Using HP-UX Workload Manager effectively with your most critical applications

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. In combination with the WLM Toolkits, you can quickly and effectively deploy 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
- Apache web server software
- BEA WebLogic Server™
- HP-UX SNMP agent
- Duration management
- SAS® software

This paper covers WLM Toolkits A.01.05, which runs on PA-RISC servers under HP-UX 11.0 and HP-UX 11i V1.0 (B.11.11). It also runs on Itanium®-based servers under HP-UX 11i V2.0 (B.11.23).

Overview

The WLM Toolkits help you capture the power of HP-UX Workload Manager. In case you're not familiar with WLM, it is a software product that provides automatic resource allocation and application performance management through the use of prioritized service-level objectives (SLOs). When using WLM, the system administrator:

- Defines workload groups for each workload
- Places the applications or users that define the workloads into their own workload groups (the workload's processes share the resources WLM makes available to the workload group)
- Creates one or more SLOs for each workload group

In defining an SLO, the system administrator specifies a priority and optionally a metric goal or a resource usage goal. As the applications run, WLM compares the application metrics or usage against the goals. WLM then automatically adjusts the amount of CPU available to the workload groups to achieve each goal, based on available CPU and SLO priority. WLM can also manage real memory, although not in response to SLO performance. Disk bandwidth allocations are statically assigned in the WLM configuration file. If multiple applications or users within a workload group are competing for resources, standard HP-UX resource management determines the resource allocation.

As you will see in the upcoming examples, WLM allocates resources in "shares". A CPU share represents 1/100 of a single CPU or 1/100 of each CPU in the server, depending on WLM's mode of operation. If the default 100 shares are available, each share is 1% of the system's CPU.

Key uses of WLM include:

- Using excess capacity on servers by consolidating multiple applications on fewer servers while ensuring the highest priority applications still get the resources they need in times of peak demand
- Automatically reallocating system resources in response to changing priorities, conditions that change over time (night/day, month-end processing, and so forth), resource demand, and application performance
- Managing reserve resources, such as Pay Per Use or iCOD CPUs
- Automatically changing the number of CPUs in HP-UX Processor Sets, HP-UX Virtual Partitions, or Node Partitions in response to SLO performance of workloads in the partitions

HP-UX WLM Oracle® database toolkit

HP-UX WLM enables you to place Oracle® instances and other applications in their own WLM workload groups. With the instances and applications separated, WLM can then manage the performance of each instance and application through prioritized SLOs. For the Oracle database administrator, this means the ability to allocate different CPU resources to Oracle database instances by modifying the WLM configuration.

This partitioning is done between separate Oracle database instances—not within a single instance. That is, all Oracle processes related to a specific database instance must be assigned to the same workload group. This includes Oracle server processes ("shadow" processes) that run on behalf of users accessing the database.

To better take advantage of WLM's partitioning ability with respect to Oracle instances, Hewlett-Packard provides the WLM Oracle Database Toolkit, or ODBTK. ODBTK simplifies getting metrics on Oracle database instances into WLM. This toolkit, which is included with WLM, works with Oracle 8.0.x, Oracle 8.1.5, Oracle 8.1.6, Oracle 8.1.7, and Oracle 9.0.1. For information on toolkit requirements, see the *HP-UX Workload* Manager *Toolkits User's Guide*.

What metrics are available?

The following types of database metrics are available:

- Time elapsed while user-defined 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 database administrator to see current performance information.)

Why use ODBTK?

The following sections illustrate a few reasons for using ODBTK to manage your instances. These sections discuss various items you would set in an SLO in a WLM configuration file. ODBTK comes with example WLM configuration files. These files, some of which are used as a basis for the examples below, show how you can take control of your Oracle instances and their resources quickly and easily.

All of the examples below focus on a single instance to simplify showing what WLM and ODBTK can accomplish. Of course, WLM is most useful when multiple database instances and applications are running on a single system.

Set response-time goals

In this first example, we have a workload group named Instance1. With priority 1, this group's SLO has the highest possible priority. However, other groups may also have priority 1 SLOs. The group's CPU will be in the range 20% to 80% of the system CPU, assuming there is enough CPU to meet this SLO and all other SLOs at priority 1. WLM will adjust the CPU allocation for the instance's workload group within this range so that the response time for the 'select' database operation is less than 10 seconds.

Group: Instance1
Priority: 1
Goal: 'Select' response time < 10
Min CPU: 20%
Max CPU: 80%

Increase CPU when a particular user connects

Our next example shows two SLOs for a workload group containing one instance. These SLOs have different priorities, though. This type of configuration allows for a higher priority, must-meet goal and an optional stretch goal. With these SLOs, Instance1's group gets 10% of the CPU at priority 1. If all other priority 1 SLOs are met, WLM then tries to satisfy the priority 2 SLOs. If the user BATCHUSER is connected to the instance at this point, the group's CPU allocation is increased to 30%.

Group: Instance1
Priority: 1
Min CPU: 10%
Max CPU: 10%

Group: Instance1
Priority: 2
Min CPU: 30%
Max CPU: 30%
Condition: user BATCHUSER connects to the instance

Increase CPU when more than *n* users connect

The following example also has multiple SLOs for the instance's workload group. The priority 1 SLO is not always in effect. WLM will attempt to grant the workload group 40% of the CPU, but only if there are more than 10 users connected to the instance. No such conditions are in place for the second SLO, which is priority 2. This SLO, which grants the workload group 20% of the CPU, can be met only when CPU remains after satisfying all priority 1 SLOs that are in effect. This helps make sure the workload group gets some CPU when there are fewer than 10 users connected.

Group: Instance1
Priority: 1
Min CPU: 40%
Max CPU: 40%
Condition: More than 10 users connected to the instance

Group: Instance1
Priority: 2
Min CPU: 20%
Max CPU: 20%

Give an instance *n* CPU shares for each process in the instance

With the next example, the instance's workload group gets an additional 3% of the CPU for each additional process, up to a maximum of 80%. So if the instance has 15 processes, the group gets 45% of the CPU. (Note that Oracle background processes for an instance may affect the process count depending on how this number is measured.)

Group: Instance2
Priority: 1
Min CPU: 5%
Max CPU: 80%
Goal: 3% of the CPU for each process in the instance

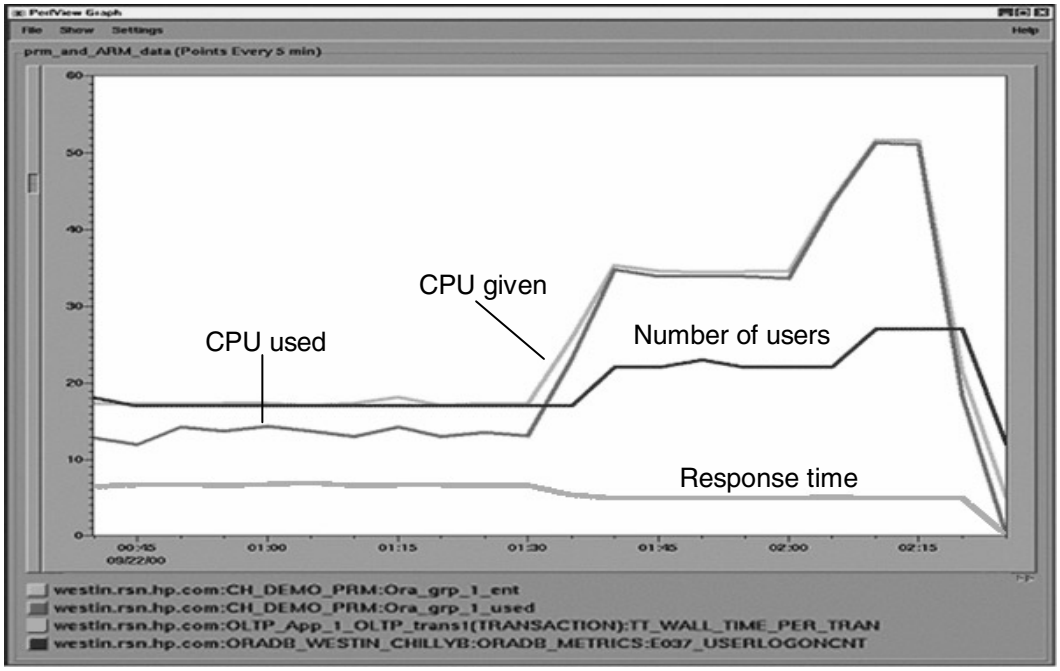
Give an instance *n* CPU shares for each user connection to the instance

Similarly, the next SLO increases the CPU for the instance's workload group based on the number of user connections. Thus, with 5 user connections, the workload group gets 15%.

Group: Instance3
Priority: 1
Min CPU: 5%
Max CPU: 90%
Goal: 3% of the CPU for each user connection to the instance

Figure 1, produced using the optional product "performance manager" (formerly known as HP PerfView), illustrates this type of goal with an actual Oracle Instance. The horizontal axis is time, and the vertical axis is CPU. However, "wall time per transaction" and "number of users" are also shown. For the wall time metric, the vertical axis is also used to measure seconds. The two increases in the "number of users" line represent going from 5 users to 10, then from 10 to 15.

Figure 1. Shares per metric goal



Because this SLO increases the workload group's CPU based on the number of users, these additional users cause WLM to grant the workload group more CPU. As a result of the extra CPU cycles, the number of users does not adversely affect the response time.

In fact, the response time stays roughly constant—providing a consistent level of service despite the increasing load.

Tools in 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 WLM. 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 WLM.
- `smooth`
  `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. The `cpushares` keyword is available starting with WLM Version A.01.02.

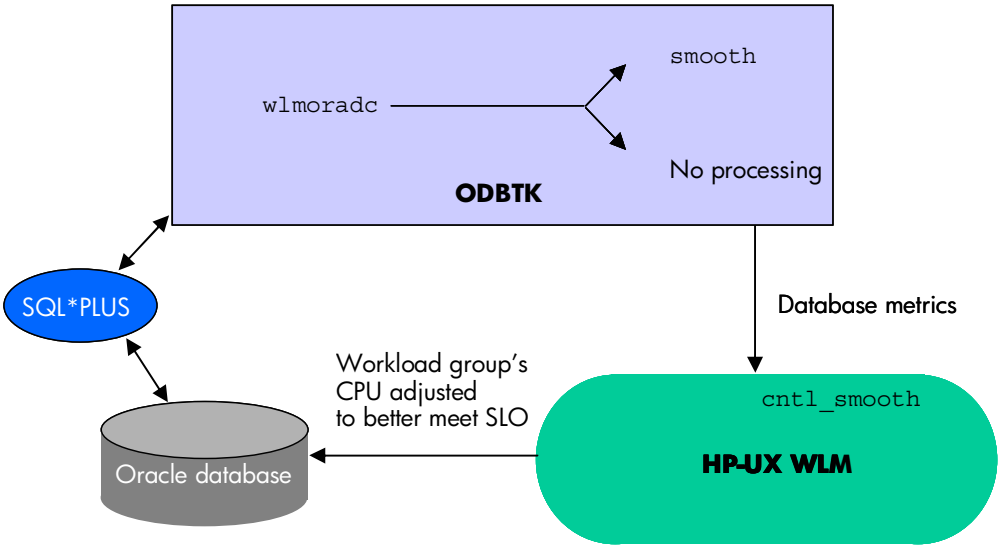
The `cpushares` keyword allows you to request shares based on the value of a metric. This enables you to create goal-based SLOs in WLM of the form "x CPU shares for each metric y," such as three CPU shares for each process in a workload group.

How ODBTK works

The figure below illustrates how 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 then adjusts the CPU for the instance's workload group so that it better meets its SLOs. (Starting with WLM A.02.02, the WLM configuration file tunable `cntl_smooth` is recommended over the `smooth` utility.)

This figure shows only a single instance, ignoring the fact that other database instances and applications may be on the system competing for resources.

Figure 2. How ODBTK works



How to use ODBTK

ODBTK comes with numerous example configuration files for both WLM and the included tool `wlmoradc`. Look at these files to get ideas of how to use ODBTK with WLM. After determining what you want to measure your instances on, you can copy and modify the WLM configuration files to reduce your implementation time. Also be sure to look at the `wlmoradc` example configuration files. These files are for storing SQL statements, among other items. The example files provide the SQL statements for such tasks as getting the number of user connections or the number of processes for an instance.

Breaking those steps out more formally, the procedure for using ODBTK to provide Oracle database metrics to WLM is:

1. Determine which example WLM configuration file is appropriate
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 a `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` and fix any errors
6. Activate the WLM configuration file with `wlmd -a`

Example configurations

ODBTK offers the following example WLM configuration files:

- alpha_shares_per_user.wlm
  The HP-UX Workload Manager Toolkits User's Guide shows the process of customizing a supplied file, shares_per_user.wlm, for a particular set of instances is given. alpha_shares_per_user.wlm is the result of that customization and is included here for educational purposes.
- batchuser_boost.wlm
  Demonstrates a conditional allocation example: Each instance has a fixed minimum allocation and gets a 'boost' if user 'BATCHUSER' connects to the instance.
- manual_payroll_boost.wlm
  Demonstrates a conditional allocation example where the condition can be triggered from the command line using the `wlmsend` utility.
- shares_per_process.wlm
  Demonstrates a parametric allocation: Each instance gets five CPU shares per process. Very similar to the shares_per_user.wlm example.
- shares_per_user.wlm
  Demonstrates a parametric allocation: Each instance gets three CPU shares per user connection. Very similar to the shares_per_process.wlm example.
- timed_select_scott.wlm
  Demonstrates a response time goal with the SCOTT/TIGER Oracle documentation example tables.
- timed_sys_table.wlm
  Demonstrates a response time goal with a contrived join (database operation) on a pair of V$ Oracle system tables.
- user_cnt_boost.wlm
  Demonstrates a conditional allocation example: Each instance has a fixed minimum allocation and gets a 'boost' if more than ten users connect.

All these example configuration files are available from the "case studies/example configurations" section of the http://www.hp.com/go/wlm site if you would like to see how to fine-tune the resource allocation for your Oracle instances.

HP-UX WLM Pay Per Use Toolkit

HP provides a feature known as Instant Capacity on Demand (iCOD). iCOD offers the ability to activate reserve CPU capacity that is already on your system. You do not pay for the iCOD CPUs until they are activated. You would activate the iCOD CPUs either to increase the active CPU capacity on the system or to replace the CPU capacity of a failing processor on that system. In either case, action can be taken—with no waiting for new CPUs to arrive—because the iCOD CPUs are already on the system.

In addition, HP offers a Pay Per Use (PPU) feature that provides the capacity to support peak anticipated demand, but with payment for the HP server based on actual metered or monitored usage of that capacity. This capacity can be increased or decreased by whole CPUs—as needed.

If you have WLM on either an iCOD system or a PPU system, you can configure WLM to let you know that SLOs are failing and the reserves are needed. When you use the `icod_thresh_pri` keyword in the WLM configuration file, it causes WLM to set the following EMS resource:

/applications/wlm/icod_reserves_needed

This resource will take one of the following values:

ICOD_NEEDED_FALSE iCOD reserves are not available or are not currently required

ICOD_NEEDED_TRUE iCOD reserves exist and, if activated, may reduce the number of SLO failures

WLM provides the Pay Per Use Toolkit (PPUTK) to automate CPU management. Using WLM with PPUTK on either type of system, you can have CPUs automatically activated when SLOs are failing. This is possible because of the `utilitydc` data collector. This data collector monitors the value of the EMS resource cited above. If it is ICOD_NEEDED_TRUE for a certain number of intervals, a CPU is activated. If it is ICOD_NEEDED_FALSE for a certain number of intervals, CPUs may be deactivated when they are no longer needed. For information on CPU deactivation, see the HP-UX Workload Manager Toolkits User's Guide.

WLM's notification through the EMS resource, as well as the activation and deactivation of processors, depends on the version of the iCOD and PPU software, as shown in the following table.

iCOD/PPU version	Notify	Activate	Deactivate
iCOD 4.x	Yes	Yes	No
iCOD 5.x	Yes	Yes	Yes
iCOD 6.x	Yes	Yes	Yes
PPU 4.x	Yes	Yes	Yes
PPU 5.x	No	No	No

Why use PPUTK?

The benefits of using PPUTK with WLM are the ability to:

- Automatically activate CPUs when they are needed to meet critical SLOs on iCOD systems
- Automatically activate and deactivate CPUs as needed to meet critical SLOs on PPU systems
- Optimize use of PPU systems

PPU cost optimization

Using WLM on a PPU system, the system's CPU capacity is increased or decreased automatically. After you set your SLOs in the WLM configuration file, WLM and its Pay Per Use Toolkit (PPUTK) automatically optimize the number of active CPUs to the minimum number of CPUs needed to satisfy the SLOs. With PPU, you pay only for the amount of CPU you use; thus, minimizing the number of active CPUs minimizes your costs.

Tools in PPUTK

This toolkit includes the `utilitydc` tool. `utilitydc` is a data collector for Workload Manager. This collector monitors the EMS resource /applications/wlm/icod_reserves_needed to determine whether CPUs should be activated or deactivated.

How PPUTK works

The following steps outline how PPUTK activates and deactivates CPUs in response to your SLOs:

1. WLM sets the EMS resource /applications/wlm/icod_reserves_needed only when you specify the `icod_thresh_pri` keyword in the WLM configuration file:

   ```
   icod_thresh_pri = x;
   ```

   The value *x* indicates that all SLOs with priorities from 1 to *x* (inclusive) should be monitored. If any SLOs in this range fail, the EMS resource is set to ICOD_NEEDED_TRUE.

   Use the `icod_filter_intervals` keyword to have WLM wait *y* consecutive WLM intervals before changing the value of the EMS resource:

   ```
   icod_filter_intervals = y;
   ```

   This feature allows WLM to ignore spikes or anomalies in utilization (over or under) and, hence avoids unnecessary CPU activation and/or deactivation.
2. The PPUTK data collector `utilitydc` monitors the EMS resource and possibly activates or deactivates a CPU

With the new number of CPUs, CPU allocations for workload groups with active SLOs are affected. For example, a workload that needs about 50% of a 4-CPU server (about 2 CPUs) will need 40% of a 5-CPU server.

How to use PPUTK

To use PPUTK:

1. Ensure your WLM configuration file uses the `icod_thresh_pri` keyword
2. Use the `icod_filter_intervals` keyword to set the number of consecutive WLM intervals that must pass before the value of the EMS resource that indicates reserves are needed is changed
3. Determine whether the system is a PPU system or an iCOD system by using the `icod_stat` command
4. Activate the `utilitydc` data collector with the option that specifies whether the system is PPU or iCOD
5. Disable WLM's `distribute_excess` feature if you explicitly or implicitly use `utilitydc`'s prm deactivation algorithm (the `distribute_excess` feature affects whether excess resources go to your workload groups or to the default workload group)
6. Create one or more SLOs to use the `utilitydc` metric

Example configuration files

PPUTK comes with the following example WLM configuration files:

- minimalist.wlm
  Demonstrates a basic configuration using `utilitydc`. The system will adjust the number of active processors based on the system load average.
- tune-by-cpucount.wlm
  Demonstrates how to use the metric returned by `utilitydc` (the number of active CPUs in a system) to enable different SLOs and to control tunable parameters.

These two example configuration files are available from the "case studies/example configurations" section of the http://www.hp.com/go/wlm site if you would like to see how to take advantage of WLM's PPU integration.

HP-UX WLM Apache Toolkit

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 version 2.x.

Why use ApacheTK?

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

ApacheTK includes a white paper and example configuration files that demonstrate resource separation of Apache-based workloads. It also includes the simple data collector time_url_fetch, which uses the Apache benchmark tool `ab` to time URL response times.

Also, ApacheTK provides wlm_watch.cgi, which provides a web-based view of the standard `prmmonitor`, `prmlist`, and other WLM and PRM command-line tools. These are view-only reports—no modifications are allowed from the CGI script.

How ApacheTK works

ApacheTK's focus is exploring several scenarios of how to use WLM with Apache-based workloads. These scenarios, based on the WLM model of workload separation, are developed in a white paper that includes example configuration files and explains various related tools.

How to use ApacheTK

The best way to use ApacheTK is to read the white paper "Using HP-UX Workload Manager with Apache" available from the "information library" at www.hp.com/go/wlm. 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
- 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 CPU allocation by performance goal

Example configuration files

ApacheTK provides example configuration files for WLM and even Apache in some cases:

- apache_vs_oradb.wlm
  WLM configuration file to demonstrate separating an Oracle database instance workload from Apache and its children.
- apache_vs_batch.wlm
  WLM configuration file to demonstrate separating a batch job workload from Apache and its children.
- apache_vs_single_cgi.wlm
  WLM configuration file to demonstrate separating a single, resource-intensive CGI workload from Apache.
- apache_vs_tomcat.wlm
  WLM configuration file to demonstrate separating the Tomcat JVM from Apache.
- 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, start_servlet_jail.sh, stop_ servlet_jail.sh
  Apache and WLM configurations 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.

HP-UX WLM WebLogic Toolkit

WLM can help you manage and prioritize WebLogic Server workloads through the use of the WLM BEA WebLogic Toolkit (WebLogicTK).

Why use WebLogicTK?

Using WLM with WebLogic you can move CPUs to or from WebLogic Server 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.

Tools in WebLogicTK

WebLogicTK includes a white paper and example configuration files that demonstrate resource separation of WebLogic-based workloads. It also includes the data collector `wlmwlsdc`, which tracks metrics for WebLogic instances. Finally, WebLogicTK provides the `expsmooth` utility for smoothing values from WLM data collectors by computing a running average in which the importance of old values diminishes (decays) over time.

how WebLogicTK works

The WebLogicTK data collector `wlmwlsdc` allows you to track metrics indicating how busy WebLogic Server instances are. You then use those metrics to create SLOs that adjust the CPU allocations for the associated workload groups.

How to 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 the "information library" at www.hp.com/go/wlm. The paper guides you through example WLM configurations that show how to manage various types of WebLogic Server instances. For example, the paper shows you how to:

- Manually provide a single instance an increasing amount of CPU resources in the form of a dynamic PSET (processor set) for benchmarking
- Separate an instance from other workloads, as well as from other instances, while automatically maintaining performance using a dynamic PSET based on:
  - Group CPU usage
  - Server instance queue metrics

Example configuration files

The following files are example WLM configuration files. Each .wlm configuration file represents a use case documented in the white paper.

- manual_cpucount.wlm
  WLM configuration file to help with benchmarking and capacity-planning tasks. Demonstrates how to 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 assigned more CPUs, to a maximum of four. See the discussion of usage goals in the wlmconf(4) man pages or in the WLM User's Guide.
- 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, its workload group is assigned more CPUs, to a maximum of four. See the discussion of usage goals in the wlmconf(4) man page or in the WLM User's Guide.
- wls_1inst_q_goal.wlm
  WLM configuration file to demonstrate how to control a WebLogic instance. The WebLogic instance's workload group gets 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 gets 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 is nonzero, instA's group gets an extra CPU. If the instA default queue depth is nonzero, the instance's group gets an additional CPU. Lastly, if the instB queue depth is nonzero, the instances' group gets an additional CPU, if any are left.

HP-UX WLM SNMP Toolkit

WLM provides integration with the HP-UX SNMP agent through the WLM SNMP Toolkit (SNMPTK).

Why use SNMPTK?

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

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

One source of metrics is PRM data, available starting at the OID .1.3.6.1.4.1.11.5.4.2.1 (hp.hpSysMgt.hpUXSysMgt.hpPRM.prmReadOnly).

Tools in SNMPTK

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.

How SNMPTK works

SNMPTK is another avenue for providing data to WLM. It pulls values from the HP-UX SNMP agent and sends them to standard out. You can then use the WLM utility `wlmrcvdc` to forward the data to WLM.

How to use SNMPTK

Consider what numeric data is available to you through SNMP and determine whether the data would be useful in driving SLO goals, setting up shares-per-metric allocations, or enabling and disabling SLOs. After determining what data is useful, use `snmpdc` in a `wlmrcvdc` statement in the WLM configuration file to pull the data and forward it to WLM.

Example configuration files

SNMPTK comes with the following example configuration file:

- minimalist.wlm
  Demonstrates a basic configuration using `snmpdc` to provide the number of currently active processes to an SLO.

This example configuration file is available from the "case studies/example configurations" section of the http://www.hp.com/go/wlm site if you would like to see how to use SNMP data in your WLM configurations.

HP-UX WLM Duration Management Toolkit and HP-UX WLM Toolkit for SAS® software

WLM and its Duration Management Toolkit (DMTK) help control CPU resources by granting a critical job only the amount of CPU needed to complete within a certain timeframe. Because the business-critical job is being duration managed, the extra compute resource on the server can be made available to other users or jobs without affecting the managed business-critical job. This translates to more efficient use of existing compute resources. 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.

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

Why use DMTK / SASTK?

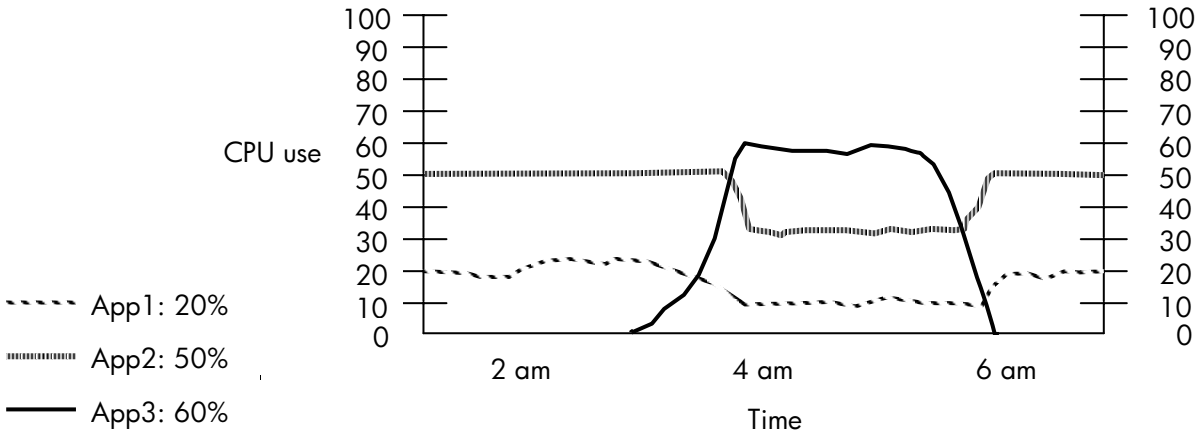
The benefits of using DMTK / SASTK with WLM are:

- 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

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.

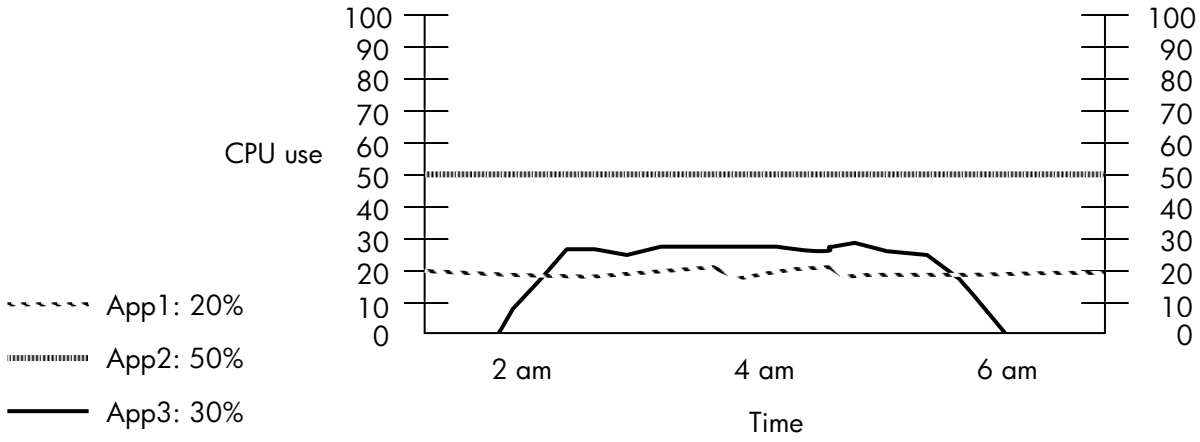
The following figure shows three unrelated applications, App1, App2, and App3, sharing a server without the benefit of duration management. The applications most critical to the business, App1 and App2, run continuously and tend to consume about 20% and 50% of the CPU, respectively. The CPU-intensive 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 3. Unrelated applications without duration management



In the next figure, 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 use their typical 20% and 50% of the CPU.

Figure 4. Unrelated applications with duration management



There is also 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 WLM. It takes the following items as input:
  – PID of the process needing duration management
  – Profile value (CPU time in integer seconds needed by the process to finish with unlimited CPU resources available)
  – Desired duration of the managed process in whole seconds (integer seconds)

  `wlmdurdc` uses the PID to determine how many CPU cycles the process is currently using, the profile value to determine 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 is a SAS macro that is useful in instrumenting SAS jobs to:
  – Get profile data indicating elapsed time for a job
  – Inform `wlmdurdc` of a job's percent completed

How DMTK / SASTK work

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 the HP-UX Workload Manger Toolkits User's Guide. 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, and 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.

How to use DMTK / SASTK

The purpose of DMTK is of course to help an application complete 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:

1. Write a discovery command to identify and correlate each target application with its profile value and desired duration (Many applications' discovery commands can be simple shell scripts. Example discovery commands are included with DMTK and SASTK.)
2. Define WLM workload groups in which to place your target applications
3. Define SLOs for each group
4. Set up `wlmdurdc` invocations in `tune` structures in the WLM configuration file
5. Check the syntax of the configuration file with `wlmd -c` and fix any errors
6. Start WLM
7. Run only an application that will have its duration managed so that it has unlimited CPU access
8. Let the application complete
   Once the application finishes, `wlmdurdc` writes its profile value to its log file.
9. Repeat Steps 7 and 8 for each application that will have its duration managed
10. Use the profile values from syslog
    (`wlmdurdc` writes these values to syslog as the target application runs) Modify your discovery command to use the profile values. Also, now that you have the profile values, you can specify reasonable desired durations.

    If you are running SAS jobs, you can instrument the jobs with the macro `hp_wlmtk_goals_report` to report "percentage complete" to `wlmdurdc` at various points during the job. `wlmdurdc` would then use this information to better tune the CPU allocation for the job's workload group. For more information on this macro, see hp_wlmtk_goals_report(1M).
11. Start the applications in the workload groups using the `prmrun` utility

Example configuration files

DMTK comes with the following example WLM configuration files:

- duration.wlm
  Demonstrates using `wlmdurdc` to monitor a process. The glanceplus toolkit is used to turn the SLO on or off based on the number of active processes in the associated workload group.
- expressconf.wlm
  Demonstrates an express lane configuration where one high-priority group has fixed CPU allocations so the processes in that group finish quickly.
- expressconf_shares.wlm
  Demonstrates an express lane configuration where a group's number of shares depends on the number of processes in the group.
- expressconf_usage.wlm
  Demonstrates an express lane configuration that is based on the actual usage of the CPU in the group.

These example configuration files are available from the "case studies/example configurations" section of the http://www.hp.com/go/wlm site if you would like to see how to manage the duration of your processes.

Summary

HP's Adaptive Enterprise and HP-UX Workload Manager help you use your computing resources more efficiently, allowing you to consolidate applications on a single system, while still guaranteeing performance levels of the critical applications. Combined with the WLM Toolkits, WLM provides concrete building blocks for quickly deploying WLM in Oracle environments, SAS environments, Apache environments, and BEA WebLogic Server environments. This combination also makes it easy to use numeric SNMP data in your WLM configuration. In addition, it offers the ability to optimize CPU usage, and reduce costs in Pay Per Use environments by automatically activating and deactivating CPUs as needed. It also provides duration management, allowing you to smooth out bursts in system usage. Together, WLM and its toolkits deliver the control of your systems' resources you need to maximize the return on your hardware investment.

For more information

For more information on HP-UX Workload Manager or HP-UX Workload Manager Toolkits, 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.

01/2004