Using WLM: A quick reference

Executive summary

Traditional IT environments are often silos where both technology and human resources are aligned around an application or business function. Capacity is fixed, resources are over-provisioned to meet peak demand, and systems are complex and difficult to change. Costs are based on owning and operating the entire vertical infrastructure—even when underutilized.

Resource optimization is one of the goals of HP's Adaptive Enterprise strategy—a strategy for helping customers to synchronize business and IT to capitalize on change. Virtualization is a cornerstone of HP's approach to helping customers realize the promise of becoming an adaptive enterprise. It is an approach to IT that pools and shares resources so utilization is optimized and supply automatically meets demand.

HP-UX Workload Manager (WLM) is an important part of virtualization. This paper presents an overview of the techniques and tools available for using WLM version A.02.02 on a PA-RISC server running HP-UX 11.0 or HP-UX 11i V1.0 (B.11.11) or on an Itanium®-based server running HP-UX 11i V2.0 (B.11.23). Readers are assumed to have a basic understanding of WLM terminology and concepts, as well as WLM configuration file syntax.

The paper first gives an overview of a WLM session. Then, it provides background information on various ways to use WLM, including how to complete several common WLM tasks. Lastly, it discusses how to monitor WLM and its effects on your workloads.

If you prefer to configure WLM using a graphical wizard, see the paper "Getting started with HP-UX Workload Manager (basic configuration and monitoring)" available from the "information library" page on http://www.hp.com/go/wlm.

Show me WLM in action

This section provides a quick overview of various commands associated with using WLM. It takes advantage of some of the configuration files and scripts that are used in the chapter "Learning WLM by example" in the HP-UX Workload Manager User's Guide. These files are in the directory /opt/wlm/examples/userguide/ as well as on http://www.hp.com/go/wlm.

To become familiar with WLM, how it works, and some related commands, complete the following procedure:

1. Log in as root
2. Start WLM with an example configuration file:

   ```
   # /opt/wlm/bin/wlmd -a \
   /opt/wlm/examples/userguide/multiple_groups.wlm
   ```

   The file multiple_groups.wlm is shown below. This configuration:

   a. Defines two workload groups: `g2` and `g3`.

   b. Assigns applications (in this case, perl programs) to the groups. (With shell/perl programs, give the full path of the shell or perl followed by the name of the program.)

   c. Sets bounds on CPU usage. The number of CPU shares for the workload groups can never go below the `gmincpu` or above the `gmaxcpu` values. These values take precedence over the minimum and maximum values that you can optionally set in the `slo` structures.

   d. Defines an SLO (service-level objective) for `g2`. The SLO is priority 1 and requests 15 CPU shares for `g2`.

   e. Defines a priority 1 SLO for `g3` that requests 20 CPU shares.

```
# Name:
#       multiple_groups.wlm
#
# Version information:
#
#       $Revision: 1.5 $
#
# This file is based on the file initial.wlm. Comments below indicate
# changes in this file from the initial.wlm file.
#
# 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.
#
# Requirements:
#       To ensure WLM places the perl scripts below in their assigned
#       workload groups, add "/opt/perl/bin/perl" (without the quotes) to
#       the file /opt/prm/shells.


prm {
    groups = g2 : 2,
             g3 : 3; # Added group g3

                     # Changed loop.pl to loop2.pl
                     # Placed loop3.pl in the new group
    apps = g2 : /opt/perl/bin/perl loop2.pl,
           g3 : /opt/perl/bin/perl loop3.pl;

                     # Added gmincpu/gmaxcpu values for group g3
    gmincpu = g2 : 5, g3 : 5;
    gmaxcpu = g2 : 30, g3 : 60;
}
slo test2 { # Changed name of SLO
    pri = 1;
    cpushares = 15 total;
    entity = PRM group g2;
}

slo test3 { # Defined an SLO for the new workload group
    pri = 1;
    cpushares = 20 total;
    entity = PRM group g3;
}
```

3. See what messages a WLM startup produces. Start another session to view the WLM message log:

```
# tail -f /var/opt/wlm/msglog
```

```
08/29/02 08:35:23 [I] (p6128) wlmd initial command line:
08/29/02 08:35:23 [I] (p6128)   argv[0]=/opt/wlm/bin/wlmd
08/29/02 08:35:23 [I] (p6128)   argv[1]=-a
08/29/02 08:35:23 [I] (p6128)   argv[2]=/opt/wlm/examples/userguide/multiple_gro
ups.wlm
08/29/02 08:35:23 [I] (p6128) what: @(#)HP-UX WLM A.02.00 (2002_03_21_17_04_11)
hpux_11.00
08/29/02 08:35:23 [I] (p6128) dir: @(#) /opt/wlm
08/29/02 08:35:23 [I] (p6128) SLO file path: /opt/wlm/examples/userguide/multipl
e_groups.wlm
08/29/02 08:35:26 [I] (p6128) wlmd 6128 starting
```

The text in the log shows when the WLM daemon `wlmd` started, as well as what arguments it was started with—including the configuration file used.

4. See that the workload groups are in effect

The `prmlist` command shows current configuration information. (This PRM, or Process Resource Manager, command is available because WLM uses PRM to provide some of the WLM functionality.)

```
# /opt/prm/bin/prmlist

PRM configured from file:  /var/opt/wlm/tmp/wmprmBAAa06128
File last modified:        Thu Aug 29 08:35:23 2002

PRM Group                       PRMID    CPU Entitlement
-------------------------------------------------------
OTHERS                              1             65.00%
g2                                  2             15.00%
g3                                  3             20.00%

PRM User                        Initial Group                 Alternate Group(s)
-------------------------------------------------------------------------------
root                            (PRM_SYS)

PRM Application                 Assigned Group                Alternate Name(s)
-------------------------------------------------------------------------------
/opt/perl/bin/perl              g2                            loop2.pl
/opt/perl/bin/perl              g3                            loop3.pl
```

Starting with WLM version A.02.00, a web interface to the `prmlist` command is available. For information, see the wlm_watch(1M) man page.

In addition, starting with WLM version A.02.01, you can use the `wlminfo` command, which shows CPU Shares and utilization (CPU Util) for each workload group.

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

Thu Aug 29 08:36:38 2002

Workload Group   PRMID  CPU Shares  CPU Util  Mem Shares  State
OTHERS               1       65.00      0.00        0.00  ON
g2                   2       15.00      0.00        0.00  ON
g3                   3       20.00      0.00        0.00  ON
```

5. Start the scripts referenced in the configuration file.

   a. WLM checks the files /etc/shells and /opt/prm/shells to ensure one of them lists any shell or interpreter, including perl, used in a script. If the shell or interpreter is not in either of those files, WLM ignores its application record (the workload group assignment in an `apps` statement).

      Add the following line to the file /opt/prm/shells so that the application manager can correctly assign the perl programs to workload groups:

      ```
      /opt/perl/bin/perl
      ```

   b. The following scripts produce output so you may want to start them in a new window. Start the two scripts loop2.pl and loop3.pl:

      ```
      # /opt/wlm/examples/userguide/loop2.pl &

      # /opt/wlm/examples/userguide/loop3.pl &
      ```

      These scripts start in the `PRM_SYS` group because you started them as the root user. However, the application manager soon moves them (within 30 seconds) to their assigned groups, `g2` and `g3`. After waiting 30 seconds, run the following `ps` command to see that the processes have been moved to their assigned workload groups:

      ```
      # ps -efP | grep loop
      ```

      The output will include the following items (column headings are included for convenience):

      ```
       PRMID    PID TTY       TIME COMMAND
          g3   6463 ttyp1     1:42 loop3.pl
          g2   6462 ttyp1     1:21 loop2.pl
      ```

6. Manage workload group assignments:

   a. Start a process in the group `g2`:

      ```
      # /opt/prm/bin/prmrun -g g2 \
      /opt/wlm/examples/userguide/loop.pl
      ```

   b. Verify that loop.pl is in `g2` with `ps`:

      ```
      # ps -efP | grep loop
      ```

      The output will confirm the group assignment:

      ```
      g2   6793 ttyp1     0:02 loop.pl
      ```

   c. Move the process to another group.

      Use the PID for loop.pl from the last step to move loop.pl to the group `g3`:

      ```
      # /opt/prm/bin/prmmove g3 -p loop.pl_PID
      ```

      In this case, *`loop.pl_PID`* is 6793.

7. Verify workload group assignments:

   The `ps` command has two options related to WLM.

   `-P`

   Shows PRM IDs (workload group IDs) for each process

   `-R` *`workload_group`*

   Shows `ps` listing for only the processes in *`workload_group`*

   Looking at all the processes, we get output including the items shown below (column headings are included for convenience):

   ```
   # ps -efP | grep loop
   ```

   ```
     PRMID    PID TTY       TIME COMMAND
        g3   6793 ttyp1     1:52 loop.pl
        g3   6463 ttyp1     7:02 loop3.pl
        g2   6462 ttyp1     4:34 loop2.pl
   ```

   Focusing on the group `g3`, we get the following output:

   ```
   # ps -R g3
   ```

   ```
    PID TTY       TIME COMMAND
   6793 ttyp1     1:29 loop.pl
   6463 ttyp1     6:41 loop3.pl
   ```

8. Check CPU usage

   The `wlminfo` command shows CPU usage (utilization) by workload group. The command's output, which may be slightly different on your system, is shown below.

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

   Workload Group   PRMID  CPU Shares  CPU Util  Mem Shares  State
   OTHERS               1       65.00      0.00        0.00  ON
   g2                   2       15.00     14.26        0.00  ON
   g3                   3       20.00     19.00        0.00  ON
   ```

   This output shows that both groups are using CPU up to their allocations. If the allocations were increased, the groups' usage would probably increase to match the new allocations.

9. Stop WLM:

   ```
   # /opt/wlm/bin/wlmd -k
   ```

10. See what messages a WLM shutdown produces:

    Running the `tail` command again

    ```
    # tail -f /var/opt/wlm/msglog
    ```

    you will see messages similar to the following:

    ```
    08/29/02 09:06:55 [I] (p6128) wlmd 6128 shutting down
    08/29/02 09:06:55 [I] (p7235) wlmd terminated (by request)
    ```

11. Stop the loop.pl, loop2.pl, and loop3.pl perl programs.

Where is WLM installed?

The following table shows where WLM and some of its components are installed.

Item	Installation path
WLM	/opt/wlm/
WLM Toolkits	/opt/wlm/toolkits/
Man pages for WLM and its Toolkits	/opt/wlm/share/man/

Installing WLM ensures the product Process Resource Manager, sometimes referred to as PRM, is also installed. This product is installed at /opt/prm/.

Can I see how WLM will perform—without actually affecting my system?

WLM provides a passive mode that allows you to see how WLM will approximately respond to a given configuration—without putting WLM in charge of your system's resources. Using this mode, enabled with the `-p` option to `wlmd`, you can safely gain a better understanding of how various WLM features work. In addition, you can check that your configuration behaves as expected—with minimal effect on the system. For example, with passive mode, you can determine:

- How does a `cpushares` statement work?
- How do goals work? Is my goal set up correctly?
- How might a particular `cntl_convergence_rate` value or the values of other tunables affect allocation change?
- How does a usage goal work?
- Is my global configuration file set up as I wanted? If I used global arbitration on my production system, what might happen to the CPU layouts?
- Is a user's default workload set up as I expected?
- Can a user access a particular workload?
- When an application is run, which workload does it run in?
- Can I run an application in a particular workload?
- Are the alternate names for an application set up correctly?

For more information on how to use WLM's passive mode, as well as explanations of how passive mode does not always represent actual WLM operations, see the section "PASSIVE MODE VERSUS ACTUAL WLM MANAGEMENT" wlm(5) man page.

Activate a configuration in passive mode by logging in as root and running the command:

```
# /opt/wlm/bin/wlmd -p -a config.wlm
```

substituting your configuration file's name for *`config.wlm`*.

The WLM global arbiter, `wlmpard`, which is used in managing SLOs across virtual partitions, also provides a passive mode.

How do I start WLM?

Before starting WLM (activating a configuration), you may want to try the configuration in passive mode, discussed in the previous section. Otherwise, you can activate your configuration by logging in as root and running the command:

```
# /opt/wlm/bin/wlmd -a config.wlm
```

substituting your configuration file's name for *config.wlm*.

When you run the `wlmd -a` command, WLM starts the data collectors you specify in the WLM configuration.

Although data collectors are not necessary in every case, be sure to monitor any data collectors you do have. Because data collection is a critical link in the effective maintenance of your configured service-level objectives, you need to be aware when a collector exits unexpectedly. One method for monitoring collectors is to use `wlminfo slo`.

For information on creating your WLM configuration, see the section "How do I create a configuration file?" on page 3.

WLM automatically logs informational messages to the file /var/opt/wlm/msglog. In addition, WLM can log data that allows you to verify WLM's management as well as to fine-tune your WLM configuration file. To log this data, use the `-l` option. This option causes WLM to log data to /var/opt/wlm/wlmdstats. The following command line starts WLM, logging data for SLOs every third WLM interval:

```
# /opt/wlm/bin/wlmd -a config.wlm -l slo=3
```

For more information on the `-l` option, see the wlmd(1M) man page.

How do I stop WLM?

With WLM running, stop it by logging in as root and running the command:

```
# /opt/wlm/bin/wlmd -k
```

How do I create a configuration file?

The WLM configuration file is simply a text file.

To create your own WLM configuration file, use one or more of the following techniques:

- Determine which example configurations can be useful in your environment and modify them appropriately. For information on example configurations, see "Where can I find example WLM configurations?" on page 13.
- Create a new configuration using:
    - `vi` or any other text editor
    - the WLM configuration wizard, which is discussed in the section "What is the easiest way to configure WLM?" on page 12
    - the WLM graphical user interface, `wlmgui`

  Using the wizard, the configuration syntax is almost entirely hidden. With `wlmgui`, you need to be somewhat familiar with the syntax.
- Enhance your configuration created from any of the above methods by modifying it based on ideas from the following sections:
    - "How do I put an application under WLM control?" on page 15
    - "What are some common WLM tasks?" on page 25

For configuration file syntax, see the wlmconf(4) man page.

What is the easiest way to configure WLM?

The easiest and quickest method to configure WLM is to use the WLM configuration wizard.

NOTE

Set your `DISPLAY` environment variable before starting the wizard.

To start the wizard, run the following command:

```
# /opt/wlm/bin/wlmcw
```

The wizard provides an easy way to create initial WLM configurations. To maintain simplicity, it does not provide all the functionality available in WLM. Also, the wizard does not allow you to edit existing configurations.

Where can I find example WLM configurations?

WLM and its toolkits come with example WLM configuration files. These files are located in the directories indicated in the table below.

For	See example WLM configurations in the directory
Examples showing a range of WLM functionality	/opt/wlm/examples/wlmconf/
Examples used in this paper	/opt/wlm/examples/userguide/
Using WLM with Apache web servers	/opt/wlm/toolkits/apache/config/
Using WLM to manage job duration	/opt/wlm/toolkits/duration/config/
Using WLM with Oracle databases	/opt/wlm/toolkits/oracle/config/
Using WLM with SAS software	/opt/wlm/toolkits/sas/config/
Using WLM with SNMP agents	/opt/wlm/toolkits/snmp/config/
Using WLM with iCOD and Pay Per Use	/opt/wlm/toolkits/utility/config/
Using WLM with BEA WebLogic Server	/opt/wlm/toolkits/weblogic/config/

These configurations are also available on http://www.hp.com/go/wlm, from the "case studies / example configurations" page.

How does WLM control applications?

WLM controls your applications after you:

1. Define workload groups for each workload
2. Place the applications or users that define the workloads into their own workload groups (the workloads' processes share the resources WLM makes available to the workload group)
3. Create one ore more SLOs for each workload group

With your workloads isolated in their own workload groups, WLM manages each group's CPU, real memory, and disk bandwidth resources according to the current configuration. WLM allocates resources to your workload groups by automating features of HP's Process Resource Manager (PRM), HP-UX Processor Sets (PSETs), HP-UX Virtual Partitions (vPars), and node partitions (nPars) that use iCOD.

With WLM's PSET management, you can dedicate whole processors to your worklaods. You can also let WLM adjust the number of processors assigned to a PSET-based workload group in response to SLOs. Similarly, WLM can manage vPars. With one or more workloads running in each vPars, WLM can adjust the number of processors in the vPars based on the requirements of the workload groups in those vPars. With WLM's nPar management, WLM uses iCOD software to deactivate a CPU on one nPar then activate a CPU on another nPar where an SLO is in need of resources.

NOTE
WLM adjusts only a workload group's CPU allocation in response to SLO performance. Thus, WLM's SLO management is most effective for workloads that are CPU-bound.

Each application must go into a workload group. By default, processes run by root go into the `PRM_SYS` group, and processes run by nonroot users go into the `OTHERS` group. However, you can change the workload group in which a particular user's processes run by adding user records to the WLM configuration file. Furthermore, you can specify the workload groups in which processes run by adding application records to your WLM configuration.

Be aware that application records take precedence over user records. So, whenever a user with a user record launches an application, the application starts in the user's initial workload group. However, if the application has an application record, the application will be moved to its assigned workload group within 30 seconds.

In addition, you can alter the workload group of an application using the `prmmove` and `prmrun` utilities, which are discussed below.

How do I put an application under WLM control?

WLM provides several methods to place processes in workload groups. It is important to understand these methods as they form the basis of the workload separation.

First, we define the workload groups for the workloads. The following snippet from a WLM configuration file creates three workload groups: `servers_grp`, `apache_grp`, and `OTHERS`. (The `OTHERS` group is a reserved workload group and must have ID 1. If you do not explicitly create this group, WLM will create it for you.)

```
prm {
    groups = OTHERS : 1,
         servers_grp : 2,
         apache_grp : 3;
}
```

Note that each workload group is given a name and a number. Later sections of the WLM configuration file assign resources to the groups. Processes within the groups then share the resources allocated to that group.

With the workload groups defined, the remainder of this section explores how processes can be placed in the workload groups.

Application records: Workload separation by binary name

One mechanism for separating workloads is the `apps` statement. This statement simply names a particular application binary and the group in which it should be placed. You can specify multiple binary-workload group combinations, separated by commas, in a single `apps` statement.

In the `prm` structure below, the `apps` statement causes the PRM application manager to place any newly running /opt/hpws/apache/bin/httpd executables in the group `apache_grp`.

```
prm {
    groups = OTHERS : 1,
         servers_grp : 2,
         apache_grp : 3;

    apps = apache_grp : /opt/hpws/apache/bin/httpd;
}
```

NOTE on polling: It is important to realize that the process is not placed in the workload group immediately after starting. Rather, the PRM application manager periodically polls for newly started processes that match records in the `apps` statement. Each matched process is placed in its designated workload group.

User records: Workload separation by process owner UID

You can place processes in workload groups according to the UIDs of the process owners. You specify your user-workload group mapping in the `users` statement. Here is an example:

```
prm {
    groups = OTHERS : 1,
             testers : 2,
             coders : 3,
             surfers : 4;

    users = moe : coders surfers,
            curly : coders surfers,
            larry : testers surfers;
}
```

Besides the default `OTHERS` group, this example has three groups of users: `testers`, `coders`, and `surfers`. The user records cause processes started by users moe and curly to be run in group `coders` by default, and user larry's processes to be run in group `testers` by default. Each user is also given permission to run jobs in group `surfers` if they wish, using the `prmrun` or `prmmove` commands mentioned below. Users not belonging to either group are placed in `OTHERS` by default.

As previously noted, application records take precedence over user records.

For more information on users' access to workload groups, see the *HP-UX Workload Manager User's Guide*.

`prmrun`: Starting a process in a workload group

You can explicitly start processes in particular workload groups using the `prmrun` command. Given the `groups` and `users` statements shown above, user larry running the command

```
# my_really_big_job &
```

would cause his job to be run in group `testers`. However, user larry also has permission to run processes in the group `surfers`. Thus, larry can use the `prmrun` command below

```
# /opt/prm/bin/prmrun -g surfers my_really_big_job &
```

to run his process in the group `surfers`.

`prmmove`: Moving an existing process to a workload group

Use the `prmmove` command to move existing processes to a different workload group. If larry from the above example has a job running with process ID (PID) 4065 in the group `testers`, he could move that process to group `surfers` by running the command:

```
# /opt/prm/bin/prmmove surfers -p 4065
```

Default: Inheriting workload group of parent process

If a process is not named in an `apps` statement or a `users` statement, or has not been started with `prmrun` or moved with `prmmove`, it simply starts and runs in the same group as its parent process. So for a setup like this

```
prm {
    groups = OTHERS : 1,
             mygrp : 2;
}
```

if user jdoe has an interactive shell running in group `mygrp`, any process spawned by that shell process would also run in `mygrp` because its parent process was there. Simple inheritance is the mechanism that determines where most processes run, especially for short-lived processes.

How do I determine a goal for my workload?

NOTE
Be aware of the resource interaction for each of your workloads. Limiting a workload's memory allocation can also limit its CPU use. For example, if a workload uses memory and CPU in the ratio of 1:2, limiting the workload to 5% of the memory implies that it cannot use more than 10% of the CPU—even if it has a 20% CPU allocation.

To characterize the behavior of a workload, use the following example WLM configuration:

/opt/wlm/examples/wlmconf/manual_entitlement.wlm

Using this configuration, you can directly set an entitlement (allocation) for a workload using the `wlmsend` command. By gradually increasing the workload's allocation with a series of `wlmsend` calls, you can determine how various amounts of CPU affect the workload and its performance with respect to some metric that you may want to use in an SLO for the workload.

In addition, you can compare this research with similar data for the other workloads that will run on the system. This comparison gives you insight into which workloads you can combine (based on their needed CPU) on a single system and still achieve the desired SLOs. Alternatively, if you cannot give a workload its optimal amount of CPU, you will know what kind of performance to expect with a smaller allocation.

Once you know how the workload behaves, you can more easily decide the type of goal, either metric or usage, you want for it. You may even decide to just allocate the workload a fixed amount of CPU or an amount of CPU that varies directly in relation to some metric. For information on the different methods for getting your workload CPU, see the section "SLO TYPES" in the wlm(5) man page.

Similar to the manual_entitlement.wlm configuration, the /opt/wlm/toolkits/weblogic/config/manual_cpucount.wlm configuration allows you to adjust a workload group's CPU allocation with a series of `wlmsend` calls. manual_cpucount.wlm, however, uses a PSET as the basis for a workload group—and changes the group's allocation by one whole CPU at a time.

Metrics for workload management

You provide metric data to WLM so that it can:

- Determine new resource allocations, allowing the workload groups to achieve their SLOs
- Set shares-per-metric allocations
- Enable or disable SLOs

Where do I get metrics?

Sources for metric data include:

- Existing metrics (from log files, scripts, or commands you already have in place)
- ARM-instrumented applications
- GlancePlus
- Oracle databases
- C source code modified to use the WLM API

For more information on these sources, see the *HP-UX Workload Manager User's Guide*.

How do I feed the metrics to WLM?

Once you have the data, you have a number of options for forwarding that data to WLM:

- Command-line/shell script
- Perl
- stdout of a command
- Instrumenting with the WLM API

Sending data from the command line or from a shell script

You can send data to update a metric's value using the `wlmsend` command on the command line or in a shell script.

The following example WLM configuration, available in its entirety at /opt/wlm/examples/wlmconf/manual_entitlement.wlm as well as on http://www.hp.com/go/wlm, allows you to set a workload group's CPU allocation from the command line using `wlmsend`.

```
# Create a workload group called grp1 and define the binaries that make it up.
# In this example, the apache webserver binary httpd is used as the workload.
prm {
    groups = OTHERS : 1, grp1 : 2;
    apps = grp1 : /opt/hpws/apache/bin/httpd;
}

# Have HP-UX WLM manage the CPU allocation such that the workload group grp1
# gets 1 share per metric. That is, setting the metric myapp.desired.allocation
# to 20 results in 20 shares for grp1, setting the metric
# myapp.desired.allocation to 50 results in 50 shares for grp1, and so forth.
slo grp1 {
    pri = 1;
    entity = PRM group grp1;
    cpushares = 1 total per metric myapp.desired.allocation;
}
```

```
# Relay a metric from the outside user.
tune myapp.desired.allocation {
    coll_argv = wlmrcvdc;
}
```

This method of getting data to WLM uses the `wlmsend` and `wlmrcvdc` commands. To use this method:

1. Define a workload group and assign a workload to it

   In your WLM configuration file, define your workload group in a `prm` structure using the `groups` keyword. Assign a workload to the group using the `apps` keyword. In the example configuration above, the default group for users, called `OTHERS`, is explicitly defined. (WLM creates this group if you do not explicitly define it.) The group `grp1` is also defined, and `httpd` is assigned to it.

   ```
   prm {
       groups = OTHERS : 1, grp1 : 2;
       apps = grp1 : /opt/hpws/apache/bin/httpd;
   }
   ```

2. Set up `wlmrcvdc` to receive data for the metric

   In your WLM configuration file, set up a `tune` structure with `wlmrcvdc` as the data collector (value for `coll_argv`). Name the `tune` structure *`metric`*. You will use *`metric`* with `wlmsend` as well.

   ```
   tune metric {
   ...
      coll_argv = wlmrcvdc;
   ...
   }
   ```

   In our example, *`metric`* is myapp.desired.allocation, and the `tune` structure is:

   ```
   tune myapp.desired.allocation {
       coll_argv = wlmrcvdc;
   }
   ```

3. Use the metric

   You must now use the metric in an `slo` structure as part of a `goal` statement, `cpushares` statement, or `condition` statement in your WLM configuration file.

   In the example above, the metric myapp.desired.allocation is used in a `cpushares` statement:

   ```
   slo grp1 {
       pri = 1;
       entity = PRM group grp1;
       cpushares = 1 total per metric myapp.desired.allocation;
   }
   ```

4. Activate the configuration

   `# /opt/wlm/bin/wlmd -a config.wlm`

   substituting your configuration file's name for *`config.wlm`*.

5. Send the data using `wlmsend`

   When invoking `wlmsend` on the command line or in a script, indicate the *`metric`* being updated and its new *`value`*, which can be an integer or a floating-point value. Invoke `wlmsend` as follows:

   ```
   wlmsend metric value
   ```

   Alternatively, you can pipe the new value to `wlmsend`:

   ```
   cmnd1 | cmnd2 | [... |] wlmsend metric
   ```

   where *`cmnd1`* and *`cmnd2`* are commands for gathering or formatting the metric data. Be aware that piping may delay updates due to I/O buffering.

   For the example above, `wlmsend` can update the metric on the command line. The following command sets the CPU allocation to 10 shares:

   ```
   # /opt/wlm/bin/wlmsend myapp.desired.allocation 10
   ```

Sending data from a perl program

You can send data to update a metric's value using the `wlmsend` command in a perl program.

The following example (/opt/wlm/examples/userguide/reporter.pl) shows how to feed metrics to WLM using a perl program. The script uses the `wlmsend` command:

```
#!/opt/perl/bin/perl -w
#
# Name:
#       reporter.pl
#
# Version information:
#
#       $Revision: 1.4 $

# Here is an infinite loop
while (1) {
    $loop2_running = `env UNIX95=  ps -C loop2.pl | wc -l`;
    --$loop2_running; # Line count is one more due to header

    $loop3_running = `env UNIX95=  ps -C loop3.pl | wc -l`;
    --$loop3_running; # Line count is one more due to header

    # The following wlmsend commands set the values for the metrics
    # loop2_active and loop3_active to $loop2_running and
    # $loop3_running respectively. WLM picks up these values because of
    # the tune structures named for the metrics in the file
    # condition.wlm.

    system "/opt/wlm/bin/wlmsend loop2_active $loop2_running";
    system "/opt/wlm/bin/wlmsend loop3_active $loop3_running";

    sleep(60); # Wait for 60 seconds
}
```

The reporter.pl file is also available on http://www.hp.com/go/wlm.

To implement feeding of metrics from a perl program:

1. Define a workload group and assign a workload to it

   In your WLM configuration file, define your workload group in a `prm` structure using the `groups` keyword. Assign a workload to the group using the `apps` keyword.

   The example configuration file that works with reporter.pl is /opt/wlm/examples/userguide/condition.wlm. This configuration defines two groups `g2` and `g3` and assigns two perl programs to the groups:

```
prm {
    groups = g2 : 2,
             g3 : 3;
    apps = g2 : /opt/perl/bin/perl loop2.pl,
           g3 : /opt/perl/bin/perl loop3.pl;
    gmincpu = g2 : 5, g3 : 5;
    gmaxcpu = g2 : 30, g3 : 60;
}
```

   In this particular case, to ensure WLM places the perl programs in their assigned workload groups, you need to add "/opt/perl/bin/perl" (without the quotes) to your /opt/prm/shells file.

   The configuration also uses the `gmincpu` and `gmaxcpu` keywords to place lower and upper bounds on the amount of CPU the workload groups can be allocated.

2. Set up `wlmrcvdc` to receive data

   In your WLM configuration file, set up a `tune` structure with `wlmrcvdc` as the data collector (value for `coll_argv`). Name the `tune` structure *`metric`*. You will use *`metric`* with `wlmsend` as well.

```
tune metric {
...
   coll_argv = wlmrcvdc;
...
}
```

   In reporter.pl, the values for *`metric`* are `loop2_active` and `loop3_active`. The corresponding `tune` structure in condition.wlm for `loop2_active` is:

```
tune loop2_active {
    coll_argv = wlmrcvdc;
}
```

3. Use the metric

   You must now use the metric in an `slo` structure as part of a `goal` statement, `cpushares` statement, or `condition` statement in your WLM configuration file. The example configuration condition.wlm uses the metric in a `condition` statement:

```
slo test2 {
    pri = 1;
    cpushares = 15 total;
    entity = PRM group g2;
    condition = metric loop2_active;
}
```

4. Activate the configuration

   ```
   # /opt/wlm/bin/wlmd -a config.wlm
   ```

   substituting your configuration file's name for *config.wlm*.

5. Send the data using `wlmsend` in your perl program

   Use `wlmsend` in your perl program. When invoking `wlmsend`, indicate the *metric* being updated and its new *value*, which can be an integer or a floating-point value. Invoke `wlmsend` in a loop as follows:

   ```
   system "/opt/wlm/bin/wlmsend $metric $value";
   ```

   or in an `open` statement with the `print` statement in a loop:

   ```
   open(WLM, "|/opt/wlm/bin/wlmsend $metric");

   print WLM "$value\n";
   ```

   As we saw above, reporter.pl uses the first method to send data:

   ```
   system "/opt/wlm/bin/wlmsend loop2_active $loop2_running";
   ```

   If you would like WLM to take advantage of new data more quickly:

   - Set `wlm_interval` to a smaller value in your WLM configuration file (The default interval is 60 seconds.)
   - Decrease the `sleep` argument in reporter.pl

Using stdout of a command for values of a metric

If you have a command or script that displays values for a metric on stdout, you can use `wlmrcvdc` to send the values to WLM.

Assume you have a program called ordercounter. You could then set up a `tune` structure similar to the one below in your WLM configuration:

```
tune order_cnt {
    coll_argv = wlmrcvdc /opt/books/ordercounter;
}
```

The `wlmrcvdc` command forwards any output from ordercounter to WLM as values for the metric order_cnt.

To use the stdout of a command for WLM metrics:

1. Define a workload group and assign a workload to it

   In your WLM configuration file, define your workload group in a `prm` structure using the `groups` keyword. Assign a workload to the group using the `apps` keyword.

   The `prm` structure below defines group `grp1` and assigns one workload to it:

   ```
   prm {
       groups = grp1 : 2;
       apps = grp1 : /opt/books/orderprocessor;
   }
   ```

2. Set up `wlmrcvdc` to receive data

   In your WLM configuration file, set up a `tune` structure with `wlmrcvdc` as the data collector (value for `coll_argv`). Name the `tune` structure *`metric`*. Given an application named *`command`* that prints the metric data on stdout, place *`command`* as an argument to `wlmrcvdc`.

   In the configuration file:

   ```
   tune metric {
   ...
      coll_argv = wlmrcvdc command;
   ...
   }
   ```

   When the configuration is activated, WLM automatically starts the application given by *`command`*.

   As seen at the beginning of this section, a `tune` structure for the metric order_cnt might look like the following example:

   ```
   tune order_cnt {
       coll_argv = wlmrcvdc /opt/books/ordercounter;
   }
   ```

   `ordercounter` should run continuously in order to keep updating the order_cnt metric.

3. Use the metric

   You must now use the metric in an `slo` structure as part of a `goal` statement, `cpushares` statement, or `condition` statement in your WLM configuration file.

   The following `slo` structure uses the order_cnt metric to request two CPU shares for each unit of order_cnt. Thus, if order_cnt is equal to 1 for 100 orders, 2 for 200 orders, and so on, an order_cnt of 10 would result in a request of 20 CPU shares for the order-processing workload to handle the 1000 orders. The `slo` structure is:

   ```
   slo orders {
      pri = 1;
      entity = PRM group grp1;
      cpushares = 2 total per metric order_cnt;
   }
   ```

4. Activate the configuration

   ```
   # /opt/wlm/bin/wlmd -a config.wlm
   ```

   substituting your configuration file's name for *`config.wlm`*.

   `wlmrcvdc` now updates *`metric`* (order_cnt in this case) with the values from *`command`*'s stdout.

WLM API

If you are writing a data collector in the C programming language, you can use the WLM API to communicate metric data directly to WLM.

After creating your data collector with the WLM API (described in the section "WLM API" on page 24), set up your WLM configuration file and activate it as follows:

1. Define a workload group and assign a workload to it

   In your WLM configuration file, define your workload group in a `prm` structure using the `groups` keyword. Assign a workload to the group using the `apps` keyword.

   The following example, which comes from the example configuration file /opt/wlm/examples/wlmconf/metric_condition.wlm, explicitly defines the default group for users, called `OTHERS`. (WLM creates this group if you do not explicitly define it.) The group `jobs` is also defined, and the `submit` application is assigned to it.

   ```
   prm {
       groups = OTHERS : 1,
                jobs : 2;
       apps =   jobs: /opt/batch/bin/submit;
   }
   ```

2. Set up the `tune` structure to receive data

   In your WLM configuration file, set up a `tune` structure with the *`program`* that uses the WLM API as the data collector (value for `coll_argv`). Name the `tune` structure *`metric`*, which matches the name of the metric used in the WLM API call `wlm_mon_attach()`.

   In the configuration file, set up the `tune` structure:

   ```
   tune metric {
   ...
      coll_argv = program;
   ...
   }
   ```

   WLM automatically starts the application given by *`program`*.

   The `tune` structure from the example metric_condition.wlm is:

   ```
   tune avg_completion_time {
      coll_argv = /opt/batch/metrics/job_complete_time -minutes;
   }
   ```

   The *`metric`* is avg_completion_time. The *`program`*, which includes WLM API calls, is /opt/batch/metrics/job_complete_time.

NOTE

The stdout and stderr for each application started by WLM are discarded. However, using the `coll_stderr` tunable, you can redirect stderr to a file of your choosing. For more information, see the wlmconf(4) man page.

---

3. Use the metric

   You must now use the metric in an `slo` structure as part of a `goal` statement, `cpushares` statement, or `condition` statement in your WLM configuration file.

   The avg_completion_time metric is used below in a `goal` statement:

```
slo job_processing {
   pri = 1;
   mincpu = 40;
   maxcpu = 90;
   entity = PRM group jobs;
   goal = metric avg_completion_time < 10;
   condition = metric number_of_active_jobs > 0;
}
```

   The metric in the `condition` statement is not mentioned here; however, you can read more about it in the metric_condition.wlm file.

4. Activate the configuration

   `# /opt/wlm/bin/wlmd -a config.wlm`

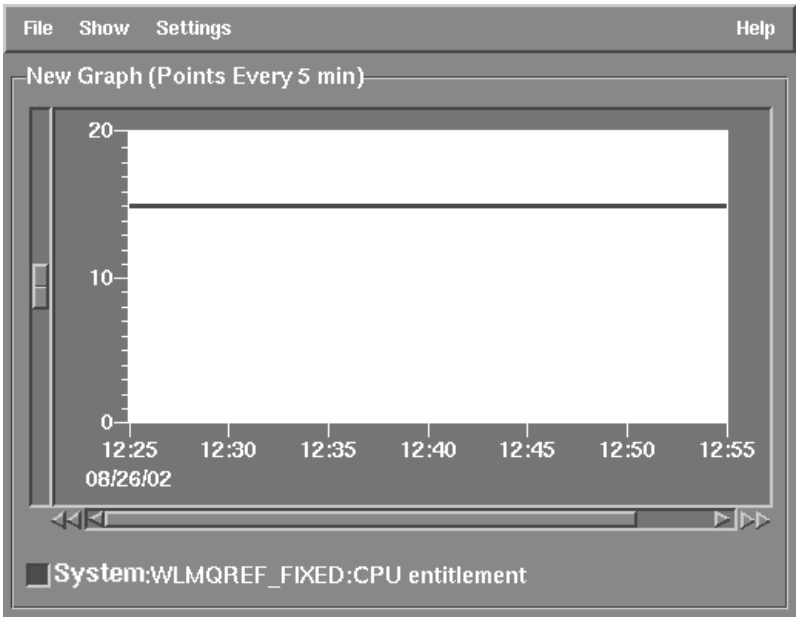
   substituting your configuration file's name for *`config.wlm`*.

What are some common WLM tasks?

WLM is a powerful tool that allows you to manage your systems in numerous ways. The sections below explain some of the more common tasks that WLM can do for you.

Providing a fixed amount of CPU

WLM allows you to give a workload group a fixed amount of CPU. The figure below shows a group with a fixed allocation of 15 CPU shares.



WLM allows you to allocate a fixed amount of CPU using:

- portions of processors
- whole processors (processor sets)

Portions of processors

One method for providing a fixed amount of CPU is to set up an SLO to give a workload group a portion of each available processor. To set up this type of SLO:

1. Define the workload group and assign a workload to it

   In your WLM configuration file, define your workload group in a `prm` structure using the `groups` keyword. Assign a workload to the group using the `apps` keyword.

   The following example defines a group named `sales`.

   ```
   prm {
       groups = sales : 2;
       apps = sales : /opt/sales/bin/sales_monitor;
   }
   ```

   The `sales_monitor` application is placed in the `sales` workload group using the `apps` statement. For other methods on placing a workload in a certain group, see "How do I put an application under WLM control?" on page 15.

2. Define a fixed-allocation SLO for the workload group

   In the `slo` structure, use the `cpushares` keyword with `total` to request CPU for the workload group. The following SLO requests 15 CPU shares for the workload group `sales`. Based on SLO priorities and available CPU, WLM attempts to meet the request.

   ```
   slo fixed_allocation_example {
       pri = 2;
       entity = PRM group sales;
       cpushares = 15 total;
   }
   ```

3. Activate the configuration

   ```
   # /opt/wlm/bin/wlmd -a config.wlm
   ```

   substituting your configuration file's name for *`config.wlm`*.

Whole processors (processor sets)

Another method for providing a workload group with a fixed amount of CPU is to define the group based on a processor set, or PSET. The processor sets feature is available for HP-UX 11i V1.0 (B.11.11) and can be downloaded from http://software.hp.com free of charge. Processor sets are included with HP-UX 11i V2.0 (B.11.23).

As in the previous case, the workload requires a constant amount of CPU. Placing the workload in a workload group based on a PSET, the group has sole access to the processors in the PSET.

To place an application in a workload group based on a PSET:

1. Define the workload group based on a processor set and assign a workload to it

   In your WLM configuration file, define your workload group in a `prm` structure using the `groups` keyword. Assign a workload to the group using the `apps` keyword.

   The following example shows the `sales` group as a PSET that is assigned two CPUs through a `gmincpu` statement. (When using `gmincpu` for a group based on a PSET, 100 represents a single processor. Thus, 200 represents two processors.) The application sales_monitor is assigned to the group:

   ```
   prm {
       groups = sales : PSET;
       apps = sales : /opt/sales/bin/sales_monitor;
       gmincpu = sales : 200;
   }
   ```

2. Activate the configuration

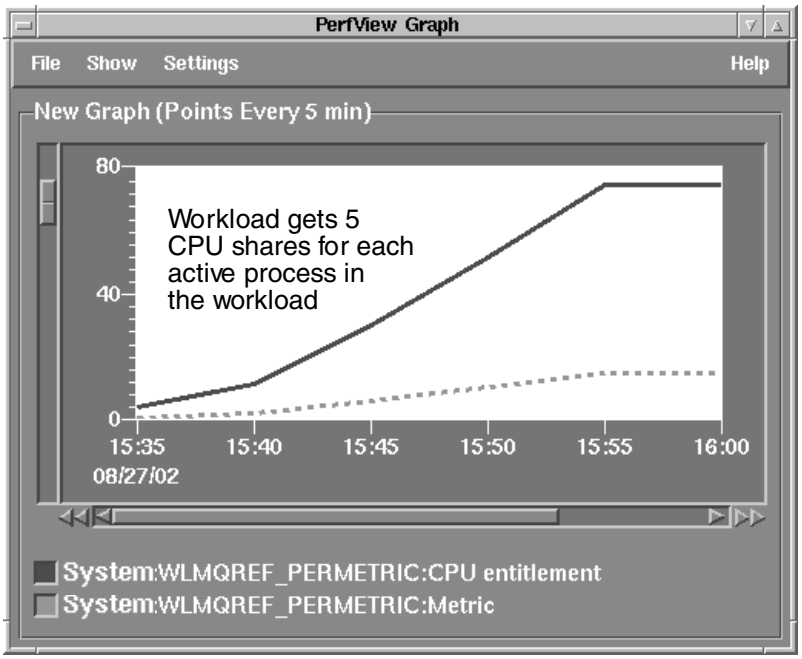
   ```
   # /opt/wlm/bin/wlmd -a config.wlm
   ```

   substituting your configuration file's name for *`config.wlm`*.

Providing CPU in proportion to a metric

To adjust a workload's CPU allocation in step with a given metric, such as number of processes in the workload, users connected to a database, or any other metric, use the `cpushares` keyword with `per metric` in an `slo` structure.

The following figure shows a workload group that is allocated 5 CPU shares for each active process in the workload.



To set up a workload group with an allocation that varies in proportion to a metric:

1. Define the workload group and assign a workload to it

   In your WLM configuration file, define your workload group in a `prm` structure using the `groups` keyword. Assign a workload to the group using the `apps` keyword.

   In this example, the focus is the `sales` group, which will get a varying amount of CPU based on a metric:

```
prm {
    groups = sales : 2;

    apps = sales : /opt/sales/bin/sales_monitor;
}
```

2. Define the SLO

   The SLO in your WLM configuration file must specify a priority (`pri`) for the SLO, the workload group to which the SLO applies (`entity`), and the `cpushares` statement to request CPU in proportion to a metric.

   The following `slo` structure shows the `cpushares` statement. This SLO requests 5 CPU shares for the `sales` group for each process in the application—as given by the metric application_procs.

```
slo per_metric_example {
    pri = 1;
    mincpu = 25;
    maxcpu = 50;
    entity = PRM group sales;
    cpushares = 5 total per metric application_procs;
}
```

   However, if application_procs is less than five or greater than ten, the values for the optional keywords `mincpu` and `maxcpu` will force the request for additional CPU shares to be between 25 and 50.

   Sources for a metric include data available through GlancePlus, SNMP, and any other metrics you may already have available.

3. Set up WLM to take values for the metric

   The simplest method for getting a metric to WLM is through the WLM data collector `wlmrcvdc`.

   To receive a value for the metric application_procs, set up a `tune` structure. The following `tune` structure takes advantage of `glance_prm`, one of the commands that WLM provides to simplify pulling data from the optional HP product GlancePlus. In this case, `glance_prm` pulls the APP_ACTIVE_PROC metric for the `sales` workload group:

```
tune application_procs {
    coll_argv = wlmrcvdc glance_prm APP_ACTIVE_PROC sales;
}
```

   As a result of this structure, each time the `glance_prm` command prints a value on standard out, `wlmrcvdc` sends the value to WLM for use in changing the CPU allocation for the `sales` group.

   For information on other ways to use `wlmrcvdc`, see the wlmrcvdc(1M) man page.

4. Activate the configuration

   `# /opt/wlm/bin/wlmd -a` *`config.wlm`*

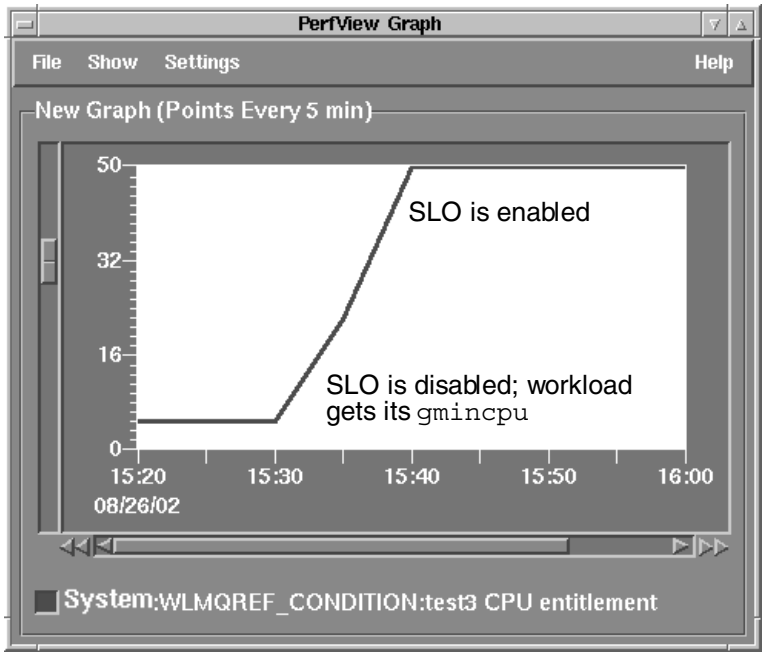
   substituting your configuration file's name for *`config.wlm`*.

Providing CPU for a given time period

For workloads that are only needed for a certain period of time, whether it is once a day, week, or any other period, WLM provides the `condition` keyword. This keyword is placed in an `slo` structure and enables the SLO when the `condition` statement is true.

By default, when the SLO is not enabled, its workload group gets its `gmincpu` value. If this value is not set, the workload group gets the default 1% of the CPU. (Setting the `transient_groups` keyword to 1 changes this behavior.)

The figure below shows the allocation for a workload group increase after its SLO is enabled.



To provide a workload group with CPU on a schedule:

1. Define the workload group and assign a workload to it

   In your WLM configuration file, define your workload group in a `prm` structure using the `groups` keyword. Assign a workload to the group using the `apps` keyword.

   In this example, the `sales` group is the group of interest again.

```
prm {
    groups = sales : 2;

    apps = sales : /opt/sales/bin/sales_monitor;
}
```

2. Define the SLO

   The SLO in your WLM configuration file must specify a priority (`pri`) for the SLO, the workload group to which the SLO applies (`entity`), and either a `cpushares` statement or a `goal` statement so that WLM grants the SLO's workload group some CPU. The `condition` keyword determines when the SLO is enabled or disabled.

   The following `slo` structure shows a fixed-allocation SLO for the `sales` group. When enabled, this SLO requests 25 CPU shares for the `sales` group. Based on the `condition` statement, the SLO is enabled between 8pm and 11pm every day.

```
slo condition_example {
    pri = 1;
    entity = PRM group sales;
    cpushares = 25 total;
    condition = 20:00 - 22:59;
}
```

   Whenever the `condition` statement is false, the SLO is disabled and does not make any CPU requests for the `sales` group. When a group has no enabled SLOs, it gets its `gmincpu` value (if set); otherwise, it gets no more than a 1% CPU allocation.

   You can use times, dates, or metrics to enable or disable an SLO.

   For information on the syntax for the `condition` keyword, see the wlmconf(4) man page.

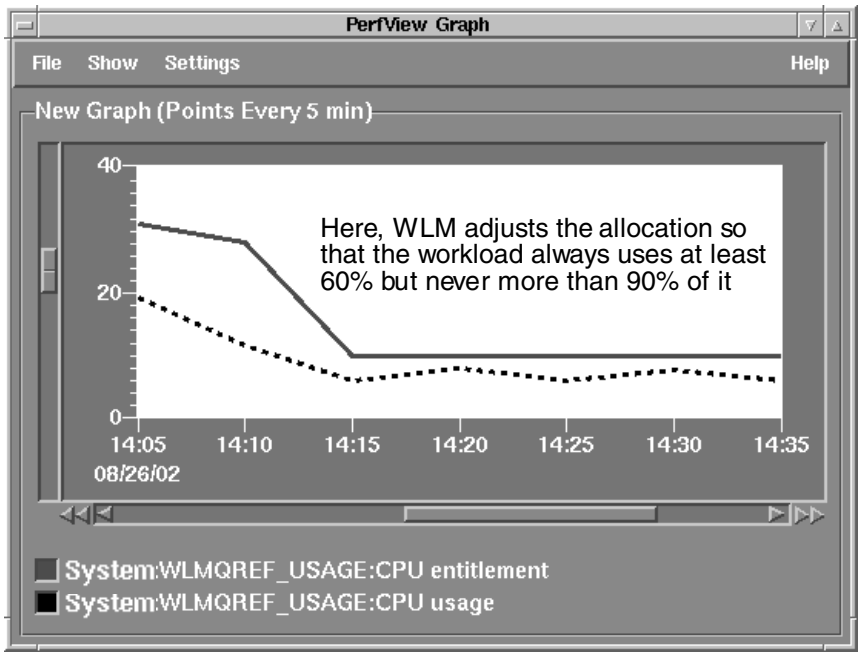
3. Activate the configuration

```
# /opt/wlm/bin/wlmd -a config.wlm
```

   substituting your configuration file's name for *`config.wlm`*.

Providing CPU as it is needed

To ensure a workload gets the CPU it needs—without preventing other workloads access to unused CPU resources—WLM allows you to define usage goals. A major advantage with usage goals is that you do not need to track and feed a metric to WLM. The figure below shows a group's allocation (or entitlement) decrease as the group's usage decreases.



With a usage goal, you indicate for a workload how much of its CPU allocation it should use. Then, a workload's CPU allocation is reduced if it is not consuming enough of its current allocation, allowing other workloads to consume more CPU if needed. Similarly, if the workload is using a high percentage of its allocation, it is granted more CPU.

To define a usage goal for a workload group:

1. Define the workload group and assign a workload to it

   In your WLM configuration file, define your workload group in a `prm` structure using the `groups` keyword. Assign a workload to the group using the `apps` keyword.

   With this example, the `sales` group is the group of interest.

```
prm {
    groups = sales : 2;

    apps = sales : /opt/sales/bin/sales_monitor;
}
```

2. Define the SLO

   The SLO in your WLM configuration file must specify a priority (`pri`) for the SLO, the workload group to which the SLO applies (`entity`), and a usage `goal` statement.

   The following `slo` structure for the `sales` group shows a usage `goal` statement.

```
slo usage_example {
    pri = 1;
    mincpu = 20;
    maxcpu = 60;
    entity = PRM group sales;
    goal = usage _CPU 80 90;
}
```

   With usage goals, WLM adjusts the amount of CPU it grants a workload group so that the group uses between 50% and 75% of its allocated CPU. In the example above though, with the values of 80 and 90 in the `goal` statement, WLM would try to increase or decrease the workload group's CPU allocation until the group is using between 80% and 90% of the allocated share of CPU. However, in attempting to meet the usage goal, WLM's new CPU allocations to the workload group will typically be within the `mincpu`/`maxcpu` range.

3. Activate the configuration

```
# /opt/wlm/bin/wlmd -a config.wlm
```

   substituting your configuration file's name for *`config.wlm`*.

What else can WLM do?

Run in passive mode to verify operation

WLM provides a passive mode that allows you to see how WLM will approximately respond to a given configuration—without putting WLM in charge of your system's resources. Using this mode, you can safely check that your configuration behaves as expected—with minimal effect on the system. Besides being useful in understanding and experimenting with WLM, passive mode can be helpful in capacity-planning activities.

Auditing and billing

When you activate a WLM configuration or a WLM global arbiter configuration, you have the option of generating audit data. You can use the audit data files directly or view them using the `wlmaudit` command. For more information, see the wlmaudit(1M), wlmd(1M), and wlmpard(1M) man pages.

Automatically resize processor sets (PSETs)

With multi-processor systems, you can group processors together to form processor sets, also known as PSETs. By creating PSETs, you isolate CPU resources for users or applications.

WLM allows you to define workload groups based on PSETs. If you then specify SLOs for these workload groups, WLM automatically adjusts the number of processors in the PSETs based on progress toward the SLOs.

Automatically manage SLOs globally, across multiple virtual systems

HP-UX Virtual Partitions (vPars) are software-based virtual systems, each running its own instance of the HP-UX operating system. WLM can move processors between these virtual systems to better achieve the SLOs you define using WLM within each virtual system.

For more information on this feature, see the *HP-UX Workload Manager User's Guide* on http://docs.hp.com/hpux/netsys/ or the wlmpard(1M) and wlmparconf(4) man pages.

Automatically manage SLOs globally, across multiple node partitions

Node Partitions (nPars) are hardware-based partitions, each running its own instance of the HP-UX operating system. Using iCOD software, WLM can simulate the movement of processors between nPars to better achieve the SLOs you define using WLM within each virtual system. (The processor movement is simulated by deactivating a processor on one nPar and activating a processor on another nPar.)

For more information on this feature, see the *HP-UX Workload Manager User's Guide* on http://docs.hp.com/hpux/netsys/ or the wlmpard(1M) and wlmparconf(4) man pages.

Optimize use of Pay Per Use systems

With Pay Per Use (PPU) systems, you pay only for the amount of CPU you use. With PPU versions prior to B.05.00, billing was based on the number of active CPUs. With WLM and its Pay Per Use Toolkit (PPUTK), you can optimize the resource utilization to the minimum needed to meet your SLOs, thereby reducing your costs.

For more information on this feature, see the *HP-UX Workload Manager Toolkits User's Guide* on http://docs.hp.com/hpux/netsys/ or the utilitydc(1M) man page.

Integrate with various third-party products

HP has developed a number of toolkits to assist you in quickly and effectively deploying WLM for use with your key applications. The freely available WLM Toolkits product consists of several toolkits, each with example configuration files. Toolkits are currently available for:

- Oracle® Databases
- HP's Pay Per Use and iCOD features
- HP-UX Apache-based Web Server software
- BEA WebLogic Server™
- SNMP agents
- SAS® software

In addition, WLM Toolkits provides a duration management toolkit.

WLM Toolkits are included with WLM. They are also freely available on http://software.hp.com.

For more information on the toolkits, see the *HP-UX Workload Manager Toolkits User's Guide* on http://docs.hp.com/hpux/netsys/ or the wlmtk(5) man page.

What status information does WLM provide?

WLM has three log files to keep you informed:

- /var/opt/wlm/msglog

  WLM prints errors and warnings about configuration file syntax on stderr. For messages about WLM's ongoing operations, WLM logs error and informational messages to /var/opt/wlm/msglog, as shown below:

```
05/07/02 14:12:44 [I] (p13931) Ready to forward data for metric
"m_apache_access_2min"
05/07/02 14:12:44 [I] (p13931) Using "/door00/wlm_cfg/apache_access.pl"
05/07/02 14:12:44 [I] (p13932) Ready to forward data for metric "m_list.cgi_procs"
05/07/02 14:12:44 [I] (p13932) Using "/door00/wlm_cfg/proc_count.sh"
05/07/02 14:12:44 [I] (p13930) Ready to forward data for metric
"m_apache_access_10min"
05/07/02 14:12:44 [I] (p13930) Using "/door00/wlm_cfg/apache_access.pl"
05/07/02 14:12:44 [I] (p13929) Ready to forward data for metric "m_nightly_procs"
05/07/02 14:12:44 [I] (p13929) Using "/opt/wlm/lbin/coll/glance_prm"
05/07/02 14:12:44 [I] (p13921) wlmd 13921 starting
05/31/02 03:42:10 [I] (p13921) /var/opt/wlm/wlmdstats has been renamed to
/var/opt/wlm/wlmdstats.old.
```

- /var/opt/wlm/wlmdstats

  WLM can optionally produce a log file that includes data useful for verifying WLM's management or for fine-tuning your WLM configuration. This log file, /var/opt/wlm/wlmdstats, is created when you start the WLM daemon with the `-l` option:

  ```
  # /opt/wlm/bin/wlmd -a config.wlm -l slo
  ```

  For information on the `-l` option, see the wlmd(1M) man page.

  Here are some lines from a wlmdstats file:

```
1024328341 SLO=s_nice_xtra_min sloactive=1 goaltype=nogoal goal=nan goalsatis=1
met=nan metfresh=0 mementitl=0 cpuentitl=20 clipped=0 controlling=0
1024328341 SLO=s_team_playground_xtra_min sloactive=1 goaltype=nogoal goal=nan
goalsatis=1 met=nan metfresh=0 mementitl=0 cpuentitl=5 clipped=0 controlling=1
```

- /var/opt/wlm/wlmpardstats

  WLM's global arbiter, used in managing SLOs across virtual partitions, can optionally produce a statistics log file. This log file, /var/opt/wlm/wlmpardstats, is created when you start the WLM global arbiter daemon with the `-l` option:

  ```
  # /opt/wlm/bin/wlmpard -a config.wlm -l vpar
  ```

  For information on the `-l` option, see the wlmpard(1M) man page.

How do I monitor WLM?

For monitoring WLM, there are several methods, as described below.

ps [-P] [-R *workload_group*]

The `ps` command has options that are specific to PRM, which WLM uses to define workload groups:

- `-P`

  Adds the column `PRMID`, showing the workload group for each process.

  ```
  # ps -P
    PRMID    PID TTY       TIME COMMAND
       g3   6793 ttyp1     1:52 loop.pl
       g3   6463 ttyp1     7:02 loop3.pl
       g2   6462 ttyp1     4:34 loop2.pl
  ```

- `-R` *`workload_group`*

  Lists only the processes in the group named by *`workload_group`*. Here is output showing processes in the workload group `g3`:

  ```
  # ps -R g3
    PID TTY       TIME COMMAND
    6793 ttyp1     1:29 loop.pl
    6463 ttyp1     6:41 loop3.pl
  ```

wlminfo

The `wlminfo` command, available in /opt/wlm/bin/, displays information about SLOs, metrics, workload groups, vPars or nPars, and the current host. Here is example `group` info:

```
# wlminfo group

Workload Group   PRMID  CPU Shares  CPU Util  Mem Shares  State
OTHERS               1       65.00      0.00        0.00  ON
g2                   2       15.00      0.00        0.00  ON
g3                   3       20.00      0.00        0.00  ON
```

wlmgui

The `wlmgui` command, available in /opt/wlm/bin/, graphically displays information about SLOs, metrics, workload groups, partitions, and the current host.

prmmonitor

The `prmmonitor` command, available in /opt/prm/bin/, displays current configuration and resource usage information:

```
PRM configured from file:  /var/opt/wlm/tmp/wmprmBAAa06128
File last modified:        Thu Aug 29 08:35:23 2002

HP-UX samples B.11.00 A 9000/889    08/29/02

Thu Aug 29 08:43:16 2002    Sample:  1 second
CPU scheduler state:  Enabled, CPU cap ON
                                                    CPU       CPU
PRM Group                          PRMID   Entitlement      Used
_____________________________________________________________________
OTHERS                                 1        65.00%    0.00%
g2                                     2        15.00%    0.00%
g3                                     3        20.00%    0.00%

PRM application manager state:  Enabled  (polling interval: 30 seconds)
```

prmlist

The `prmlist` command, available in /opt/prm/bin, displays current CPU allocations plus user and application configuration information:

```
PRM configured from file:  /var/opt/wlm/tmp/wmprmBAAa06128
File last modified:        Thu Aug 29 08:35:23 2002

PRM Group                       PRMID    CPU Entitlement
-------------------------------------------------------
OTHERS                              1             65.00%
g2                                  2             15.00%
g3                                  3             20.00%

PRM User                        Initial Group                 Alternate Group(s)
-------------------------------------------------------------------------------
root                            (PRM_SYS)

PRM Application                 Assigned Group                Alternate Name(s)
-------------------------------------------------------------------------------
/opt/perl/bin/perl              g2                            loop2.pl
/opt/perl/bin/perl              g3                            loop3.pl
```

GlancePlus

The optional HP product GlancePlus allows you to display resource allocations for the workload groups as well as list processes for individual workload groups.

wlm_watch.cgi

This CGI script, available in /opt/wlm/toolkits/apache/bin/, allows you to monitor WLM using a web browser interface to `prmmonitor`, `prmlist`, and other monitoring tools.

For information on setting up this script, see the wlm_watch(1M) man page.

Status and message logs

WLM provides the following logs:

- /var/opt/wlm/msglog
- /var/opt/wlm/wlmdstats
- /var/opt/wlm/wlmpardstats

For information on these logs, including sample output, see the section “What status information does WLM provide?” on page 34.

EMS

EMS (Event Monitoring Service) polls various system resources and sends messages when events occur. WLM’s `wlmemsmon` command provides numerous resources for event monitoring. For information on these resources, see the wlmemsmon(1M) man page.

For more information

For more information on HP-UX Workload Manager, contact any of our worldwide sales offices or visit our web site at: http://www.hp.com/go/wlm.

To learn more about the adaptive enterprise and virtualization, please visit http://www.hp.com/go/virtualization.

The following list of references provides useful background information on related products and topics:

HP-UX Workload Manager (HP-UX WLM)
http://www.hp.com/go/wlm
HP-UX Workload Manager User's Guide http://www.docs.hp.com/hpux/netsys/

HP-UX Workload Manager Toolkits, which consists of:
HP-UX WLM Oracle Database Toolkit (ODBTK)
HP-UX WLM Pay Per Use Toolkit (PPUTK)
HP-UX WLM Duration Management Toolkit (DMTK)
HP-UX WLM Toolkit for SAS Software (SASTK)
HP-UX WLM Toolkit for Apache (ApacheTK)
HP-UX WLM BEA WebLogic Server Toolkit (WebLogicTK)
HP-UX WLM SNMP Toolkit (SNMPTK)
http://www.hp.com/go/wlm
HP-UX Workload Manager Toolkits User's Guide http://www.docs.hp.com/hpux/netsys/

HP Process Resource Manager (PRM)
http://www.hp.com/go/prm
Process Resource Manager User's Guide http://docs.hp.com/hpux/ha/

HP Partitioning Continuum
http://www.hp.com/go/servicecontrol

Application Response Measurement (ARM)
Application Response Measurement (ARM) API
http://www.cmg.org/regions/cmgarmw/

Event Monitoring Service (EMS)
http://www.unix.hp.com/highavailability

Servicecontrol Manager
http://www.hp.com/go/servicecontrol

Instant Capacity on Demand (iCOD)
http://www.hp.com/go/icod

01/2004

invent