So that the oracle paths match your `$ORACLE_HOME`

Step 3. Check the syntax of your my_shares_per_user.wlm file:

```
# wlmd -c my_shares_per_user.wlm
```

Step 4. Fix any errors with the file. If there are no errors, activate the configuration:

```
# wlmd -a my_shares_per_user.wlm
```

Step 5. Use `wlminfo` to view CPU allocations by workload group. Invoke the command as follows to show allocations with live updates:

```
# /opt/wlm/bin/wlminfo group -l
```

Keep the `wlminfo` output visible.

Step 6. Connect to your instance with SQL*Plus and watch the allocation for the instance's workload group grow as the number of active connections increases.

NOTE If you have any difficulties, use the `tail` command to look at the end of the file /var/adm/syslog/syslog.log. Look for warnings and errors originating from the WLMTK facility.

Step 7. Once you have a feel for how the components work together, modify the my_shares_per_user.wlm file to perform the SQL queries needed in your environment.

For additional example configuration files that you can modify to better fit your needs, see the files ending in ".wlm" in the directory /opt/wlm/toolkits/oracle/config/.

Executing a query to get response time

The configuration in this section executes a timed query. The CPU allocation for the instance's workload group is increased to keep the response time under 2.5 seconds. Note that this goal value would change based on the hardware, software, and parameters.

To implement this configuration:

Step 1. Install the SCOTT/TIGER examples that come with Oracle databases.

Step 2. Copy the timed_select_scott.wlm file to your local directory and rename it:

```
# cp /opt/wlm/toolkits/oracle/config/timed_select_scott.wlm \
./my_timed_select_scott.wlm
```

Step 3. Edit the my_timed_select_scott.wlm file:

a. To be specific to your instance by changing the instance strings

b. So that the oracle paths match your `$ORACLE_HOME`

Step 4. Check the syntax of your my_timed_select_scott.wlm file:

```
# wlmd -c my_timed_select_scott.wlm
```

Step 5. Fix any errors with the file. If there are no errors, activate the configuration:

```
# wlmd -a my_timed_select_scott.wlm
```

For information on additional example configuration files that you can modify to better fit your needs, see Table 2-2 on page 18 and Table 2-3 on page 20.

To implement your own time-based goals from SQL statements:

Step 1. Identify SQL statements that perform the transaction to be timed.

Step 2. Try the SQL statements manually using SQL*Plus.

NOTE

Oracle is very efficient at caching data. When you are getting a walltime by hand for a baseline, run the query twice using the option `--iterations=2` to `wlmoradc`. The first iteration encaches the data, and the second iteration represents the query with the data encached, providing a more realistic execution time.

Whether data comes from a disk or cache memory is not important. WLM and ODBTK are only concerned with the walltime needed to execute the SQL statements. Either way, if your data stays encached or if the system is so busy that data is quickly moved out of cache memory, your walltimes represent the typical times for queries on the system.

Step 3. Place the working SQL, along with any desired `wlmoradc` options, in an `wlmoradc` configuration file, say times.oradc.

Step 4. Invoke `wlmoradc` on the command line:

```
# wlmoradc --configfile times.oradc
```

If there are problems, add the option `--debug 2` to see the SQL input and output and determine the causes:

```
# wlmoradc --configfile times.oradc --debug 2
```

Step 5. Invoke `wlmoradc` from a WLM configuration file

a. Specify your SLO.

Here, the goal is to keep the transaction time under 300, with the time units determined by the data collector used.

```
slo Ora_1_slo {
        pri = 1;
        mincpu = 15;
        maxcpu = 75
        goal = metric O1_transaction01_time < 300;
        entity = PRM group Ora_grp_1;
}
```

b. Invoke `wlmoradc`.

In the `tune` structure below, `wlmoradc` uses the times.oradc file to track the time for the transaction for the instance named instance1. The result is forwarded to HP-UX WLM by `wlmrcvdc`.

```
tune O1_transaction01_time {
    coll_argv = wlmrcvdc
                wlmoradc
                  --configfile times.oradc
                  --instance instance1
    ;
}
```

Step 6. Check the WLM configuration file's syntax:

```
# wlmd -c configfile
```

Fix any errors that are found.

Step 7. Activate the WLM configuration file:

```
# wlmd -a configfile
```

Specifying an amount of CPU per metric (parametric)

You can specify an SLO for an instance's workload group that requests an allocation that varies in direct proportion to a given metric. Such an SLO must make use of the `cpushares` keyword in the WLM configuration file. Given a metric, `cpushares` determines how many CPU shares that each unit of the metric is currently allocated. You then create a goal to maintain a shares-per-metric level that provides the desired performance.

To implement this type of SLO:

Step 1. Identify what database condition is interesting, for example: number of users, number of processes, and so forth.

Step 2. Identify the SQL statements that capture that information.

Step 3. Confirm the SQL statements using SQL*Plus by hand.

Step 4. Place the working SQL in an `wlmoradc` configuration file (copy the file /opt/wlm/toolkits/oracle/config/user_cnt.oradc to a local filename, say my_user_cnt.oradc, and change the SQL section).

Step 5. Invoke `wlmoradc` on the command line:

```
# wlmoradc --configfile my_user_cnt.oradc
```

If there are problems, add the option `--debug 2` to see the SQL input and output and determine the causes:

```
# wlmoradc --configfile my_user_cnt.oradc --debug 2
```

Step 6. In your WLM configuration file:

a. Specify your SLO, as shown in this shares-per-process example. Here, the goal is for each process to have at least three CPU shares:

```
slo Ora_1_slo {
       pri = 1;
       mincpu = 5;
       maxcpu = 90;
       entity = PRM group Ora_grp_1;
       cpushares = 3 total per metric oracle.instance1.proc_cnt;

}
```

b. Invoke `wlmoradc`.

In the `tune` structure below, `wlmoradc` uses the process_cnt.oradc file to determine the number of processes for the instance named instance1. The number of processes is then forwarded to WLM by `wlmrcvdc`. HP-UX WLM uses `cpushares` to divide the number of CPU shares for workload group `ora_grp_1` by the number of processes.

```
tune oracle.instance1.proc_cnt {
   coll_argv =
       wlmrcvdc
         wlmoradc
           --configfile
             /opt/wlm/toolkits/oracle/config/process_cnt.oradc
           --home /oracle/app/oracle/product/8.1.5
           --instance instance1
           --interval 30
   ;
}
```

Step 7. Check the WLM configuration file's syntax:

```
# wlmd -c configfile
```

Fix any errors that are found.

Step 8. Activate the WLM configuration file:

```
# wlmd -a configfile
```

Specifying a desired metric level (goal-based)

Rather than worrying about how much CPU an instance gets per metric, you can specify a desired goal for the metric. Using a `goal` statement in the WLM configuration, you indicate the current value of the metric (obtained from `wlmoradc`) and your desired goal for the metric. The following example, from /opt/wlm/toolkits/oracle/config/timed_select_scott.wlm, shows a `goal` statement, with the goal for the metric value to be less than 2.5:

```
slo Ora_1_slo {
        pri = 1;
        mincpu = 20;
        maxcpu = 80;
        entity = PRM group Ora_grp_1;
        goal = metric oracle.instance1.sel_scott_resptime < 2.5;
}
```

The metric in the `goal` statement, `oracle.instance1.sel_scott_resptime`, is kept current by using `wlmoradc` in its `tune` structure, in the same configuration file:

```
tune oracle.instance1.sel_scott_resptime {
    coll_argv =
        wlmrcvdc
          wlmoradc
            --configfile
              /opt/wlm/toolkits/oracle/config/select_scott_resptime.oradc
            --home /oracle/app/oracle/product/8.1.5
            --instance instance1
    ;
    cntl_kp=3;
}
```

HP-UX WLM then automatically adjusts the CPU for the instance's associated workload group, in an attempt to meet the goal value. Generally speaking, the goal could be time-driven or a threshold for the number of transactions, or any number of other types of goals.

Time-driven goals

An important element of goal-based workload management is a time-driven goal: WLM determines the correct CPU allocation so that some action is performed in *n* seconds. This action could be a transaction for an instance doing OLTP, a query that is run frequently, or any unit of work with a completion time that can be used as a proxy to gauge the overall performance of the instance.

ODBTK helps you create time-driven goals through the `wlmoradc` command, which can measure the walltime consumed by SQL code. To form such goals:

Step 1. Choose representative SQL code to form a “proxy transaction” or “proxy unit of work”.

For information on fine-tuning your SQL statements with the SQL trace and TKPROF utilities, see the manual *Oracle 8i Tuning*.

Step 2. Follow the steps in “Executing a query to get response time” on page 36 to establish a baseline measurement of the time needed for the transaction.

Step 3. Create a WLM goal that is based either on the baseline or a Service-Level Agreement. (In some cases, there may be a Service-Level Agreement that the proxy transaction take less than Y seconds.)

Successful proxy transactions typically have the following characteristics:

- The proxy transaction is sensitive to CPU allocation changes.

  Remember that WLM modifies the CPU allocation to adjust the performance of the Oracle instance. Thus, a transaction that is mainly disk or memory work does not help WLM understand the instance’s need for more or less CPU.

- The Proxy transaction SQL does not have side effects on the instance.

  The SQL will be repeatedly executed, so the proxy should not include statements—such as UPDATE, DELETE, or INSERT—that affect the integrity of the data.

- The proxy transaction is of significant duration.

  The `wlmoradc` tool uses an SQL*Plus timer that has, at most, resolution of hundredths of seconds. However, proxy transactions of less than approximately 0.2 seconds may experience instability, especially if the overall volume of transactions is low. This is a guideline, not a hard limit, but take time to establish good baseline measurements if the length of the proxy transaction is very short. Because of the length of UNIX timeslices and measurement errors for very short duration events, also consider using the `cntl_smooth` keyword in your WLM configuration to add stability to short duration measurements.

- The proxy transaction should be a small percentage of the total instance work.

  Be careful that the proxy transaction does not consume too much time. A CPU-sensitive transaction must be consuming CPU resources, so a proxy transaction should be selected that does not consume too large a percentage of the instance's resources. You do not want your measurement of the performance to negatively affect the instance's performance.

  For example, if a given proxy 'SELECT X from table Y' is used every 30 seconds to gauge an instance's performance and 10,000 other similar queries are done during that period, the proxy's resource use is probably a very small percentage of the overall work. However, for a system where table Y is extremely large and only ten transactions are completed in the 30 seconds, using the proxy may be too great a percentage of the work. In this case, you should consider performing some smaller SELECT or executing the proxy less frequently than 30 seconds.

Command reference

ODBTK consists of two utilities:

- `wlmoradc`
- `smooth`

These utilities are described below.

wlmoradc

`wlmoradc` is a data collector for HP-UX Workload Manager. It provides an easy building block for Oracle instance management with wlmd(1M).

It takes one or more SQL statements and uses the Oracle tool SQL*Plus to connect to an Oracle instance and execute the statements, returning either the raw value returned from the SQL or the elapsed execution time. The results are sent to stdout for human viewing, logging, or most often, for use by wlmd(1M) by means of the wlmrcvdc(1M) or wlmsend(1M) utilities.

Several items are needed for `wlmoradc` to function. These items are:

- Oracle home
- Instance name
- username/password
- SQL code

For more information on this utility, see the wlmoradc(1M) man page.

smooth

NOTE As of WLM A.02.02, it is recommended that you use the `cntl_smooth` WLM keyword instead of the `smooth` tool. For information on `cntl_smooth`, see the wlmconf(4) man page.

`smooth` takes a stream of newline-delimited numbers and outputs a stream of numbers that are a running average of the last *n* values, where *n* is a value that can be set on the command line. The principal use for the smooth utility is to remove short spikes or dips in data collector output used with WLM, but it can be used with any stream of floating-point numbers.

For more information on this utility, see the smooth(1M) man page.

What about security issues?

Be aware of the following security issues so that you can better protect your data.

Potential password visibility issue

With the `wlmoradc` utility, if `--username` and `--passwd` are used on the command line so that `wlmoradc` can connect to an instance, the username and password are visible via the UNIX `ps` command.

If this is a problem, place the username and password in an `wlmoradc` configuration file, setting them as follows:

```
$opt_username="myname/mypassword";
```

This data will not be visible via `ps`. Set the configuration file to be readable only by root, protecting the integrity of the password.

For information on the `wlmoradc` configuration file, see wlmoradc(1M).

Potential password visibility/file tampering

With the WLM and `wlmoradc` configuration files, be careful what commands are specified, and set the UNIX file permissions appropriately. In particular, having world write permissions on the file or placing the file in a world-writable directory could allow other users to edit or replace the file, causing `wlmoradc` or WLM to execute the new commands when it is next invoked.

Also, be sure to set the read permissions to limit access when either of these files contains a username and password for an instance.

Potential perl code issue in the configuration file

The `wlmoradc` configuration file is a perl file. It is specified through the `wlmoradc` command-line option `--configfile` *`file`*. This file is executed, so if it contains malicious commands, your system could be at risk.

Be careful what commands are specified, and set the UNIX file permissions appropriately for the configuration file. In particular, having world write permissions on the file or placing the file in a world-writable directory could allow other users to edit or replace the file, causing `wlmoradc` to execute the new commands when it is next invoked.

`wlmoradc` issues a warning if the configuration file is world-writable or is in a world-writable directory, but still executes it. `wlmoradc` does not check the parent directories of the file.

Potential SQL code issue using `--sqlfile` *`file`*

`wlmoradc` reads in the SQL file *`file`* when `--sqlfile` *`file`* is specified on the command line. This SQL file is executed; thus, malicious code could damage the database, or—via the SQL HOST command—damage or compromise the surrounding UNIX environment.

With the SQL file, be careful what commands are specified, and set the UNIX file permissions appropriately. In particular, having world write permissions on the file or placing the file in a world-writable directory could allow other users to edit or replace the file, causing `wlmoradc` to execute the new commands when it is next invoked.

`wlmoradc` issues a warning if the SQL file is world-writable or is in a world-writable directory, but still executes it. It does not check the parent directories of the file.

Troubleshooting

This section explains how to view error and informational messages. In addition, it discusses debugging techniques, as well as common errors and how to fix them.

Error and informational messages

Messages are provided on stderr and through syslog.

- stderr

  You can redirect stderr with the `wlmoradc` option `--stderrfile` *file*

  Alternatively, you can use the `coll_stderr` keyword in your WLM configuration. This keyword allows you to capture stderr for all data collectors in your configuration, not just `wlmoradc`. You can send stderr to syslog or another file of your choosing.

- syslog

  Typically, syslog is at /var/adm/syslog/syslog.log; if not, see /etc/syslog.conf for a pointer to syslog on your system.

Also, WLM has a message log at /var/opt/wlm/msglog.

Debugging suggestions

Here are some pointers when you are debugging:

1. Check syslog for indication of any problems.
2. Use the `wlmoradc` option `--debug 1` to check that `wlmoradc` settings match expected values.

Common errors and their solutions

Here are some of the more common problems you may encounter. The error messages below are not exactly the output you will see, but rather are indicative of a general class of errors that may occur. Solutions are also provided.

`smooth`: nonnumeric value ...

Run the tool with the option `--debug` to see what values are being fed into it by the data collector (either `wlmoradc` or a user-supplied collector). Then fix `wlmoradc` or the user-supplied collector feeding the invalid input to `smooth`.

`wlmoradc`: can't connect to instance

Confirm the `--home`, `--instance`, and `--username` values (or their `$opt_`*`name`* equivalents in the `wlmoradc` configuration file) by setting the associated environment variables and running SQL*Plus manually, as in the following POSIX shell example:

Step 1. Set the `ORACLE_HOME` env variable to the `--home` value:

```
# export ORACLE_HOME=/oracle/app/oracle/product/8.1.5
```

Step 2. Set the `ORACLE_SID` env variable to the `--instance` value:

```
# export ORACLE_SID=myinstance
```

Step 3. Use the `--username`/`--password` values with SQL*Plus to confirm that the instance can be connected to:

```
# $ORACLE_HOME/bin/sqlplus scott/tiger
```

```
SQL*Plus: Release 8.1.5.0.0 - Production on Thu Jan 18 19:48:40 2001

(c) Copyright 1999 Oracle Corporation.  All rights reserved.

Connected to:
Oracle8i Enterprise Edition Release 8.1.5.0.0 - Production
With the Partitioning and Java options
PL/SQL Release 8.1.5.0.0 - Production

SQL> exit
Disconnected from Oracle8i Enterprise Edition Release 8.1.5.0.0 - Production
With the Partitioning and Java options
PL/SQL Release 8.1.5.0.0 - Production
```

`wlmoradc`: bad perl code in configfile

Use `perl -c` to check the syntax of the configuration file, then run `wlmoradc` by hand.

`wlmoradc`: bad SQL code in *`sqlfile`* or *`sqlstring`*

Use SQL*Plus manually to check syntax and functionality, then use the file or string with `wlmoradc`.

`wlmoradc`: missing environment variables

Run `wlmoradc` using `wlmd`, but include the options `--debug 2 --stderrfile /tmp/test` to see the complete environment. Make sure the `wlmoradc` command line has the correct options, including `-h` *`oracle_home`* (`--home` *`oracle_home`*).

Related information

For information about ODBTK and its related products, consult the following documentation:

- HP-UX Workload Manager Oracle Database Toolkit
  - wlmoradc(1M) man page
  - smooth(1M) man page
  - wlmtk(5) man page (Workload Manager Toolkits overview)
  - HP-UX Workload Manager Toolkits A.01.08 Release Notes /opt/wlm/toolkits/doc/Rel_Notes
- Oracle
  - Oracle documentation

- HP-UX Workload Manager
  - wlm(5) man page
  - wlmd(1M) man page
  - wlmconf(4) man page
  - wlmrcvdc(1M) man page
  - White paper on writing data collectors (also known as "performance monitors")
    /opt/wlm/share/doc/howto/perfmon.html
  - *HP-UX Workload Manager User's Guide*:
    http://docs.hp.com/hpux/netsys
    /opt/wlm/share/doc/WLMug.pdf
  - Workload Management homepage:
    http://www.hp.com/go/wlm
- SQL
  - *SQL*Plus user's guide and Reference*
  - *Oracle 8i SQL Reference*
  - *Oracle 8i Tuning* (provides information on the SQL trace and TKPROF utilities for fine-tuning your SQL statements)

HP-UX WLM Oracle Database Toolkit: Providing Database Metrics to WLM

Related information

3 HP-UX WLM Pay Per Use Toolkit (PPUTK)

The toolkit, including its `utilitydc` command, are being deprecated. Support for the command will be removed in a future release. In the meantime, for `utilitydc` documentation, see the utilitydc(1M) man page.

Please begin upgrading to the simpler and more robust solution provided by `wlmpard` and the new keyword `utilitypri`. For information on the new solution and keyword, see the wlmpard(1M) and wlmparconf(4) man pages.

4 HP-UX WLM Apache Toolkit (ApacheTK)

The Apache web server is a very popular front end for a number of different web-based systems. The basic Apache process, `httpd`, and other processes it starts and manages comprise much of the workload for many middle tier systems in the enterprise, including many resource-intensive workloads.

Beyond commercial packages, much of the emerging web services infrastructure is being built around Apache or Apache-managed workloads. These include such mechanisms as XML-RPC and SOAP interfaces implemented as CGI, mod_*, or servlets combined in various ways with Apache.

If you need a more robust Apache-based web server for the enterprise, especially for a mission-critical environment, HP recommends the HP-UX Apache-based Web Server. Engineered through state-of-the art processes for the highest quality and tailored to run smoothly on the HP-UX platforms, the HP-UX Apache-based Web Server is a total solution for web serving deployment in the enterprise. The Open Source Apache Web Server software developed by the Apache Software Foundation (Apache HTTP Server Project described at http://httpd.apache.org) serves as the foundation for the HP-UX Apache-based Web Server. In addition to the base HTTP server, HP has combined numerous popular modules from other Open Source projects as well as HP-developed valued features, such as performance tuning, user guides, and security modules.

WLM can help you manage and prioritize Apache-based workloads. WLM can be used with Apache processes, Tomcat, CGI scripts, and related tools using the HP-UX Apache-based Web Server.

Supported installations

A large component of ApacheTK is the white paper *Using HP-UX Workload Manager with Apache*. This paper illustrates how to manage a variety of Apache-based workloads. This white paper was written using the following packages:

- HP-UX Workload Manager (WLM) Version A.02.02 (B8843CA)

  However, you can use WLM Version A.01.01 or later in most circumstances to accomplish the management outlined in the paper. Later WLM versions should also perform as expected.

- HP-UX Apache-based Web Server B.1.0.09.01 based on Apache version 2.0.47

  For information on techniques for use with Apache 1.3.x, see the "information library" page available on http://www.hp.com/go/wlm.

- HP's perl B.5.6.1.C available from http://software.hp.com/

Why use ApacheTK with WLM?

Assume your enterprise (or corporate) applications use Apache as a front end to dispatch work to workloads running Java software, CGI scripts, and other such processes. You can then use WLM and ApacheTK to manage these processes in a manner that reflects the business priorities they support.

Tools in ApacheTK

This toolkit includes the following tools:

Using HP-UX Workload Manager with Apache

This white paper provides examples showing how to manage various Apache-based workloads.

`time_url_fetch`

`time_url_fetch` is a data collector for Workload Manager. This collector uses the Apache benchmark tool `ab` to time URL response times.

`wlm_watch.cgi`

`wlm_watch.cgi` is a script that provides web-based, view-only reports of the standard `prmmonitor`, `prmlist`, and other WLM and PRM command-line tools.

Where do I get ApacheTK?

ApacheTK is installed at /opt/wlm/toolkits/apache/.

If it is not already installed on your system, you can download it free of charge from http://www.hp.com/go/wlm.

What comes with ApacheTK?

ApacheTK comes with the white paper, the `time_url_fetch` and `wlm_watch.cgi` tools, the associated man pages, and a number of example configuration files. Table 4-1 lists many of the included files.

Table 4-1 ApacheTK directories and files

Directory/file	Description
/opt/wlm/share/man/man5.Z/wlmtk.5	Man page that provides an overview of the Workload Manager Toolkits
/opt/wlm/toolkits/apache/	Directory for ApacheTK and all its files
/opt/wlm/toolkits/doc/WLMTKug.pdf	WLMTK User's Guide (PDF)
/opt/wlm/toolkits/doc/WLMTKug.ps	WLMTK User's Guide (PostScript)
/opt/wlm/share/man/man1m.Z/time_url_fetch.1m	Man page for the WLM data collector
/opt/wlm/share/man/man1m.Z/wlm_watch.1m	Man page for the WLM report viewer
/opt/wlm/toolkits/README	Readme for all the WLM Toolkits
/opt/wlm/toolkits/apache/README	Readme for the WLM Apache Toolkit
/opt/wlm/toolkits/apache/bin/time_url_fetch	The WLM data collector, which provides data to WLM

Table 4-1 ApacheTK directories and files (Continued)

Directory/file	Description
/opt/wlm/toolkits/apache/config/apache_vs_oradb.wlm	Example WLM configuration file*
/opt/wlm/toolkits/apache/config/apache_vs_batch.wlm	Example WLM configuration file*
/opt/wlm/toolkits/apache/config/apache_vs_single_cgi.wlm	Example WLM configuration file*
/opt/wlm/toolkits/apache/config/apache_vs_tomcat.wlm	Example WLM configuration file*
/opt/wlm/toolkits/apache/config/apache1_servlet_vs_apache2_ssl.wlm /opt/wlm/toolkits/apache/config/apache1_servlet_vs_apache2_ssl.conf	Example WLM and Apache configuration files*
/opt/wlm/toolkits/apache/config/apache_2mods_rewrite.wlm /opt/wlm/toolkits/apache/config/apache_2mods_rewrite.conf	Example WLM and Apache configuration files*
/opt/wlm/toolkits/apache/config/apachectl.split	Example utility file*
/opt/wlm/toolkits/apache/config/apache_single_servlet.wlm	Example WLM configuration file*
/opt/wlm/toolkits/apache/config/apache_resptimes.wlm	Example WLM configuration file*
/opt/wlm/toolkits/apache/doc/apache_wlm_howto.html	*Using HP-UX Workload Manager with Apache* white paper
/opt/wlm/toolkits/apache/doc/images/	Images used in the apache_wlm_howto .html file

* For descriptions of the example files, see “Example WLM and Apache configurations” on page 59.

How do I install ApacheTK?

Use the SD-UX `swinstall` command to install ApacheTK, which is part of the WLM Toolkits product (product number T1302AA).

The Apache toolkit is installed at /opt/wlm/toolkits/apache/. The man pages are installed at /opt/wlm/share/man/.

For more information about installation procedures and related issues, refer to the following documentation:

- *Managing HP-UX Software with SD-UX*
- swinstall (1M) man page

How do I use ApacheTK?

The best way to use ApacheTK is to read the white paper *Using HP-UX Workload Manager with Apache* available from http://www.hp.com/go/wlm and /opt/wlm/toolkits/apache/doc/apache_wlm_howto.html. The paper guides you through the steps and tools needed to have WLM:

- Separate Apache from Oracle database instances
- Separate Apache from batch work
- Isolate a resource-intensive CGI workload
- Isolate a resource-intensive servlet workload
- Separate all Apache Tomcat workloads from other Apache workloads
- Separate two departments' applications using two Apache instances
- Separate module-based workloads with two Apache instances
- Manage Apache allocation by performance goal

Example WLM and Apache configurations

ApacheTK provides many WLM and Apache configurations in the /opt/wlm/toolkits/apache/config/ directory. These configurations are:

- apache_vs_oradb.wlm

  WLM configuration file to demonstrate separating an Oracle database instance workload from `httpd` and its children.

- apache_vs_batch.wlm

  WLM configuration file to demonstrate separating a batch job workload from `httpd` and its children.

- apache_vs_single_cgi.wlm

  WLM configuration file to demonstrate separating a single, resource-intensive CGI workload from `httpd`.

- apache_vs_tomcat.wlm

  WLM configuration file to demonstrate separating the Tomcat JVM from `httpd`.

- apache1_servlet_vs_apache2_ssl.wlm, apache1_servlet_vs_apache2_ssl.conf

  Apache and WLM configurations to support two separate Apache instances, one running Tomcat workloads and one running mod_ssl.

- apache_2mods_rewrite.wlm, apache_2mods_rewrite.conf

  Apache and WLM configurations to support two separate Apache instances, each running different mod_* (module) based workloads in a workload group.

- apache1.split

  File used to start and stop separate Apache instances for the apache_2mods_rewrite.wlm configuration.

- apache_single_servlet.wlm

  WLM configuration to support running a specific, resource-intensive servlet in a second copy of Tomcat.

- apache_resptimes.wlm

  WLM configuration to support managing an Apache instance by URL fetch response time.

The above configuration descriptions are based on Revision 1.12 of the file /opt/wlm/toolkits/apache/config/README.

Command reference

ApacheTK consists of the `time_url_fetch` data collector and the `wlm_watch.cgi` CGI script.

`time_url_fetch`

`time_url_fetch` is a data collector for Workload Manager. It is specified in the WLM configuration file. The WLM daemon `wlmd` invokes `time_url_fetch` at startup. It provides URL response times for wlmd(1M).

For more information on this utility—including its options, see the time_url_fetch(1M) man page.

`wlm_watch.cgi`

`wlm_watch.cgi` is a CGI script that provides a web interface to various Workload Manager monitoring tools.

For more information on setting up and using this utility, see the wlm_watch(1M) man page.

Related information

For information about ApacheTK and its related products, consult the following documentation:

- Workload Manager Apache Toolkit
  - Using HP-UX Workload Manager with Apache-based Applications white paper /opt/wlm/toolkits/apache/doc/apache_wlm_howto.html
  - time_url_fetch(1M) man page
  - wlm_watch(1M) man page
  - wlmtk(5) man page (Workload Manager Toolkits overview)
  - Workload Manager Toolkits A.01.08 Release Notes /opt/wlm/toolkits/doc/Rel_Notes
- HP-UX Workload Manager
  - wlm(5) man page
  - wlmd(1M) man page
  - wlmconf(4) man page
  - *HP-UX Workload Manager User's Guide*: http://docs.hp.com/hpux/netsys /opt/wlm/share/doc/WLMug.pdf
  - Workload Management homepage: http://www.hp.com/go/wlm

HP-UX WLM Apache Toolkit (ApacheTK)

Related information

5 HP-UX WLM BEA WebLogic Server Toolkit (WebLogicTK)

WLM can help you manage and prioritize BEA WebLogic Server™ workloads.

Supported installations

A large component of WebLogicTK is the white paper *Using HP-UX Workload Manager with BEA WebLogic*. This paper illustrates how to manage a variety of WebLogic workloads. This white paper was written using the following packages:

- HP-UX WLM Version A.02.02 on HP-UX 11i v1 (B.11.11)

  Processor sets (PSETs), which are available starting in HP-UX 11i v1, are used as workload containers.
- BEA WebLogic Server 7.0, BEA WebLogic Server 8.1

Why use WLM with BEA WebLogic?

Using WLM with WebLogic you can move CPUs to or from WebLogic instances as needed to maintain acceptable performance. By managing the instances' CPU resources, the instances will tend to use less net CPU resources over time. You can then use the additional CPU resources for other computing tasks.

As indicated above, WLM and WebLogicTK control CPU allocation to individual WebLogic instances. However, the latest version of the paper *Using HP-UX Workload Manager with BEA WebLogic* expands the methods for controlling instances to control WebLogic Server clusters.

Tools in WebLogicTK

This toolkit includes the following tools:

Using HP-UX Workload Manager with BEA WebLogic

This white paper, in the file weblogic_wlm_howto.html in the directory /opt/wlm/toolkits/weblogic/doc/, provides examples showing how to manage various WebLogic workloads.

`wlmwlsdc`

`wlmwlsdc` is a data collector for Workload Manager. This collector tracks metrics created to gauge how busy an instance is.

`expsmooth`

NOTE

As of WLM A.02.02, it is recommended that you use the `cntl_smooth` WLM keyword instead of the `expsmooth` tool. For information on `cntl_smooth`, see the wlmconf(4) man page.

`expsmooth` is a filter that smooths values from WLM data collectors by computing a running average in which the importance of old values diminishes (decays) over time.

`expsmooth` uses /opt/perl/bin/perl.

Where do I get WebLogicTK?

WebLogicTK is installed at /opt/wlm/toolkits/weblogic/.

If it is not already installed on your system, you can download it free of charge from http://www.hp.com/go/wlm.

What comes with WebLogicTK?

WebLogicTK comes with the white paper, the `wlmwlsdc` and `expsmooth` tools, the associated man pages, a number of example configuration files, and properties files used by `wlmwlsdc`. Table 5-1 lists many of the included files.

Table 5-1 WebLogicTK directories and files

Directory/file	Description
/opt/wlm/share/man/man5.Z/wlmtk.5	Man page that provides an overview of the Workload Manager Toolkits
/opt/wlm/toolkits/weblogic/	Directory for WebLogicTK and all its files
/opt/wlm/toolkits/doc/WLMTKug.pdf	WLMTK User's Guide (PDF)
/opt/wlm/toolkits/doc/WLMTKug.ps	WLMTK User's Guide (PostScript)
/opt/wlm/share/man/man1m.Z/wlmwlsdc.1m	Man page for the `wlmwlsdc` data collector
/opt/wlm/share/man/man1m.Z/expsmooth.1m	Man page for `expsmooth`
/opt/wlm/toolkits/README	Readme for all the WLM Toolkits
/opt/wlm/toolkits/weblogic/README	Readme for the WLM WebLogic Toolkit

Table 5-1 WebLogicTK directories and files (Continued)

Directory/file	Description
/opt/wlm/toolkits/weblogic/bin/wlmwlsdc	The WLM data collector, which provides data to WLM
/opt/wlm/toolkits/weblogic/bin/expsmooth	The data smoothing utility
/opt/wlm/toolkits/weblogic/config/manual_cpucount.wlm	Example WLM configuration file*
/opt/wlm/toolkits/weblogic/config/wls_1inst_3level.wlm	Example WLM configuration file*
/opt/wlm/toolkits/weblogic/config/wls_1inst_CPUusage.wlm	Example WLM configuration file*
/opt/wlm/toolkits/weblogic/config/wls_1inst_hybrid_qc.wlm	Example WLM configuration file*
/opt/wlm/toolkits/weblogic/config/wls_1inst_q_goal.wlm	Example WLM configuration file*
/opt/wlm/toolkits/weblogic/config/wls_2inst_3level.wlm	Example WLM configuration file*
opt/wlm/toolkits/weblogic/config/wls_2inst_CPUusage.wlm	Example WLM configuration file*
/opt/wlm/toolkits/weblogic/config/wls_2inst_hybrid_qc.wlm	Example WLM configuration file*
/opt/wlm/toolkits/weblogic/config/wls_2inst_q_goal.wlm	Example WLM configuration file*
/opt/wlm/toolkits/weblogic/config/wls_2queue_2inst.wlm	Example WLM configuration file*
/opt/wlm/toolkits/weblogic/config/instA_qbusy.props	Example WLM properties file for `wlmwlsdc`*

Table 5-1 WebLogicTK directories and files (Continued)

Directory/file	Description
/opt/wlm/toolkits/weblogic/config/instA_qdepth.props	Example WLM properties file for `wlmwlsdc`*
/opt/wlm/toolkits/weblogic/config/instA_qdepth_hipri.props	Example WLM properties file for `wlmwlsdc`*
/opt/wlm/toolkits/weblogic/config/instB_qbusy.props	Example WLM properties file for `wlmwlsdc`*
/opt/wlm/toolkits/weblogic/config/instB_qdepth.props	Example WLM properties file for `wlmwlsdc`*
/opt/wlm/toolkits/weblogic/doc/weblogic_wlm_howto.html	*Using HP-UX Workload Manager with BEA WebLogic* white paper

* For descriptions of the example files, see “Example WLM configurations and properties files” on page 68.

How do I install WebLogicTK?

Use the SD-UX `swinstall` command to install WebLogicTK, which is part of the WLM Toolkits product (product number T1302AA).

The toolkit is installed at /opt/wlm/toolkits/weblogic/. The man pages are installed at /opt/wlm/share/man/.

For more information about installation procedures and related issues, refer to the following documentation:

- *Managing HP-UX Software with SD-UX*
- swinstall (1M) man page

How do I use WebLogicTK?

The best way to use WebLogicTK is to read the white paper *Using HP-UX Workload Manager with BEA WebLogic Server* available from:

- http://www.hp.com/go/wlm
- /opt/wlm/toolkits/weblogic/doc/weblogic_wlm_howto.html

The paper guides you through the steps and tools needed to use WLM with WebLogic.

Example WLM configurations and properties files

WebLogicTK provides many WLM configurations in the /opt/wlm/toolkits/weblogic/config/ directory. Each configuration file represents a use case documented in the white paper. These configurations are:

- manual_cpucount.wlm

  WLM configuration file to help with benchmarking and capacity-planning tasks. You can resize the workload (WebLogic instance) PSET by manually running `wlmsend`. For example:

  ```
  % /opt/wlm/bin/wlmsend wls1_grp.desired.cpucount 2.0
  ```

  would assign two CPUs to the `wls1_grp`.

- wls_1inst_3level.wlm

  WLM configuration file to demonstrate how to control a single WebLogic instance and monitor its execution queue to determine if it is idle (normal allocation), busy (boost of one additional CPU), or very busy (boost of two additional CPUs).

- wls_2inst_3level.wlm

  WLM configuration file to demonstrate how to control two WebLogic instances and monitor the execution queue of each to determine if it is idle (normal allocation), busy (boost of one additional CPU), or very busy (boost of two additional CPUs).

- wls_1inst_CPUusage.wlm

  WLM configuration file to demonstrate how to control a single WebLogic instance versus other workloads. The WebLogic instance's workload group starts with one CPU. As its CPU consumption grows, it is allocated more CPUs, to a maximum of four.

- wls_2inst_CPUusage.wlm

  WLM configuration file to demonstrate how to control a pair of WebLogic instances. Each WebLogic instance's workload group starts with one CPU. As each instance's CPU consumption grows, it is allocated more CPUs, to a maximum of four.

- wls_1inst_q_goal.wlm

  WLM configuration file to demonstrate how to control a WebLogic instance. The WebLogic instance's workload group is allocated from one to four CPUs, which `wlmd` allocates in an attempt to keep the instance's queue_busy metric below 0.

- wls_2inst_q_goal.wlm

  WLM configuration file to demonstrate how to control a pair of WebLogic instances. Each WebLogic instance's workload group is allocated from one to four CPUs, which `wlmd` allocates in an attempt to keep the instance's queue_busy metric below 0.

- wls_2queue_2inst.wlm

  WLM configuration file to demonstrate how to control a pair of WebLogic instances, instA and instB. InstA is higher priority and the administrator has created a special 'hipri' queue for high-priority work. Each WebLogic instance's workload group starts with one CPU. If the hipri queue depth has work, instA's group is allocated an additional CPU. If the instA default queue depth shows activity, its group is allocated an additional CPU. Lastly, if the instB queue depth has work, its workload group is allocated an additional CPU, if any are left.

- wls_1inst_hybrid_qc.wlm

  WLM configuration file to demonstrate how to control a WebLogic instance. The WebLogic instance's workload group is allocated from one to four CPUs, which `wlmd` allocates using a pair of SLOs: a CPU usage SLO similar to the wls_2inst_CPUusage.wlm example, and a queue metric based goal similar to the wls_2inst_q_goal.wlm example. This two SLO per group or hybrid approach will control both queue and non-queue based instance workloads.

- wls_2inst_hybrid_qc.wlm

  WLM configuration file to demonstrate how to control a pair of WebLogic instances. Each WebLogic instance's workload group is allocated from one to four CPUs, which `wlmd` allocates using a pair of SLOs: a CPU usage SLO similar to the wls_2inst_CPUusage.wlm example, and a queue metric based goal similar to the wls_2inst_q_goal.wlm example. This two SLO per group or hybrid approach will control both queue and non-queue based instance workloads.

The following are `wlmwlsdc` properties files. Each is used to tell the `wlmwlsdc` data collector which WebLogic instance to monitor and the metric to collect.

- instA_qbusy.props

  `wlmwlsdc` properties file with example properties showing how to fetch the queue_busy metric from WebLogic instance 'instA'. It is used in the single or multiple instance use cases whose resource allocations are based on the JMX ExecuteQueue MBean metrics.

- instB_qbusy.props

  `wlmwlsdc` properties file with example properties showing how to fetch the queue_busy metric from WebLogic instance 'instB'. It is used in the multiple instance use cases whose resource allocations are based on the JMX ExecuteQueue MBean metrics.

- instA_qdepth.props

  `wlmwlsdc` properties file with example properties showing how to fetch the queue_depth metric from WebLogic instance 'instA'. It is used in the single or multiple instance use cases whose resource allocations are based on the JMX ExecuteQueue MBean metrics.

- instB_qdepth.props

  `wlmwlsdc` properties file with example properties showing how to fetch the queue_depth metric from WebLogic instance 'instB'.

- instA_qdepth_hipri.props

  `wlmwlsdc` properties file with example properties showing how to fetch the queue_depth metric from WebLogic instance 'instA' high priority queue (named 'hipri'). It is used in the multiple instance, multiple queue use case whose resource allocations are based on the JMX ExecuteQueue MBean metrics.

The above descriptions are based on Revision 1.13 of the file /opt/wlm/toolkits/weblogic/config/README.

Command reference

WebLogicTK consists of the `wlmwlsdc` data collector and the `expsmooth` command.

wlmwlsdc

`wlmwlsdc` is a data collector for Workload Manager. It is specified in the WLM configuration file. The WLM daemon `wlmd` invokes `wlmwlsdc` at startup. It provides metrics indicating how busy WebLogic instances are.

For more information on this utility—including its options, see the wlmwlsdc(1M) man page.

expsmooth

NOTE As of WLM A.02.02, it is recommended that you use the `cntl_smooth` WLM keyword instead of the `expsmooth` tool. For information on `cntl_smooth`, see the wlmconf(4) man page.

`expsmooth` is a utility that smooths data (giving more importance to more recent data), softening spikes in the data, for consumption by WLM.

`expsmooth` uses /opt/perl/bin/perl.

For more information on setting up and using this utility, see the expsmooth(1M) man page.

What about security issues?

Be aware of the following security issues so that you can better protect your system.

Potential password visibility

Be sure to set the read permissions to limit access when your .props files contain a username and password for an instance.

posix_sh_setup_file issues

The *posix_sh_setup_file*, used with `wlmwlsdc`, can potentially pose security issues. For information on the issues, as well as several methods for alleviating the issues, see section the white paper *Using HP-UX Workload Manager with BEA WebLogic*, available at /opt/wlm/toolkits/weblogic/doc/weblogic_wlm_howto.html.

Related information

For information about WebLogicTK and its related products, consult the following documentation:

- Workload Manager BEA WebLogic Server Toolkit
  - *Using HP-UX Workload Manager with BEA WebLogic*
    /opt/wlm/toolkits/weblogic/doc/weblogic_wlm_howto.html
  - wlmwlsdc(1M) man page
  - expsmooth(1M) man page
  - wlmtk(5) man page (Workload Manager Toolkits overview)
  - Workload Manager Toolkits A.01.08 Release Notes
    /opt/wlm/toolkits/doc/Rel_Notes
- HP-UX Workload Manager
  - wlm(5) man page
  - wlmd(1M) man page
  - wlmconf(4) man page
  - *HP-UX Workload Manager User's Guide*:
    http://docs.hp.com/hpux/netsys
    /opt/wlm/share/doc/WLMug.pdf
  - Workload Management homepage:
    http://www.hp.com/go/wlm

Related information

6 HP-UX WLM Duration Management Toolkit and HP-UX WLM Toolkit for Base SAS Software

HP-UX WLM can provide a number of features to help you better use your HP servers. For example, WLM and its Duration Management Toolkit (DMTK), can help by reducing the need for overprovisioning CPU resources—granting jobs only the amount of CPU needed to complete within a certain timeframe. This feature, known as duration management, is useful in Base SAS® environments and many other environments. Furthermore, even without DMTK, WLM can grant computing resources upon demand—providing an “express lane”—to ensure that your most important jobs are given top priority. Express lanes can be particularly beneficial in a Base SAS environment—where any user has the ability to launch a job that can, regardless of its business priority, significantly affect the performance of other jobs.

This chapter explains how to use duration management and express lanes.

DMTK offers integrators and administrators nonintrusive solutions to managing jobs through WLM configuration files. For developers, the toolkit offers a SAS macro to programmatically manage SAS jobs to take advantage of WLM features.

For more information on WLM, see the *HP-UX Workload Manager User’s Guide* or the wlm(5) man page.

NOTE The functionality described below that pertains to SAS is actually from the HP-UX WLM Toolkit for Base SAS Software (SASTK). However, as SASTK is tightly coupled with DMTK, they are discussed together.

Supported installations

DMTK works with:

- HP-UX WLM Version A.02.02 on HP-UX 11i v1 (B.11.11), and HP-UX 11i v2 (B.11.23)
- HP-UX WLM Version A.02.01.01 on HP-UX 11i v2 (B.11.23)
- HP-UX WLM Version A.02.01 on HP-UX 11i v1 (B.11.11)
- HP-UX WLM Version A.01.02 (which is the first version to include the `cpushares` keyword) on HP-UX 11i v1 (B.11.11)

SASTK works with the WLM versions listed above and with:

- Base SAS software Version 8.1 and later

Audience for the DMTK / SASTK documentation

This information is for anyone in need of duration management for long-running jobs or examples illustrating express lanes. Also, this documentation is for integrators, administrators, and developers working with Base SAS software in an environment where certain performance levels must be maintained.

Why use DMTK / SASTK with WLM?

The benefits of using DMTK / SASTK with WLM are the:

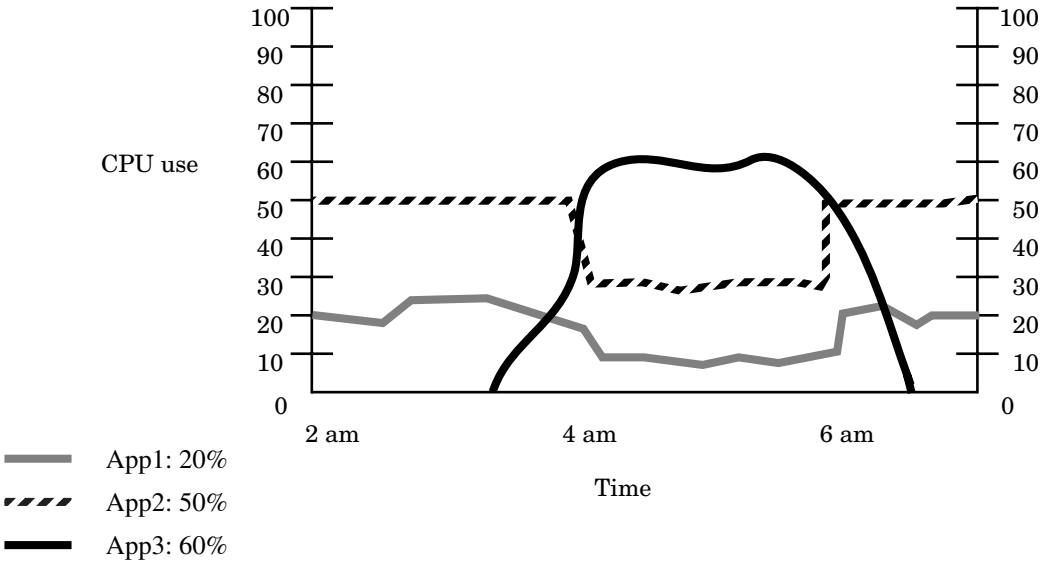
- Examples that show how express lanes can be used to quickly complete urgent jobs
- Ability to define goals for job duration so that jobs get just the right amount of CPU—not too much or too little—to finish within user-specified time ranges

For examples showing how to set up these types management, see the files with names ending in ".wlm" in the directories /opt/wlm/toolkits/duration/config/ and /opt/wlm/toolkits/sas/config/.

NOTE DMTK does not reduce the amount of CPU time an application must have to complete; it merely attempts to regulate the application's access to CPU resources. For example, if an application takes one hour to complete when using 100% of the CPU, DMTK cannot make its duration less than one hour.

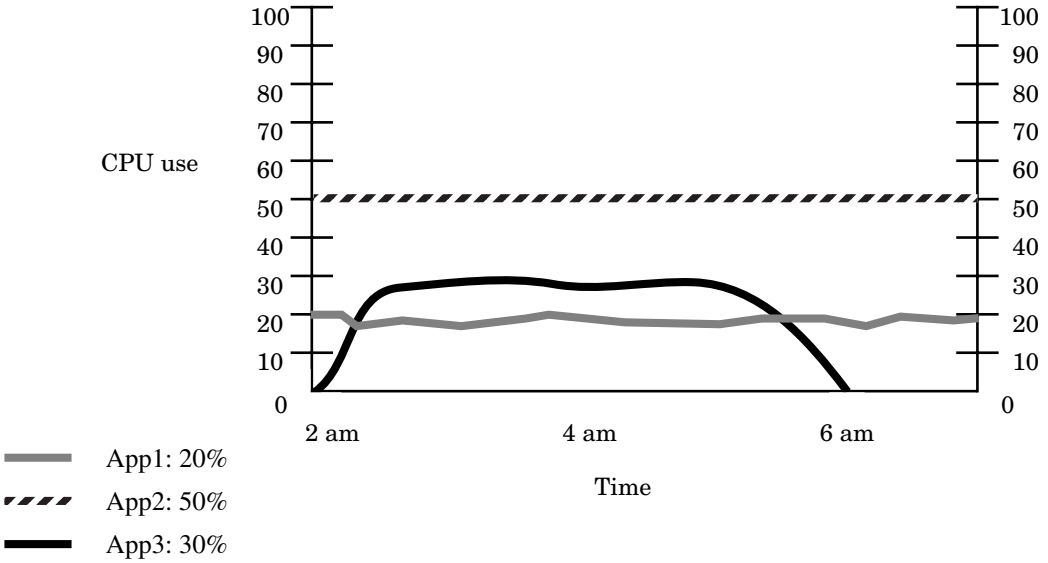
Figure 6-1 shows three unrelated applications, App1, App2, and App3, sharing a server without the benefit of duration management. The applications most critical to the business, App1 and App2, run continuously and tend to consume about 20% and 50% of the CPU, respectively. The CPU-intensive App3 is started once a day and runs for about two hours, using about 60% of the CPU and finishing about 6 am. Unfortunately, it also greatly reduces the CPU that App1 and App2 receive. These other applications quickly recover once App3 finishes, though.

Figure 6-1 Unrelated applications without duration management



In Figure 6-2, we have the same three applications, App1, App2, and App3. This time, however, duration management is used. App3 is started earlier in the day—with duration management ensuring it still finishes by 6 am. With duration management in effect, App3 now consumes about 30% of the CPU. Consequently, App1 and App2 are able to user their typical 20% and 50% of the CPU.

Figure 6-2 Unrelated applications with duration management



Another example of when DMTK / SASTK is useful is the case when several related processes run simultaneously. Duration management can remove spikes in the load by making all the processes complete about the same time. Each process needs to be in a workload group of its own so that WLM can manage the durations individually.

Tools in DMTK / SASTK

DMTK and SASTK include the following items for duration management:

`wlmdurdc`

`wlmdurdc` is a duration data collector for Workload Manager. It takes the following items—in the order shown—as input:

- PID of the process needing duration management
- Profile value

  (CPU time in integer seconds needed by the process to finish when unlimited CPU resources are available)
- Desired duration of the managed process in whole seconds (integer seconds)

`wlmdurdc` uses these values to determine how many CPU cycles the process is currently using, how many cycles it needs to finish, and the desired duration to determine by when the process should get all those cycles. Based on these values, `wlmdurdc` sends a request to WLM for more CPU for the process's workload group if the process needs more resources to finish within the timeframe. Similarly, a request for fewer CPU resources is sent to WLM if the process is finishing too quickly.

`hp_wlmtk_goals_report`

This macro is useful in instrumenting SAS jobs to:

- Get profile data indicating elapsed time for a job
- Inform `wlmdurdc` of the job's percent completed

Overview of how WLM's duration management works

Before you can manage duration, you need profile values for your applications. These values represent the CPU time needed by your application to complete. You can get these values as explained in "Completing an application within a certain duration (duration management)" on page 87. You also need to create a discovery command to identify each application you plan to manage. Each discovery command, which runs exactly once during the life of its target application, will determine the PID of its target application. This command then outputs the PID, the profile value, and the desired duration for its target application. The duration data collector `wlmdurdc` takes the information from the discovery command, does some calculations, then feeds the result into WLM. Next, WLM determines a new CPU allocation for the workload group containing the target application. This allocation is based on whether the application is going to complete too quickly or too slowly. If the application is on schedule to complete at the desired duration, WLM simply attempts to maintain the current CPU allocation. Figure 6-3 illustrates this process. The graphic applies to any environment where SAS jobs do not exist or are not instrumented with the macro `hp_wlmtk_goals_report`.

Figure 6-3 Overview of how applications can be managed

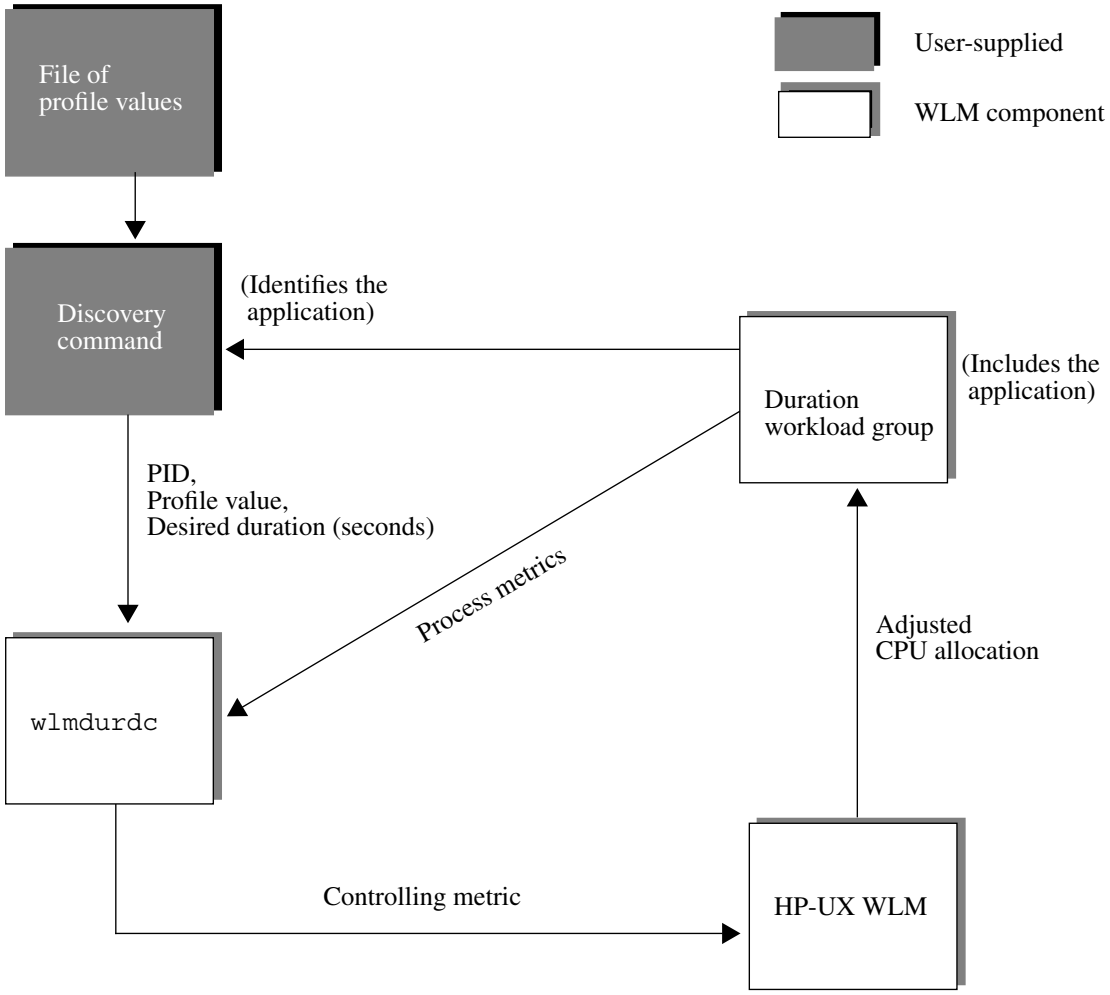
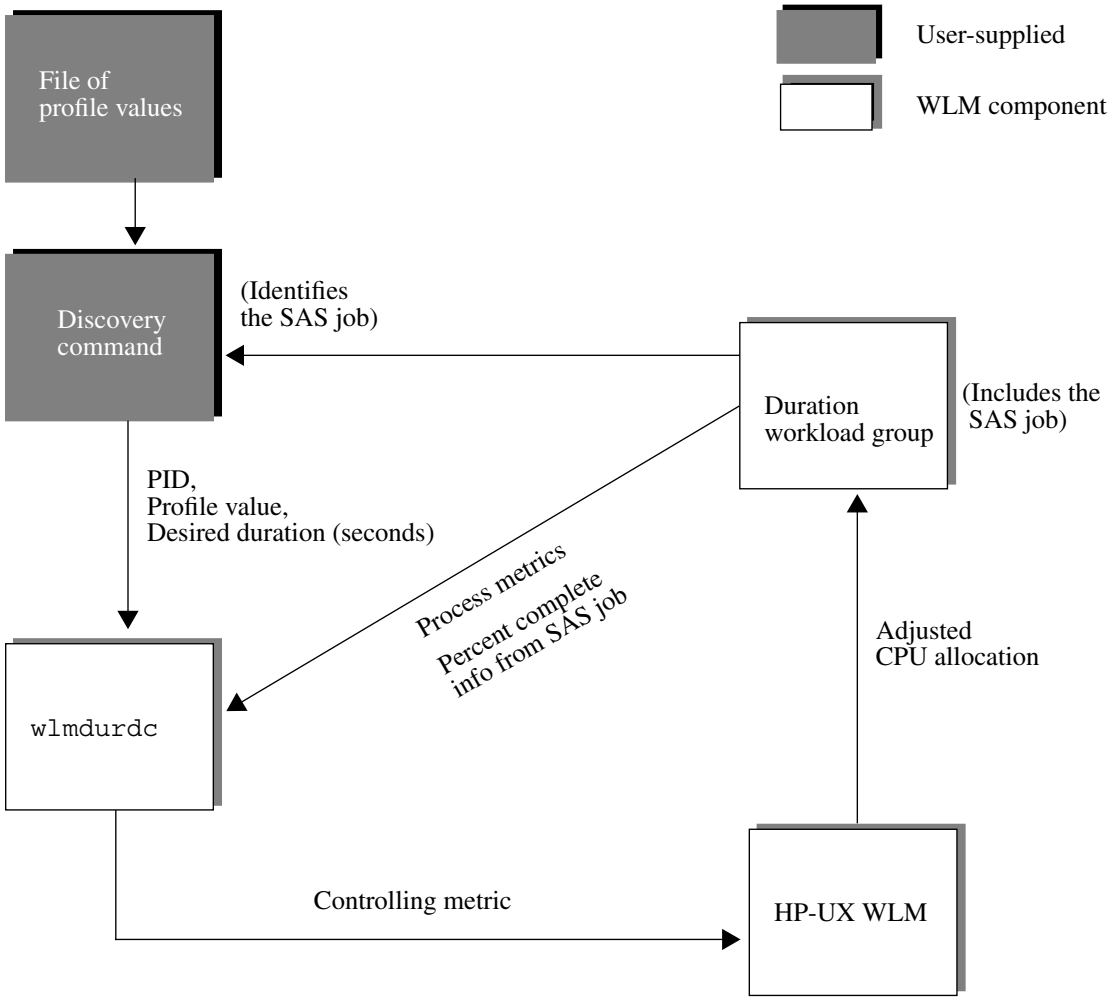


Figure 6-4 shows how duration management works in environments with SAS. It is the same situation as in Figure 6-3 except that you can instrument the SAS jobs. You instrument a SAS job with the macro `hp_wlmtk_goals_report` to fine-tune duration management. This fine-tuning is desirable when there are fluctuations in data set size. For example, the macro can inform `wlmdurdc` when the job is 25% complete, despite the fact that the time to complete 25% of the job changes as the size of the data set changes.

Figure 6-4 Overview of how SAS jobs can be managed



Of course, both of these diagrams ignore the fact that other applications may be running on the system and competing for resources. For a diagram that shows the interaction of several applications, see the *HP-UX Workload Manager User's Guide*.

Where do I get DMTK / SASTK?

DMTK is installed at /opt/wlm/toolkits/duration/. SASTK is installed at /opt/wlm/toolkits/sas/.

If these toolkits are not already installed on your system, you can download them free of charge from http://www.hp.com/go/wlm.

What comes with DMTK / SASTK?

DMTK / SASTK consist of a utility and a macro (`wlmdurdc` and `hp_wlmtk_goals_report`), the associated man pages, and a number of example configuration files. Table 6-1 lists many of the included files.

Table 6-1 DMTK/SASTK directories and files

Directory/file	Description
/opt/wlm/share/man/man5.Z/wlmtk.5	Man page that provides an overview of the WLM Toolkits
/opt/wlm/toolkits/README	Readme for all the WLM toolkits
/opt/wlm/toolkits/doc/WLMTKug.pdf	WLMTK User's Guide (PDF)
/opt/wlm/toolkits/doc/WLMTKug.ps	WLMTK User's Guide (PostScript)
/opt/wlm/toolkits/duration/	Directory for DMTK and all its files
/opt/wlm/toolkits/duration/bin/wlmdurdc	The WLM duration data collector, which provides data to WLM
/opt/wlm/share/man/man1m.Z/wlmdurdc.1m	Man page for the WLM duration data collector

Table 6-1 DMTK/SASTK directories and files (Continued)

Directory/file	Description
/opt/wlm/toolkits/duration/config/README	Readme for the WLM Duration Management Toolkit example configurations
/opt/wlm/toolkits/duration/config/duration.wlm	Example configuration file
/opt/wlm/toolkits/duration/config/expressconf.wlm	Example configuration file
/opt/wlm/toolkits/duration/config/expressconf_shares.wlm	Example configuration file
/opt/wlm/toolkits/duration/config/expressconf_usage.wlm	Example configuration file
/opt/wlm/toolkits/duration/examples/duration.sh	Example discovery command
/opt/wlm/toolkits/sas/config/sas_child_config.wlm	Example configuration file
/opt/wlm/toolkits/sas/config/sas_command_config.wlm	Example configuration file
/opt/wlm/toolkits/sas/examples/sas_child_discovery.sh	Example discovery command
/opt/wlm/toolkits/sas/examples/sas_command_discovery.sh	Example discovery command
/opt/wlm/toolkits/sas/macros/hp_wlmtk_goals_report.sas	SAS macro for reporting “percent complete” info
/opt/wlm/share/man/man1m.Z/hp_wlmtk_goals_report.1m	Man page for the SAS macro

How do I install DMTK / SASTK?

Use the SD-UX `swinstall` command to install DMTK / SASTK, which are both part of the WLM Toolkits product (product number T1302AA).

The DMTK is installed at /opt/wlm/toolkits/duration/. The SASTK is installed at /opt/wlm/toolkits/sas/.

For more information about installation procedures and related issues, refer to the following documentation:

- *Managing HP-UX Software with SD-UX*
- swinstall (1M) man page

Where can I find example files showing how to use DMTK / SASTK?

The directories /opt/wlm/toolkits/duration/config/ and /opt/wlm/toolkits/sas/config/ contain example WLM configuration files. The directories /opt/wlm/toolkits/duration/examples/ and /opt/wlm/toolkits/sas/examples/ contain example discovery commands. Copy these files to a working directory and modify them to match your environment.

Some of these examples are also in this document. For example WLM configuration files showing duration management and express lanes, see “Completing an application within a certain duration (duration management)” on page 87 and “Completing an urgent job as quickly as possible (express lanes)” on page 101.

For the example discovery command file, see “Example discovery command” on page 96.

How do I use DMTK / SASTK?

This section gives step-by-step procedures and examples for:

- Completing an application within a certain duration (duration management)
- Completing an urgent job as quickly as possible (express lanes)

Completing an application within a certain duration (duration management)

Duration management allows you to control the duration of a process by managing its CPU allocation. With DMTK, you manage duration by specifying how soon a job should complete. The job then regularly informs WLM how close it is to completion. Next, WLM will:

- Increase the job's CPU resources if the job may not complete in time
- Decrease the job's CPU resources if the job is completing too quickly

In either case, WLM and DMTK help you use just the right amount of CPU to get the job done on schedule. This leaves the rest of the CPU resources to be used by the other jobs and applications on the system, preventing the need for extra processing resources that may go unused in all but the most extreme cases.

To request that a job (target application) complete within a certain duration:

Step 1. Write a discovery command to identify and correlate each target application with its profile value and desired duration

Each discovery command must print to standard out the target application's:

- PID
- Profile value (CPU time needed by the process to finish when unlimited CPU resources are available) in integer seconds
- Desired duration in whole seconds (integer seconds)

Many applications' discovery commands can be simple shell scripts. Example discovery commands are included with DMTK and SASTK.

NOTE If your target application is started by a script, be sure that your discovery command looks for the executable rather than the script that starts it.

When first setting up a discovery command, just focus on getting the target application's PID to standard out. For now, you can enter a value of 0 (zero) for both the profile value and for the desired duration:

```
$PID 0 0
```

`wlmdurdc` will display the profile value for the application in syslog (/var/adm/syslog/syslog.log). You can then have your discovery command report the target application's actual profile value and your desired duration for the application.

For an example discovery command showing one way to get the PID, see the "Example discovery command" on page 96. For example discovery commands that are specific to SAS, see /opt/wlm/toolkits/sas/examples/.

Step 2. Define WLM workload groups in which to place your target applications

In a WLM configuration file, say *`configfile`*, define a WLM workload group for each target application.

NOTE For optimal duration management, place each application for which you'll be managing its duration in its own workload group. This ensures that the resources being allocated by WLM for the application's management are used only by that application.

For example, if we have two groups `GroupA` and `GroupB` for our target applications `app1` and `app2`, the group definitions would be:

```
prm {
   groups = GroupA : 1, GroupB : 2;
}
```

Step 3. Define SLOs for each group

Moving to the `slo` structure in the WLM configuration file, we define goals for each of the target applications. With duration management, the goal is always to have the metric from `wlmdurdc` be greater than 0. First, we set up an SLO for application1:

```
slo app1_slo {
   pri = 1;
   mincpu = 5;
   maxcpu = 75;
   entity = PRM group GroupA;
   goal = metric app1_met > 0;
}
```

We follow that with application2's SLO:

```
slo app2_slo {
   pri = 1;
   mincpu = 10;
   maxcpu = 60;
   entity = PRM group GroupB;
   goal = metric app2_met > 0;
}
```

Step 4. Set up `wlmdurdc` invocations

With the `slo` structures in place, we need to get values for the metrics app1_met and app2_met. These values come from `wlmdurdc`, which bases its values on the data from the discovery commands. We get the metric values through `wlmdurdc` invocations in `tune` structures (in the WLM configuration) named for the metrics:

```
tune app1_met {
   coll_argv = wlmrcvdc wlmdurdc discovery;
}

tune app2_met {
   coll_argv = wlmrcvdc wlmdurdc discovery;
}
```

NOTE The discovery commands are launched using `exec()`.

To see all the configuration file fragments from the previous steps together, see “Example WLM configuration file for duration management from steps above” on page 92.

Step 5. Check the syntax of the configuration file

```
# wlmd -c configfile
```

and fix any errors found in the syntax check

Step 6. Start WLM

```
# wlmd -a configfile
```

Step 7. Run only one application that will have its duration managed (run it by itself so that it has unlimited CPU access)

Step 8. Let the application complete

Once the application finishes, `wlmdurdc` writes its profile value to the file /var/adm/syslog/syslog.log.

Step 9. Repeat Steps 7 and 8 for each application that will have its duration managed

Step 10. Use the profile values from syslog

Modify your discovery command to use the profile values. Also, now that you have the profile values, you can specify reasonable desired durations.

You can generate “percent complete” reports for your applications to feed to `wlmdurdc` at various points during the application’s duration. `wlmdurdc` would then use this information to better tune the CPU allocation for the application’s workload group. For SAS jobs, you can instrument the jobs with the macro `hp_wlmtk_goals_report` to report “percent complete”. For more information on this macro, see hp_wlmtk_goals_report(1M).

For other applications, implement the reporting represented by the following pseudo-code, either inside the application or as an external utility:

```
PID=process_id # Get the application's PID

# wlmdurdc creates this file and checks it for data
file=/var/opt/wlm/tk_config/wlmdurdc.met.PID

point_in_dur=0 # Number points in the application's duration
               # at which "percent complete" data is reported
percent=pct_complete # Indicate the percent completed

if(file exists){ # Write the data to the file
   write(file, "%d %d %d\n", PID, point_in_dur, percent)
}
```

`wlmdurdc` automatically creates the file wlmdurdc.met.*PID* and uses any data present in it.

For an example of how to set up your discovery command, see the "Example discovery command" on page 96.

Step 11. Start the applications in workload groups

NOTE You must start your applications in the correct groups; otherwise, WLM controls a group's CPU resources based on duration management data for an application not in the group.

Next time you need to run the applications with duration management in effect, start each of them with the `prmrun` command. To place applications in the `GroupA` workload group, use the command:

```
% prmrun -g GroupA app1 app1_arguments
```

Similarly, to place applications in the `GroupB` workload group, use the command:

```
% prmrun -g GroupB app2 app2_arguments
```

Example WLM configuration file for duration management from steps above

This first configuration brings together all the configuration file fragments from the section “Completing an application within a certain duration (duration management)” on page 87.

```
prm {
   groups = GroupA : 2, GroupB : 3;
}

slo app1_slo {
   pri = 1;
   mincpu = 5;
   maxcpu = 75;
   entity = PRM group GroupA;
   goal = metric app1_met > 0;
}

slo app2_slo {
   pri = 1;
   mincpu = 10;
   maxcpu = 60;
   entity = PRM group GroupB;
   goal = metric app2_met > 0;
}

tune app1_met {
   coll_argv = wlmrcvdc wlmdurdc discovery;
}

tune app2_met {
   coll_argv = wlmrcvdc wlmdurdc discovery;
}
```

Example WLM configuration file for duration management

The following configuration is from /opt/wlm/toolkits/duration/config/duration.wlm. It shows how to use the example discovery command (/opt/wlm/toolkits/duration/examples/duration.sh) given in the section “Example discovery command” on page 96.

```
# Name:
#       duration.wlm
#
# Version information:
#
#       (C) Copyright 2001-2005 Hewlett-Packard Development Company, L.P.
#
#       $Revision: 1.14 $
#
# Caveats:
#       DO NOT MODIFY this file in its /opt/wlm/examples/wlmconf location!
#       Make modifications to a copy and place that copy outside the
#       /opt/wlm/ directory, as items below /opt/wlm will be replaced
#       or modified by future HP-UX WLM product updates.
#
# Purpose:
#       Demonstrate the use of wlmdurdc, the WLM duration data collector.
#
#       A WLM workload group is needed to handle a special processing
#       requirement from the SAS development group. This requirement is
#       for two processes must complete at about the same time. The
#       programs are timed. The shorter running program is configured to
#       run in the special workload group and to run for a longer amount of
#       time to match the duration of the other program.
#
# Components:
#       The glance toolkit is used. See glance_app(1M) for more
#       information on the glance data collectors.
#
# Allocation change frequency:
#       Because these jobs are relatively short in nature, the default
#       wlm_interval (60 seconds) is too long. We change the interval to 5
#       seconds to minimize the time before WLM adjusts the CPU
#       allocations.
```

```
#------------------------------------------------------------------------
#
# prm structure
#       Create workload groups. We will be actively managing one workload,
#       duration_1, and leaving the rest of the CPU resources to the
#       workload group OTHERS.
#
#       Also, we need a method for getting the SAS development group's
#       special processes into the duration_1 group. Two methods are
#       discussed below in the prm structure.
#
#       See wlmconf(4) for complete HP-UX WLM configuration information.

prm {
    groups =   OTHERS : 1,
               duration_1 : 2;

    # Create a user entry for each individual who is going to be running
    # the application. The example below shows an entry for the user
    # jdoe. If you have a netgroup for the SAS development group, say
    # "SAS_dev_netgroup", you can give everyone in the netgroup access to
    # the duration_1 group with the following users statement. User jdoe or
    # members of this netgroup would then start their processes with the
    # prmrun command:
    #
    #   % prmrun -g duration_1 <app_name> <app_arguments>

#    users = jdoe : duration_1, +SAS_dev_netgroup : duration_1;
    # Line is commented out because the undefined user and netgroup
    # would result in errors if this file were checked by wlmd.
}

# Set the wlm_interval value shorter than the default value of 60 (seconds)
# if the condition metric duration_1_procs_active needs to be updated more
# than once a minute. Here, the interval is set to 5 seconds. A shorter
# wlm_interval allows the SLO to react more quickly.

tune {
        wlm_interval = 5;
}
```

```
# Use a tune structure with glance_prm to determine the number of active
# processes in the duration_1 group. This tune structure generates the
# metric duration_1_procs_active, which is used to make the SLO below
# inactive when no processes are present in the duration_1 workload group.

tune duration_1_procs_active {
        coll_argv = wlmrcvdc glance_prm -i 2 APP_ACTIVE_PROC duration_1;
}

# Configure wlmdurdc to launch the example discovery script
# (/opt/wlm/toolkits/duration/examples/duration.sh). The duration.sh script
# takes two arguments: a workload group name (duration_1) and a file
# (dur_1_rate) containing application names, profile values, and desired
# durations. This file is created by the user based on values found in
# /var/adm/syslog/syslog.log when using "wlmdurdc -P".

tune dur_1_rate {
    coll_argv = wlmrcvdc /opt/wlm/toolkits/duration/bin/wlmdurdc
                /opt/wlm/toolkits/duration/examples/duration.sh
                duration_1 dur_1_rate;
}

# Next, create the SLO slo_dur_1. This SLO has a priority of 1. It has a
# CPU request range of 1 to 50 (mincpu=1, maxcpu=50). A priority 1 SLO has
# its shares allocated before a lower priority SLO. Thus, WLM will satisfy
# slo_dur_1 at the expense of the group OTHERS. The SLO defines a goal
# that allows its target application's run-rate to be managed by WLM. The
# goal statement is typical for SLOs pertaining to duration management,
# with a goal that the metric be greater than 0.
#
# In addition, slo_dur_1 has a condition metric,
# duration_1_procs_active. This condition metric specifies that at least
# one process must be executing in the group before the SLO becomes active.
#
# This SLO is inactive when there are no processes in its associated
# workload group. Otherwise, the SLO provides enough shares to satisfy the
# desired duration requirements.

slo slo_dur_1 {
    pri = 1;
    entity = PRM group duration_1;
    mincpu = 1;
    maxcpu = 50;
    goal = metric dur_1_rate > 0;
    condition = metric duration_1_procs_active > 0;
}
```

Example discovery command

The discovery command invoked in the previous section is given below. Note that this example is merely a starting point; there are more sophisticated ways to collect this information. Also be aware that there are example discovery commands specifically for SAS users in /opt/wlm/toolkits/sas/examples/.

In essence, this script:

1. Takes a workload group name on its command line
2. References a file where each line specifies:
   - An executable name
   - That executable's profile value
   - That executable's desired duration
3. Makes a list from the file's executable names
4. Makes a list of active processes for the workload group
5. Checks the lists against each other:

   Checks each active process in the workload group to see whether there is data for it in the file from Step 2
6. Determines the process's PID if there is a match in Step 5 else returns to Step 3
7. Prints the process's PID, profile value, and desired duration
8. Exits

The script is:

```
#!/usr/bin/sh
#
# Name:
#       duration.sh
#
# Version information:
#
#       (C) Copyright 2001-2005 Hewlett-Packard Development Company, L.P.
#
#       $RCSfile: duration.sh,v $
#       $Date: 2005/04/19 12:21:13 $
#       $Revision: 1.6 $
#
```

```
# Example discovery script for wlmdurdc.
#
# This script is launched by wlmdurdc to discover a process that is to be
# managed.  It returns a PID, the profile value in CPU seconds,
# and the desired duration in seconds.  This script could be used when
# multiple process will be run in the same workload group at different
# times.
#
#
# The output of duration.sh is the input to wlmdurdc:
#   * Process ID
#   * Profile value (CPU time needed to finish process in integer seconds)
#   * Desired duration (in integer seconds)
#
# This script expects the group and metric names on the command line:

PRM_GROUP=$1
WLM_METRIC=$2

# If you use this script, you must first create a config file.  There needs
# to be a config file for each WLM metric in the WLM configuration file.
# Config file format:
#
# {executable name} {profile value} {desired duration}
#
# The executable name is the base name of the executable.  This can be seen
# in the output of ps -e.  The following example is for the my_process
# executable that has a profile value of 100 CPU seconds and a desired
# duration of 200 seconds.
#
# Example:
#       my_process 100 200
#

if [ "x$PRM_GROUP" = "x" ]; then
    echo "ERROR: WLM workload group needed"
    exit 1
fi

if [ "x$WLM_METRIC" = "x" ]; then
    echo "ERROR: WLM metric name needed"
    exit 1
fi

JOB_CONFIG_FILE="/tmp/profiles/${PRM_GROUP}/${WLM_METRIC}"
```

```
if [ ! -e "$JOB_CONFIG_FILE" ]; then
    echo "ERROR: $JOB_CONFIG_FILE does not exist.  It must be created"
    echo "before this script can be used."
    exit 1
fi


# Fill an array with the first argument of each line in the file (the
# executable process name):

set -A profiled_procs `cat ${JOB_CONFIG_FILE} | awk '{print $1}'`

# forever

while [[ 1 = 1 ]] ; do

    # Create a list of active processes in this group:

    set -A active_procs `ps -R ${PRM_GROUP} | awk '{print $NF}'`;
    let i=0;

    # Walk the list of active processes, looking for a match in config file

    while [[ "x${active_procs[i]}" != "x" ]] ; do

        # We found a process running in the selected PRM

        let j=0;
        while [[ "x${profiled_procs[j]}" != "x" ]] ; do

            # We found an active process that is in our list of profiled jobs

            if [[ "${active_procs[i]}" = "${profiled_procs[j]}" ]] ; then

                # This active process is in my profile list.
                # Read the rest of the line for this particular application

                # active_entry is the entry in the config file we are
                # looking for

                # entry_fields is an array that is filled with the elements
                # of the process entry in the config file
                # entry_fields[0] = executable name
                # entry_fields[1] = profile value (CPU seconds)
                # entry_fields[2] = desired duration (real seconds)
```

```
                set -A entry_fields `cat ${JOB_CONFIG_FILE} | \
                        awk -v active_entry=${active_procs[i]} \
                        '$1~active_entry {print $0}'`

                # Get the actual PID of the process

                pid=`ps -R ${PRM_GROUP} | \
                        awk -v target=${active_procs[i]} \
                        '$NF~target {print $1}'`

                # Send the PID, profile value, and desired duration for the
                # application:

                echo "${pid} ${entry_fields[1]} ${entry_fields[2]}"

                # We've done the job

                exit 0
            fi
            let j=j+1 ;
        done
        let i=i+1;
    done ;

    # Wait a bit before doing this again.

    sleep 1
done
```

Instrumenting SAS jobs for better duration management

DMTK comes with the macro `hp_wlmtk_goals_report`. Use this macro to instrument your SAS jobs for more precise duration management. Place the macro throughout the source code to report "percent complete" to the HP-UX Workload Manager duration management data collector `wlmdurdc`. The syntax is:

`hp_wlmtk_goals_report(`*`percent_complete`*`)`

where *`percent_complete`* is an integer from 1 to 99 inclusive.

Using this macro is optional. If you are satisfied with your current duration management, do not use the `hp_wlmtk_goals_report` macro. If, however, you prefer more precise duration management, use this macro to guide `wlmdurdc` to more accurate management. With accurate input from `hp_wlmtk_goals_report`, `wlmdurdc` can fine-tune its estimates on how much CPU a job needs to finish in the desired duration.

Each invocation of `hp_wlmtk_goals_report` in a SAS job collects the following data:

- *`percent_complete`*, as read from the user-supplied argument

  If the *`percent_complete`* value is invalid (not an integer from 1 to 99 inclusive), the value is logged to syslog but ignored for the purposes of duration management.
- The job's PID

This data is then passed to `wlmdurdc`, which makes the information available in syslog.

NOTE

Each macro invocation is numbered in syslog. This allows you to match the data in the file to the invocation in the SAS job, which is helpful in debugging. For example, assume syslog has an entry including:

```
Percent update: 79, Tag: 5 (pid 20179)
```

From this, we know that the 5th invocation (Tag: 5) of `hp_wlmtk_goals_report` in the SAS job running with PID 20179 claims that the job is 79% complete. If the job is not actually 79% at this point, adjust the value in the `hp_wlmtk_goals_report` invocation.

Example instrumentation

The following example code fragment shows how to insert `hp_wlmtk_goals_report` calls into your SAS job. After the first run statement, we know the job is 6% done. We also know that after the second run statement, it is 53% done. Placing the macro in the code, we provide that information to `wlmdurdc`.

```
/*  Fit a stepwise linear regression.        */
proc reg data=sample;
  model netresp= &indepent /selection=STEPWISE sle=.15 sls=.05 vif collin stb;
run;

%hp_wlmtk_goals_report(6);

/*  Fit a stepwise logistic regression.      */
proc logistic data=sample;
  model netresp= &indepent / maxiter=200 risklimits selection=S
  lackfit;
run;

%hp_wlmtk_goals_report(53);

/*  Fit a stepwise logistic regression.      */
proc logistic data=sample;
  model netresp= &indepent / maxiter=200 risklimits selection=S
  lackfit;
run;
```

Completing an urgent job as quickly as possible (express lanes)

NOTE Although this functionality is part of the standard WLM product, it is of special interest in a Base SAS environment when a SAS report is needed urgently.

WLM allows you to create an express lane for quicker completion of your most critical jobs. These jobs may run on a regular schedule or on an as-needed basis. Nevertheless, through the use of a specially defined WLM workload group for these jobs, you can be sure the jobs execute quickly.

Setting up an express lane

Consider the following scenario:

Friday morning, the CEO learns that she must give a presentation to market analysts at 2 pm; she needs to present a current financial report. Unfortunately, it is also the end of the month, so accounting must have a completed inventory at the start of the next business day. Furthermore, the development team has just updated a report and must run its scheduled regression test. These types of jobs are all priority-based:

- The CEO's job is an unscheduled, high-priority task
- The accounting program is a time-critical application
- The development team's job is scheduled and discretionary

These processes all have specific priorities and must all be asynchronously allocated resources based on the business needs of the company.

With WLM, you can insert a job into a normal day's workload and give it priority access to resources to ensure it completes as quickly as possible. This can be done without stopping normal work, although it does slow down the normal work.

The administrator can create an express lane for unscheduled, high-priority jobs. An express lane is a normal WLM workload group. It differs from other groups in that it does not have automatic job assignments through the WLM configuration file. It is assigned jobs manually using the `prmrun` command.

Be sure to configure the express lane workload group with enough CPU resources that the job has the needed resources to complete quickly.

The following procedure gives the general steps in creating an express lane. For example WLM configurations with express lanes, see the sections following the procedure.

To create an express lane group and run a job in it:

Step 1. Add an express lane workload group to your current WLM configuration. If you do not already have a WLM configuration, use the following to create an express lane that has priority access to 60% of the system CPU:

```
prm {
   groups = express_lane : 2; # Define the express lane
}

slo slo_express_lane {
   pri = 1;
   entity = PRM group express_lane;
   cpushares = 60 total; # Grant the express lane CPU
   condition = metric express_lane_procs_active > 0;
### Syntax is WLM Version A.01.02 or later ###
}
```

The `condition` statement above enables the SLO only when there are one or more active processes in the `express_lane` workload group. The required metric `express_lane_procs_active` is created using the `glance_prm` command shown in the next step.

NOTE Configure the express lane group with no more CPU than is necessary; otherwise, the CPU resources it does not use are wasted.

Step 2. Create the metric used in the `condition` statement by using the following `tune` structure named for the metric:

```
tune express_lane_procs_active {
   coll_argv = wlmrcvdc glance_prm -i 2 APP_ACTIVE_PROC express_lane;
}
```

The `wlmrcvdc` and `glance_prm` commands come with WLM. The `glance_prm` command uses HP's GlancePlus product to determine the number of active processes in the workload group `express_lane`.

Step 3. Check the syntax of the configuration file—named *`configfile`* below:

```
# wlmd -c configfile
```

Step 4. Fix any errors found in the syntax check.

Step 5. Activate the configuration:

```
# wlmd -a configfile
```

Step 6. Start the high-priority job in the `express_lane` workload group:

```
# prmrun -g express_lane critical_job
```

where *`critical_job`* is a command for launching the job.

You can also move an existing high-priority job to the `express_lane` workload group. To accomplish this, use the `prmmove` command:

```
# prmmove express_lane -p PID
```

where *`PID`* is the process ID of *`critical_job`*.

For more information on these commands, see the prmrun(1) and prmmove(1) man pages.

Example express lane: fixed allocation

This example (/opt/wlm/toolkits/duration/config/expressconf.wlm) shows an express lane SLO that has a fixed CPU allocation for our workload group.

```
# Name:
#       expressconf.wlm
#
# Version information:
#
#       (C) Copyright 2001-2005 Hewlett-Packard Development Company, L.P.
#
#       $Revision: 1.9 $
#
# Caveats:
#       DO NOT MODIFY this file in its /opt/wlm/examples/wlmconf location!
#       Make modifications to a copy and place that copy outside the
#       /opt/wlm/ directory, as items below /opt/wlm will be replaced
#       or modified by future HP-UX WLM product updates.
#
# Purpose:
#       Demonstrate an express lane. Express lanes allow applications to
#       finish more quickly.
```

```
#       A WLM workload group is needed to handle special queries from the
#       marketing department. These special requests are very important
#       and must be completed as quickly as possible without regard to
#       the impact on other services. From past experience, just running
#       these special processes in a workload group containing a fixed
#       allocation gives the best response characteristics. This workload
#       group only receives CPU when it has active processes.
#
# Dependencies:
#       This example was designed to run with HP-UX WLM version A.01.02
#       or later.  It uses the cpushares keyword introduced in A.01.02
#       and is, consequently, incompatible with earlier versions of
#       HP-UX WLM.
#
# Components:
#       The glance toolkit is used. See glance_app(1M) for more
#       information on the glance data collectors.
#
# Allocation change frequency:
#       Because these jobs are relatively short in nature, the default
#       wlm_interval metric (60 seconds) is too long. We change the
#       interval to 5 seconds to minimize the time before WLM adjusts the
#       CPU allocations.
#
#-------------------------------------------------------------------------------
#
# prm structure
#       Create workload groups. We will be actively managing one workload,
#       express_lane, and leaving the rest of the CPU resources to the
#       workload group OTHERS.
#
#       Also, we need a method for getting marketing's special processes
#       into the express_lane group.
#
#       See wlmconf(4) for complete HP-UX WLM configuration information.

prm {
    groups =   OTHERS : 1,
               express_lane : 2;

    # Create a user entry for each individual who is going to be running
    # the application. The example below shows an entry for the user
    # jdoe. If you have a netgroup for marketing, say "marketing_netgroup",
    # you can give everyone in the netgroup access to the express_lane
    # group with the following users statement. User jdoe or members of
    # this netgroup would then start the marketing processes in the
```

```
    # express_lane workload group with the prmrun command:
    #
    #        % prmrun -g express_lane <app_name> <app_arguments>

#    users = jdoe : express_lane, +marketing_netgroup : express_lane;
    # Line is commented out because the undefined user and netgroup
    # would result in an error if this file were checked by wlmd.
}

# Set the wlm_interval value shorter than the default value of 60 (seconds)
# if the condition metric express_lane_procs_active needs to be updated
# more than once a minute. Here, the interval is set to 5 seconds.
# A shorter wlm_interval allows the SLO to react more quickly.

tune {
        wlm_interval = 5;
}

# Use a tune structure with glance_prm to determine the number of active
# processes in the express_lane group. This structure generates the metric
# express_lane_procs_active, which is used to make the SLO below inactive
# when no processes are present in the express_lane workload group.

tune express_lane_procs_active {
        coll_argv = wlmrcvdc glance_prm -i 2 APP_ACTIVE_PROC express_lane;
}

# Next, create the SLO slo_express_lane. This SLO has a priority of 1.
# Also, it has a fixed CPU request. A priority 1 SLO has its shares
# allocated before a lower priority SLO. Thus, WLM will satisfy
# slo_express_lane at the expense of the group OTHERS. In addition, the SLO
# has a condition metric, express_lane_procs_active. This condition metric
# specifies that at least one process must be executing in the express_lane
# workload group before the SLO becomes active.
#
# This SLO is inactive when there are no processes in its associated
# group. Otherwise, the SLO provides 45 CPU shares to the group.

slo slo_express_lane {
    pri = 1;
    entity = PRM group express_lane;
    cpushares = 45 total;
    condition = metric express_lane_procs_active > 0;
}
```

Example express lane: shares-per-metric allocation

This example (/opt/wlm/toolkits/duration/config/expressconf_shares.wlm) creates an express lane that receives more CPU with each process that is active in the workload group.

```
# Name:
#       expressconf_shares.wlm
#
# Version information:
#
#       (C) Copyright 2001-2005 Hewlett-Packard Development Company, L.P.
#
#       $Revision: 1.8 $
#
# Caveats:
#       DO NOT MODIFY this file in its /opt/wlm/examples/wlmconf location!
#       Make modifications to a copy and place that copy outside the
#       /opt/wlm/ directory, as items below /opt/wlm will be replaced
#       or modified by future HP-UX WLM product updates.
#
# Purpose:
#       Demonstrate an express lane that has a shares-per-metric
#       goal. Express lanes allow applications to finish more quickly.
#
#       A WLM workload group is needed to handle special queries from the
#       marketing department. These special requests are very important
#       and must be completed as quickly as possible without regard to the
#       impact on the other services. From past experience, a base
#       allocation of 5 CPU shares gives a good response for a single
#       random query. Also, for each additional job added to the group, an
#       additional 5 CPU shares are added to the group's CPU allocation to
#       maintain performance. This workload group only receives CPU when
#       it has active processes.
#
# Components:
#       The glance toolkit is used. See glance_app(1M) for more
#       information on the glance data collectors.
#
# Allocation change frequency:
#       Because these jobs are relatively short in nature, the default
#       wlm_interval metric (60 seconds) is too long. We change the
#       interval to 5 seconds to minimize the time before WLM adjusts the
#       CPU allocations.
```

```
# Dependencies:
#       This example was designed to run with HP-UX WLM version A.01.02 or
#       later. It uses the cpushares keyword introduced in A.01.02, so
#       is incompatible with earlier versions of WLM.
#
#-------------------------------------------------------------------------------
#
# prm structure
#       Create workload groups. We will be actively managing one workload,
#       express_lane, and leaving the rest of the CPU resources to the
#       workload group OTHERS.
#
#       Also, we need a method for getting marketing's special processes
#       into the express_lane group.
#
#       See wlmconf(4) for complete HP-UX WLM configuration information.

prm {
    groups =   OTHERS       : 1,
               express_lane : 2;

    # Create a user entry for each individual who is going to be running
    # the application. The example below shows an entry for the user
    # jdoe. If you have a netgroup for marketing, say "marketing_netgroup",
    # you can give everyone in the netgroup access to the express_lane group
    # with the following users statement. User jdoe or members of this
    # netgroup would then start the marketing processes in the express_lane
    # workload group with the prmrun command:
    #
    #   % prmrun -g express_lane <app_name> <app_arguments>

#    users = jdoe : express_lane, +marketing_netgroup : express_lane;
    # Line is commented out because the undefined user and netgroup would
    # result in errors if this file were checked by wlmd.

}

# Set the wlm_interval value shorter than the default value of 60 (seconds)
# if the condition metric express_lane_procs_active needs to be updated
# more than once a minute. Here, the interval is set to 5 seconds. A
# shorter wlm_interval allows the SLO to react more quickly.

tune {
    wlm_interval = 5;
}
```

```
# Use a tune structure with glance_prm to determine the number of active
# processes in the express_lane group. This structure generates the
# metric express_lane_procs_active, which is used to make the SLO below
# inactive when no processes are present in the express_lane workload
# group.

tune express_lane_procs_active {
        coll_argv = wlmrcvdc glance_prm -i 2 APP_ACTIVE_PROC express_lane;
}

# Next, we create the SLO slo_express_lane. This SLO has a priority of 1.
# Also, it has a CPU request range of 5 to 45 (mincpu=5, maxcpu=45).
# A priority 1 SLO will have its shares allocated before a lower priority
# SLO. Thus, WLM will satisfy slo_express_lane at the expense of the group
# OTHERS. In addition, slo_express_lane has a condition metric,
# express_lane_procs_active. This condition metric specifies that at least
# one process must be executing in the express_lane workload group before
# the SLO becomes active.
#
# The cpushares statement requests shares based on the number of processes
# in the workload.
#
# This SLO is inactive when there are no processes in its associated
# workload group. Otherwise, the SLO provides 5 additional CPU shares to
# the group for each of its processes.

slo slo_express_lane {
    pri = 1;
    entity = PRM group express_lane;
    mincpu = 5;
    maxcpu = 45;
    cpushares = 5 more per metric express_lane_procs_active;
    condition = metric express_lane_procs_active > 0;
}
```

Example express lane: CPU utilization goal

The last example (/opt/wlm/toolkits/duration/config/expressconf_usage.wlm) shows an express lane based on an SLO with a CPU usage, or utilization, 7goal. Recall that a usage goal tries to keep the workload's CPU utilization of its allocated CPU within a certain range—by changing the amount of CPU allocated.

```
# Name:
#       expressconf_usage.wlm
#
# Version information:
#
#       (C) Copyright 2001-2005 Hewlett-Packard Development Company, L.P.
#
#       $Revision: 1.9 $
#
# Caveats:
#       DO NOT MODIFY this file in its /opt/wlm/examples/wlmconf location!
#       Make modifications to a copy and place that copy outside the
#       /opt/wlm/ directory, as items below /opt/wlm will be replaced
#       or modified by future HP-UX WLM product updates.
#
# Purpose:
#       Demonstrate an express lane that has a usage goal. Express lanes
#       allow applications to finish more quickly.
#
#       A WLM workload group is needed to handle special queries from the
#       marketing department. These special requests are very important
#       and must be completed as quickly as possible. However, care must be
#       taken with regard to the impact on the other services. From past
#       experience, running these special processes in a group with CPU
#       allocations based on the actual resource usage give the best
#       response characteristics. This workload group only receives CPU
#       when it has active processes.
#
# Components:
#       The glance toolkit is used. See glance_app(1M) for more
#       information on the glance data collectors.
#
# Allocation change frequency:
#       Because these jobs are relatively short in nature, the default
#       wlm_interval (60 seconds) is too long. We change the interval to 5
#       seconds to minimize the time before WLM adjusts the CPU allocations.
#
```

```
#-------------------------------------------------------------------------
#
# prm structure
#       Create workload groups. We will be actively managing one workload,
#       express_lane, and leaving the rest of the CPU resources to workload
#       group OTHERS.
#
#       Also, we need a method for getting marketing's special processes
#       into the express_lane group.
#
#       See wlmconf(4) for complete HP-UX WLM configuration information.

prm {
    groups =   OTHERS       : 1,
               express_lane : 2;

    # Create a user entry for each individual who is going to be running
    # the application. The example below shows an entry for the user
    # jdoe. If you have a netgroup for marketing, say "marketing_netgroup",
    # you can give everyone in the netgroup access to the express_lane
    # group with the following users statement. User jdoe and members of
    # this netgroup would then start the marketing processes in the
    # express_lane workload group with the prmrun command:
    #
    #   % prmrun -g express_lane <app_name> <app_arguments>

#    users = jdoe : express_lane, +marketing_netgroup : express_lane;
    # Line is commented out because the undefined user and netgroup would
    # result in errors if this file were checked by wlmd.
}

# Set the wlm_interval value shorter than the default value of 60 (seconds)
# if the condition metric express_lane_procs_active needs to be updated
# more than once a minute. Here, the interval is set to 5 seconds. A
# shorter wlm_interval allows the SLO to react more quickly.

tune {
    wlm_interval = 5;
}
```

```
# Use a tune structure with glance_prm to determine the number of active
# processes in the express_lane group. This tune structure generates the
# metric express_lane_procs_active, which is used to make the SLO below
# inactive when no processes are present in the express_lane workload group.

tune express_lane_procs_active {
    coll_argv = wlmrcvdc glance_prm -i 2 APP_ACTIVE_PROC express_lane;
}

# Next, create the SLO slo_express_lane. This SLO has a usage, or
# utilization, goal. With this type of goal, WLM adjusts the workload's
# allocation so that its utilization of that allocation falls within a
# certain range. For example, if a workload is using 20 CPU shares and has
# a 50 share allocation, its utilization is 20/50 or 40%. By default, WLM
# attempts to keep a workload's utilization between 50% and 75%. Reducing
# the allocation to 40 shares, utilization rises to 50%.
#
# This SLO is inactive when there are no processes in its associated
# workload group. Otherwise, the SLO provides additional CPU shares to the
# group as needed.

slo slo_express_lane {
    pri = 1;
    entity = PRM group express_lane;
    mincpu = 5;
    maxcpu = 45;
    goal = usage _CPU;
    condition = metric express_lane_procs_active > 0;
}
```

Command and macro reference

DMTK / SASTK consist of:

- `wlmdurdc` duration data collector
- `hp_wlmtk_goals_report` SAS macro

These item are described below.

wlmdurdc

`wlmdurdc` is a data collector for Workload Manager. It provides duration management for wlmd(1M).

`wlmdurdc` requires three values on standard out in the following order from the user-supplied discovery command:

- PID of the process needing duration management
- Profile value

  This value is the CPU time in integer seconds needed by the process to finish when it has unlimited CPU resources.
- Desired duration of the managed process in whole seconds (integer seconds)

You can develop the discovery command in any fashion you desire as long as it provides the values above on standard out. For an example discovery command, see the “Example discovery command” on page 96.

`wlmdurdc` uses these values to determine how many CPU cycles the process is currently using, how many cycles it needs to finish, and by when the process should get all those cycles. Based on these values, `wlmdurdc` sends a request to WLM for more CPU for the process’s workload group if the process needs more resources to finish within the timeframe. Similarly, a request for fewer CPU resources is sent to WLM if the process is finishing too quickly.

`wlmdurdc` works as described below:

Step 1. Launches discovery commands

`wlmdurdc` launches all the discovery commands using `exec()`.

Step 2. Gets discovery data

`wlmdurdc` waits for each discovery command to output data.

Step 3. Locates target application

`wlmdurdc` locates the process matching the PID from the discovery command output. If no such PID is found, `wlmdurdc` returns to Step 1.

Step 4. Sends request to WLM

`wlmdurdc` uses the data from each discovery command to calculate a request to WLM for an adjusted CPU allocation.

Step 5. Repeats

After sending data to WLM, `wlmdurdc` begins the process again.

For more information on this utility—including its options, see the wlmdurdc(1M) man page.

hp_wlmtk_goals_report

`hp_wlmtk_goals_report` is a macro. Use this macro to instrument your SAS jobs for more accurate duration management. Place the macro throughout the SAS job source code to report “percent complete” to `wlmdurdc`. The syntax is:

`hp_wlmtk_goals_report(`*`percent_complete`*`)`

where *`percent_complete`* is an integer value ranging from 1 to 99 inclusive.

For more information on this utility, see the hp_wlmtk_goals_report(1M) man page.

Frequently asked questions

Here are various questions that may occur in using DMTK / SASTK.

I want a process to finish in five minutes. How do I set that up?

Your discovery command feeds three values into `wlmdurdc`: a process PID, a profile value for that process, and a desired duration. You want to set the desired duration value to five minutes. This value must be in seconds, so set it to 300. Also, be sure that the `wlm_interval` keyword is set to a value less than 1/5 of your duration, or 60 seconds. (Its default value is 60 seconds.)

For more information on discovery commands, see “Completing an application within a certain duration (duration management)” on page 87 and “Example discovery command” on page 96.

What’s the shortest job that can be managed?

For the purposes of duration management, your jobs should last at least four or five times the `wlm_interval` value. This interval is 60 seconds by default and should not be set less than five seconds.

Troubleshooting

If you encounter problems, check the syslog (/var/adm/syslog/syslog.log) to determine what values the discovery command is sending to `wlmdurdc`, as well as the values `wlmdurdc` sends to WLM.

For WLM messages in general, see the message log at /var/opt/wlm/msglog.

Related information

For information about DMTK, SASTK, and their related products, consult the following documentation:

- DMTK / SASTK
  - wlmdurdc(1M) man page
  - hp_wlmtk_goals_report(1M) man page
  - wlmtk(5) man page (Workload Manager Toolkits overview)
  - Workload Manager Toolkits A.01.08 Release Notes /opt/wlm/toolkits/doc/Rel_Notes
- HP-UX Workload Manager
  - wlm(5) man page
  - wlmd(1M) man page
  - wlmconf(4) man page
  - wlmrcvdc(1M) man page
  - White paper on writing data collectors (also known as “performance monitors”) /opt/wlm/share/doc/howto/perfmon.html
  - *HP-UX Workload Manager User’s Guide*: http://docs.hp.com/hpux/netsys /opt/wlm/share/doc/WLMug.pdf
  - Workload Management homepage: http://www.hp.com/go/wlm

7 HP-UX WLM SNMP Toolkit

WLM provides integration with the HP-UX SNMP agent and other SNMP agents through the WLM SNMP Toolkit (SNMPTK). This toolkit is part of the freely available WLM Toolkits, or WLMTK.

SNMPTK provides a WLM data collector called `snmpdc`, which fetches values from an SNMP agent so you can use them as metrics in your WLM configuration.

For more information on WLM, see the *HP-UX Workload Manager User's Guide* or the wlm(5) man page.

Supported installations

The tools discussed in this document work with:

- HP-UX WLM Version A.02.02 on HP-UX 11i v1 (B.11.11) and HP-UX 11i v2 (B.11.23)
- HP-UX WLM Version A.02.01.01 on HP-UX 11i v2 (B.11.23)
- HP-UX WLM Version A.02.01 on HP-UX 11i v1 (B.11.11)
- HP-UX WLM Version A.01.02 (which is the first version to include the `cpushares` keyword) on HP-UX 11i v1

Audience for the SNMPTK documentation

This SNMPTK information is intended for anyone implementing WLM and would like to use SNMP data as metrics in WLM configurations.

Why use SNMPTK?

SNMPTK allows easy access to SNMP data that you can use in your WLM configuration to:

- Drive SLO goals
- Set up shares-per-metric allocations
- Enable and disable SLOs

Sources of data include:

- PRM data, available starting at the OID `.1.3.6.1.4.1.11.5.4.2.1` (hp.hpSysMgt.hpUXSysMgt.hpPRM.prmReadOnly)
- Pay Per Use data, available starting at the OID `.1.3.6.1.4.1.11.11.1` (hp.hpUtility.cpuppu)

Tools in SNMPTK

This toolkit includes the following tool:

`snmpdc`

`snmpdc` is a data collector for WLM. It fetches values from an SNMP agent for use as metrics in your WLM configurations.

Where do I get SNMPTK?

SNMPTK is installed at /opt/wlm/toolkits/snmp/.

If it is not already installed on your system, you can download it free of charge from http://www.hp.com/go/wlm.

What comes with SNMPTK?

SNMPTK comes with the `snmpdc` utility, its man page, and an example configuration file. Table 7-1 lists many of the included files.

Table 7-1 PPUTK directories and files

Directory/file	Description
/opt/wlm/share/man/man5.Z/wlmtk.5	Man page that provides an overview of the Workload Manager Toolkits
/opt/wlm/toolkits/snmp/	Directory for SNMPTK and all its files
/opt/wlm/toolkits/doc/WLMTKug.pdf	WLMTK User's Guide (PDF)
/opt/wlm/toolkits/doc/WLMTKug.ps	WLMTK User's Guide (PostScript)
/opt/wlm/share/man/man1m.Z/snmpdc.1m	Man page for the WLM SNMP data collector
/opt/wlm/toolkits/README	Readme for all the WLM Toolkits
/opt/wlm/toolkits/snmp/README	Readme for the WLM SNMP Toolkit
/opt/wlm/toolkits/snmp/bin/snmpdc	The WLM SNMP data collector, which provides data to WLM
/opt/wlm/toolkits/snmp/config/minimalist.wlm	Example configuration file*

* For a description of the example configuration file, see "Example WLM configuration" on page 122.

How do I install SNMPTK?

Use the SD-UX `swinstall` command to install SNMPTK, which is part of the WLM Toolkits product (product number T1302AA).

The SNMP toolkit is installed at /opt/wlm/toolkits/snmp/. The man pages are installed at /opt/wlm/share/man/.

For more information about installation procedures and related issues, refer to the following documentation:

- *Managing HP-UX Software with SD-UX*
- swinstall (1M) man page

How do I use SNMPTK?

To use SNMPTK:

Step 1. Activate the `snmpdc` data collector

In the WLM configuration file, create a `tune` structure. The name you give this `tune` structure will be used as a metric elsewhere in the configuration.

The following `tune` structure retrieves the SNMP value of the resource given after the `-r` option. The value represents the number of currently active processes on the system.

```
tune active_processes {
    coll_argv = wlmrcvdc
                /opt/wlm/lbin/coll/snmpdc -r .1.3.6.1.4.1.11.2.3.1.4.1.0;
}
```

Step 2. Use the SNMP data item as a metric

In the WLM configuration file, use the metric, in this case active_processes, in an SLO in any of the following statements:

- `goal` statement
- `cpushares` statement
- `condition` or `exception` statement

In the following example, our metric is used in a `condition` statement. This SLO will request 40 CPU shares for the group `g3` when there are fewer than 200 processes currently active on the system.

```
slo low_proc {
    pri       = 1;
    cpushares = 40 total;
    entity    = PRM group g3;
    condition = metric active_processes < 200;
}
```

Step 3. Check the WLM configuration file's syntax:

```
# wlmd -c configfile
```

Fix any errors that are found.

Step 4. Activate the WLM configuration file:

```
# wlmd -a configfile
```

Example WLM configuration

This section includes the example configuration /opt/wlm/toolkits/snmp/config/minimalist.wlm.

This configuration shows how to use an SNMP value to enable and disable SLOs.

```
# Name:
#       minimalist.wlm
#
# Version information:
#
#       (C) Copyright 2002-2005 Hewlett-Packard Development Company, L.P.
#
#       $Revision: 1.10 $
#
# Caveats:
#       DO NOT MODIFY this script in its /opt/wlm/toolkits location!
#       Make modifications to a copy and place that copy outside the
#       /opt/wlm/ directory, as items below /opt/wlm will be replaced
#       or modified by future HP-UX WLM product updates.
#
# Purpose:
#       Demonstrates how to configure snmpdc to retrieve data from an
#       SNMP agent and use it as input to drive a WLM SLO.
#
# Dependencies:
#       This example was designed to run with HP-UX WLM version A.01.02
#       or later.  It uses the cpushares keyword introduced in A.01.02
#       and is, consequently, incompatible with earlier versions of
#       HP-UX WLM.
#
```

```
# Workload group definitions
#
# Change the application record to include the names of the binaries that
# make up the workload.  When WLM starts, it will automatically move
# those processes into the specified group.  See the wlmconf(4) man page
# for more details on application records.
#
prm {
    groups = OTHERS   : 1,
             workload : 2;

    apps = workload : "/bin/tar";
}

#
# This SLO sets a fixed allocation of 40% when there are fewer
# than 200 processes running on the system.
#
slo s_workload_low_proc {
    pri       = 1;
    cpushares = 40 total;
    entity    = PRM group workload;
    condition = metric m_workload < 200;
}

#
# This SLO sets a fixed allocation of 20% when there are 200
# or more processes running on the system.
#
slo s_workload_high_proc {
    pri       = 1;
    cpushares = 20 total;
    entity    = PRM group workload;
    condition = !(metric m_workload < 200);
}

#
# Use snmpdc to fetch the metric values from an SNMP agent.
#
# The resource:
#
#       .1.3.6.1.4.1.11.2.3.1.4.1.0
#
# is the numeric representation of:
#
#       hp.nm.system.general.processes.processNum.0
```

```
#
# It reports the number of currently active processes.
#
tune m_workload {
    coll_argv = wlmrcvdc
                /opt/wlm/lbin/coll/snmpdc -r .1.3.6.1.4.1.11.2.3.1.4.1.0;
}
```

Command reference

SNMPTK consists of the `snmpdc` data collector.

The data collector is described below.

snmpdc

`snmpdc` is a data collector for WLM. It is specified in the WLM configuration file. The WLM daemon `wlmd` invokes `snmpdc` at startup.

For more information on this utility—including its options, see the utilitydc(1M) man page.

Related information

If you would like to learn more about SNMPTK, see:

- wlmtk(5) man page
- snmpdc(1M) man page
- HP-UX Workload Manager Toolkits A.01.08 Release Notes /opt/wlm/toolkits/doc/Rel_Notes