Server Consolidation with HP-UX Workload Manager (WLM)

Raja Daoud, Tom Turicchi

Hewlett-Packard Company
3000 Waterview Parkway
Richardson, TX 75080

Tel: (972) 497-4731, (972) 497-4410
Fax: (972) 497-4245
Email: raja@rsn.hp.com, turicchi@rsn.hp.com

Abstract

To guarantee a timely response from mission-critical applications, data centers often assign a dedicated server to each workload. This provides each application the resources it needs to handle peak demand and allows enough room for growth. However, it also results in a large number of servers that need to be managed, with resources that remain idle for significant time periods.

HP-UX Workload Manager (WLM) enables the consolidation of workloads on a smaller number of servers while maintaining the service level objectives (SLOs) of the business. It manages the allocation of system resources to the applications and dynamically reacts to changes in service levels in order to maintain consistent performance. SLOs are prioritized and can be configured to be active only at certain time periods. This gives HP-UX WLM the configuration flexibility to adapt to the data center's schedule.

This paper gives an overview of HP-UX WLM and its operation and presents examples of its use to meet various business objectives.

Introduction

Over the last two decades, data centers that adopted the open systems model grew their capabilities by adding new servers when new functionality was deployed. To guarantee performance, these servers were dedicated to the new applications. They were also sized to provide the extra capacity to handle unplanned peak demand and allow room for future growth. This approach worked well but eventually resulted in a proliferation of servers, increasing the complexity and the cost of managing the data center. In addition, the spare capacity built into the data center was fragmented between the various servers, making it hard to exploit. Often, these spare resources were not utilized.

The recent availability of large or high-end servers provides companies the opportunity to consolidate multiple applications onto a smaller number of servers. This promises to reduce the cost and complexity of managing the data center, and to yield higher resource utilization by reducing the fragmentation of the spare resources. However, for such a deployment to be successful, a mechanism is needed to guarantee that each application receives its appropriate share of the resources, and does not impede on the performance of the other workloads on the server. As an application's load changes throughout the day, its resource allocation also needs to be adjusted in order to maintain the required level of performance. HP-UX Workload Manager (WLM) provides this functionality to enable workload consolidation.

HP-UX WLM Overview

HP-UX WLM is a software product, part of the system management suite for HP 9000 servers. It works in conjunction with HP Process Resource Manager (PRM) to enhance the manageability of the servers and control the allocation of resources among the workloads consolidated on a server.

HP-UX WLM automatically distributes system resources among applications to achieve the performance goals defined by the system administrator. When goals are not being met, the user is notified via an alarm. The user provides HP-UX WLM a configuration file, listing the applications to be monitored, the desired performance goals, their relative importance (priority), and the tools to monitor each application's performance. Each goal may be configured to be actively managed only during specific time periods. A goal is referred to as a Service Level Objective (SLO). In addition to the configuration file, the user provides an executable that measures an application's performance and reports it through API calls. These executables–referred to as performance monitors, or more generically as data collectors–enable HP-UX WLM to determine if workloads are meeting, exceeding, or missing their stated goals. The separate monitors provide users with the flexibility and freedom to define the appropriate performance metrics specific to each application. Periodically, HP-UX WLM reads the recent data provided by the monitors, compares it to the goals specified, and adjusts the distribution of CPU entitlements among workloads accordingly, relying on the goal priorities to resolve excess demand. Figure 1 illustrates how HP-UX WLM operates.

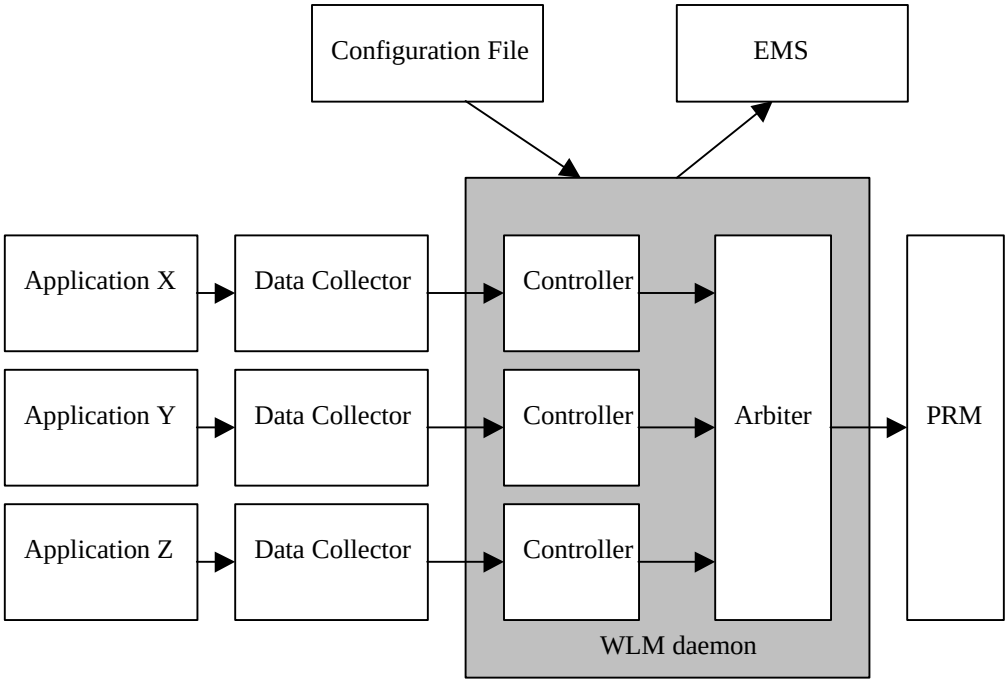

Figure 1: HP-UX WLM Diagram

At startup, the HP-UX WLM daemon reads the configuration file that specifies the applications monitored and the SLOs attached to them. It then spawns the appropriate instances of data collectors for each application performance measure, also referred to as a metric. While the applications are running, the data collectors gather the performance statistics for their metrics and provide them to the WLM daemon. The metric used depends on the type of application and the user's needs. For example, a metric may be a running average of database transaction response times, or job completion times for a collection of batch jobs. At periodic time intervals, the WLM daemon reads the metric values provided by the data collectors and forwards them to their corresponding controllers. A controller instance is created, at startup time, for each SLO goal. The controller compares the metric values read to the goal specified in the configuration file. If the application is under-performing, the controller determines the increase in CPU entitlement to request for it in order to rectify the situation. Likewise, if the application is over-achieving, the controller determines the decrease in CPU entitlement to request for it. The requests generated by all controllers are sent to an arbiter. The arbiter attempts to satisfy as many requests as possible, based on the priority of each SLO goal. If there are not enough resources to satisfy all the requests, some low priority goals receive less than requested by the controller. The arbiter updates the allocation of CPU resources and activates a new PRM configuration. These actions are repeated by the WLM daemon at the next time interval.

This iterative process adapts the allocation of CPU entitlements to the needs of the workloads, tracking the changes throughout the day, always attempting to satisfy the highest priority business goals first. The status of the SLOs, along with configuration and performance information, is periodically sent to the HP Event Monitoring System (EMS). The user can configure EMS–using the System Administrator Manager (SAM)–to select how to be notified when an SLO goal fails to be met. The current version of HP-UX WLM allows users to set fixed memory and disk bandwidth entitlements for workloads, however it does not dynamically modify them.

SLO goals can be configured to be active at selected times or dates. This gives the system administrator the flexibility of adapting the WLM configuration to the schedule of the data center. For example, a performance goal may be specified for a mission-critical application on weekdays, between 7am and 9pm, and a less stringent goal activated at night or on weekends to give more resources to batch jobs. When an SLO goal is not active, it is ignored.

SLO Types

HP-UX WLM supports two SLO types:

- **Goal-based SLO:** This SLO specifies a performance goal for the workload. The user defines a performance metric for the application, and provides a data collector that continuously gathers statistics on this metric and sends the data to the WLM daemon. The daemon automatically changes the CPU allocation for the workloads on the system based on their performance and the priority of their SLOs. For example, a goal-based SLO may specify that average database transactions must complete in less than 3 seconds.
- **Entitlement-based SLO:** This SLO allows the user to specify a fixed CPU entitlement for the workload, and not monitor the performance of some metrics. This SLO does not require a data collector to be provided by the user. However, it implies that the user knows the CPU entitlements sufficient for the workload. This SLO type is suitable for discretionary or optional workloads that coexist on a server.

Both types of SLOs can coexist in a single HP-UX WLM configuration, allowing system administrators to consolidate mission-critical applications that have strict performance requirements, with optional workloads or workloads that are satisfied with fixed resource entitlements. The user may also specify multiple SLOs for the same application, on any number of metrics.

Data Collector API

HP-UX WLM exports a simple API to be used by data collectors to send the metric statistics to the WLM daemon. Figure 2 lists the three functions and the header file to include. The API provides a low-latency mechanism to send data to the daemon. However, it requires that data collectors be developed in C or C++. A future release will include a scripting interface, allowing system administrators to develop data collectors in their favorite scripting language, at the cost of a slightly higher latency for the data transfer.

```
#include <wlm.h>

int wlm_mon_attach ( const char *metric_name );                          Start WLM communication

ssize_t wlm_mon_write ( int handle, const void *buf, size_t nbytes );    Send metric data to WLM

int wlm_mon_detach ( int handle );                                       End WLM communication
```

Figure 2: WLM API

For each metric used in one or more goal-based SLOs, the WLM daemon spawns an instance of the data collector specified by the user. The data collector starts by calling *wlm_mon_attach( )* to establish a communication channel to the WLM daemon. The metric name is found in the WLM_METRIC_NAME environment variable, set by the WLM daemon. Following that, and for the duration of its operation, the data collector gathers statistics on the metric for the workload it monitors, and calls *wlm_mon_write( )* to send the data to the daemon. The first version of HP-UX WLM only accepts metric data of type *double*. Before it exits, the data collector calls *wlm_mon_detach( )* to close its communication channel. Figure 3 shows a code fragment for a data collector.

```
int handle;
double metric_val;

handle = wlm_mon_attach ( getenv ( "WLM_METRIC_NAME" ) );

while ( ! is_done ( ) ) {
        metric_val = gather_workload_stats ( );
        wlm_mon_write ( handle, &metric_val, sizeof ( metric_val ) );
}

wlm_mon_detach ( handle );
```

Figure 3: Data Collector Code Fragment

Configuration File

The HP-UX WLM configuration is specified in an ASCII text file. HP-UX WLM relies on the PRM product to enforce the resource allocation it specifies. PRM allocates resource shares to groups of processes referred to as PRM groups. PRM groups are independent of the UNIX process groups used for job control. This grouping mechanism allows users the flexibility of controlling multiple application processes as a single entity. However, it requires the user to plan the mapping of application processes to PRM groups and to describe this mapping in the WLM configuration file. Applications are typically specified by the name, or pathname, of their executables. Once a configuration file is created, the *wlmd(1)* command can be used to check its validity before activating it. Figure 4 lists the command lines used to check then activate a configuration.

% wlmd –c config_file	Check validity of configuration file (any user)
% wlmd –a config_file	Activate the configuration (requires *root* permissions)

Figure 4: Main WLM Daemon Command Options

To present the syntax of the configuration file, several examples are given below for different user-case scenarios.

Fixed CPU Entitlements

It is possible to use HP-UX WLM to set up a configuration with a fixed CPU distribution, and date-driven SLO activation. In this example, the system runs multiple applications, two of them must be guaranteed some CPU shares. The */opt/stocks/bin/trading* application runs during the workweek and handles stock trading. It has the highest priority and requires a minimum of 60% CPU share. The */opt/stocks/bin/histdata* application runs all the time and handles requests for historical stock data. It has a lower priority and requests a minimum of 50% CPU share. To control them separately, each application is mapped to a different PRM group, *stocks* and *history* respectively. Other applications on the system are mapped to the default *OTHERS* group, created automatically by HP-UX WLM.

Figure 5 shows the configuration file for this scenario. It defines two *slo* structures, specifying the priority, PRM group, and minimum and maximum CPU shares requested for the two controlled applications. The stock trading application is given the higher priority of 1, and restricted to be active between Monday and Friday. The PRM groups and application executables are defined in the *prm* structure. The groups are given the IDs 2 and 3 because group IDs 0 and 1 are reserved by PRM. Note that neither *mincpu* nor *maxcpu* need to sum up to 100%.

During the workweek, both SLOs are active. Because they do not have a goal defined, HP-UX WLM treats them as if they were each requesting their *mincpu* values. Thus *stock_trading* requests 60% and, being the highest priority SLO, it is granted the 60 CPU shares. The *historical_data* SLO requests 50%, but can only be given 39 shares, because the default *OTHERS* group must be given at least 1 CPU share. On weekends, *stock_trading* is inactive and gets a nominal 1 CPU share. The *historical_data* SLO gets the 50 shares requested. The remaining 49 shares are given to the *OTHERS* group.

The *maxcpu* value is not currently used by entitlement-based SLOs. However, its specification enables future enhancements to the resource distribution algorithm to make better use of this information.

```
#
# Configuration for two applications with fixed CPU shares.
# Condition used to activate one application Mon-Fri only.
#
slo stock_trading {
    pri = 1;
    entity = PRM group stocks;
    mincpu = 60;
    maxcpu = 90;
    condition = Mon - Fri;
}

slo historical_data {
    pri = 2;
    entity = PRM group history;
    mincpu = 50;
    maxcpu = 80;
}

prm {
    groups = stocks : 2, history : 3;
    apps = stocks : /opt/stocks/bin/trading, history : /opt/stocks/bin/histdata;
}
```

Figure 5: Fixed CPU Entitlements

Independent Goal-Based SLOs

Figure 6 extends the previous case by adding performance-driven goals to the SLOs. In this version of HP-UX WLM, only user-defined goals are supported, thus the *udef* goal-type specifier. The highest priority goal is for stock trading transactions to complete within 3 seconds. The second goal is for historical data requests to complete within 10 seconds. In this example, we assume that the data collectors for both SLOs are reporting transaction times in seconds. For each metric used in a goal expression, a *tune* structure is defined, specifying the value of the *coll_argv* tunable. This tunable gives HP-UX WLM the command-line of the data collector to spawn for each metric.

During the workweek, the performance data collected for the stock trading SLO determines how many CPU shares its application gets. The data collected for the historical data SLO determines how many of the remaining shares its application gets. The *OTHERS* group gets any shares left beyond its nominal 1 CPU share. On weekends, the stock trading SLO gets 1 CPU share, and *OTHERS* gets the shares left after satisfying the requests of the active SLO.

```
#
# Configuration for two applications with user-defined performance goals.
#
slo stock_trading {
    pri = 1;
    entity = PRM group stocks;
    goal = udef stock_trans < 3;
    mincpu = 60;
    maxcpu = 90;
    condition = Mon - Fri;
}

slo historical_data {
    pri = 2;
    entity = PRM group history;
    goal = udef hist_trans < 10;
    mincpu = 50;
    maxcpu = 80;
}

tune stock_trans {
    coll_argv = /opt/stocks/bin/perf_trading -o logfile;
}

tune hist_trans {
    coll_argv = /opt/stocks/bin/perf_histdata -f 20;
}

prm {
    groups = stocks : 2, history : 3;
    apps = stocks : /opt/stocks/bin/trading, history : /opt/stocks/bin/histdata;
}
```

Figure 6: Independent Goal-Based SLOs

Stretch Goal SLO

Figure 7 extends the previous case by adding a stretch goal to the stock trading SLO. The stretch goal is given a lower priority than the historical data SLO. It is also active in the workweek between 8AM and 4PM (until the end of the 15:59 minute). While active, any CPU shares unused by the historical data SLO are given to the stretch goal in an attempt to improve the performance of the stock trading transactions from 3 seconds down to 1 second. HP-UX WLM makes it possible to define stretch goals by allowing multiple SLOs to be specified using the same metric.

```
#
# Configuration for two applications with user-defined performance goals.
# The stock trading application has a stretch-goal defined.
#
slo stock_trading {
    pri = 1;
    entity = PRM group stocks;
    goal = udef stock_trans < 3;
    mincpu = 60;
    maxcpu = 90;
    condition = Mon - Fri;
}

slo historical_data {
    pri = 2;
    entity = PRM group history;
    goal = udef hist_trans < 10;
    mincpu = 50;
    maxcpu = 80;
}

slo stock_trading_stretch {
    pri = 3;
    entity = PRM group stocks;
    goal = udef stock_trans < 1;
    mincpu = 60;
    maxcpu = 90;
    condition = (Mon - Fri) && (8:00 - 15:59);
}

tune stock_trans {
    coll_argv = /opt/stocks/bin/perf_trading -o logfile;
}

tune hist_trans {
    coll_argv = /opt/stocks/bin/perf_histdata -f 20;
}

prm {
    groups = stocks : 2, history : 3;
    apps = stocks : /opt/stocks/bin/trading, history : /opt/stocks/bin/histdata;
}
```

Figure 7: Stretch Goal SLO

Fixed Disk Bandwidth Entitlements

Figure 8 extends the example in figure 6 by adding fixed disk bandwidth entitlements to ensure that workloads receive appropriate shares of the disk bandwidth. The disks must be under the control of the Logical Volume Manager (LVM). The disk bandwidth entitlement of a group can be specified for a logical volume group. It must be specified for all groups or for none, and the sum of the disk entitlements for a logical volume group

must equal 100%. Because HP-UX WLM creates the default group *OTHERS* if it is not specified, this restriction requires the user to define *OTHERS* and assign it a share of the disk bandwidth for each logical volume group used. The *OTHERS* group must be given the reserved group ID of 1. HP-UX WLM also requires that any group named in the *prm* structure be given an SLO. Thus an SLO is specified for *OTHERS*, and given a very low priority, no goal, and unrestricted CPU values. The */dev/vg01* bandwidth is distributed among the groups, giving the stock trading application 70 shares, the historical data retrieval application 29 shares, and 1 share to applications running in the *OTHERS* group.

```
#
# Configuration for two applications with user-defined performance goals and fixed disk
# bandwidth entitlements.  OTHERS must be defined as well to assign it a share of the disk.
#
slo stock_trading {
    pri = 1;
    entity = PRM group stocks;
    goal = udef stock_trans < 3;
    mincpu = 60;
    maxcpu = 90;
    condition = Mon - Fri;
}

slo historical_data {
    pri = 2;
    entity = PRM group history;
    goal = udef hist_trans < 10;
    mincpu = 50;
    maxcpu = 80;
}

slo others {
    pri = 1000;
    entity = PRM group OTHERS;
    mincpu = 1;
    maxcpu = 100;
}

tune stock_trans {
    coll_argv = /opt/stocks/bin/perf_trading -o logfile;
}

tune hist_trans {
    coll_argv = /opt/stocks/bin/perf_histdata -f 20;
}

prm {
    groups = OTHERS : 1, stocks : 2, history : 3;
    apps = stocks : /opt/stocks/bin/trading, history : /opt/stocks/bin/histdata;
    disks = OTHERS : /dev/vg01 1, stocks : /dev/vg01 70, history : /dev/vg01 29;
}
```

Figure 8: Disk Bandwidth Entitlements

Specifying Dates and Times

The optional *condition* and *exception* keywords specify logical expressions using dates and date ranges. The SLO is active when the *condition* is TRUE and the *exception* is FALSE. For example:

condition = Mon - Fri;

has the same effect as:

exception = Sat - Sun;

They can both be used in the same SLO, one restricting the scope of the other. For example:

condition = 8:00 - 15:59;
exception = Sat - Sun;

has the same effect as:

condition = (Mon - Fri) && (8:00 - 15:59);

A date has 3 optional components:

- The day of the week: Mon, Tue, Wed, Thu, Sat, Sun.
- The date, in the form month/day/year. The year must be a 4-digit number.
- The time, in the form hour:minutes. The hour uses the 24-hour notation, where 10:00 is 10am, and 22:00 is 10pm.

The month, day, year, hour, and minutes fields can also be set to the wildcard character * (asterisk) to indicate that all values are accepted. Some examples are given below:

```
condition = Wed;                 # on Wednesdays
condition = 3/13/1999 - 4/15/1999;  # March 13 to April 15, 1999
condition = Mon 8:00 - Fri 16:59;   # from Mon @ 8am to Fri @ 5pm
condition = 4/*/1999;            # all of April 1999
condition = */10/*;              # the 10th of each month
condition = *:10;                # at 10 min past the hour
```

Conclusion

HP-UX WLM is a new tool designed to help data centers consolidate workloads and reduce the number of servers to manage. It provides the means to specify and track business goals for mission-critical applications. It maintains consistent performance by automatically adjusting resource allocation in reaction to changes in service levels. Its flexible date range specification handles various data center schedules.

References

- **HP-UX Workload Manager User's Guide,** Version 1.0, April 2000, http://www.docs.hp.com/hpux/network/#hpuxworkdoc

- **HP Process Resource Manager User's Guide,** Version 1.07, December 1999, http://docs.hp.com/hpux/ha

- **Workload Performance Monitor Considerations,** February 2000, http://resourcemanagement.unixsolutions.hp.com/WaRM/docs/perfmon.html

- **HP-UX Workload Manager 1.0 Configuration,** December 1999, http://resourcemanagement.unixsolutions.hp.com/WaRM/docs/config.html

- **HP-UX Workload Manager,** February 2000, http://www.unixsolutions.hp.com/products/sys_mgmt/wlm_wp.html

- **http://www.hp.com/go/wlm**