Meeting Performance Goals with the HP-UX Workload Manager

(an extended abstract)

Indira Subramanian
Hewlett Packard Co.
Cupertino, CA
indira@cup.hp.com

Cliff McCarthy
Hewlett Packard Co.
Richardson, TX
mccarthy@rsn.hp.com

Michael Murphy
Hewlett Packard Co.
Richardson, TX
mmurphy@rsn.hp.com

The HP-UX Workload Manager helps workloads meet user-specified performance goals by dynamically adjusting their access to resources such as CPU. We implemented this workload manager as a part of a *feedback control system*, using existing resource control and performance instrumentation infrastructure.

1. Introduction

Successful consolidation of multiple workloads on to a single server demands that users be guaranteed consistent levels of workload performance. Users should be able to define Service Level Objectives (SLO), specifying the performance goals they seek and their relative importance. To achieve target performance consistently, applications' access to resources such as CPU and memory must be adjusted automatically.

Workload managers can be classified in to two categories. An *entitlement-based* manager allocates resources based on a specification of resource entitlements. *Goal-based* workload managers adjust the resources allocated to a workload, based on a specification of performance goals.

Entitlement-based and goal-based workload managers have been supported in some commercial and experimental systems. Several UNIX OS vendors implement entitlement-based resource managers[9, 5, 4], which do not use any feedback mechanism to meet performance goals. IBM's OS/390 goal-based workload manager (WLM) employs extensive instrumentation to gather detailed information about an application's resource needs, and adjusts resource allocations[1]. Adjusting the use of system resources to meet response time goals has been in wide use in Transaction Processing (TP)[2]. Several experimental transaction processing systems have exploited feedback mechanisms to meet response time goals[7, 8].

The HP-UX Workload Manager is distinct from the systems discussed above that also use feedback control to adjust resources. First, the workload manager does not monitor workload performance directly. Instead, it receives a workload's performance data through an API from a performance monitor created by the application provider (or system administrator). Second, the workload manager uses a simple proportional controller to determine the resources that must be allocated to a workload. Third, the workload manager has been designed to take advantage of the existing infrastructure of tools that includes Process Resource Manager (PRM), and the Event Monitoring Service (EMS, which raises an alarm when a performance goal is not being met). Fourth, unlike TP monitors, which handle workload management exclusively for transaction processing systems, the HP-UX Workload Manager can handle a wide variety of application workloads.

2. Background

The HP PRM (Process Resource Manager), enables a system administrator or the workload manager to control the resources allocated to a workload[3, 4]. PRM is tightly integrated with the HP-UX kernel, and is supported through enhancements to certain HP-UX system calls and commands. PRM monitors resource usage, and it can guarantee a minimum entitlement of CPU, memory, and disk bandwidth available to a group of processes (a PRM group, or a resource group). PRM can also enforce capping (upper bound) of CPU and memory allocated to a resource group. The PRM scheduler selects (to run) a group with larger CPU entitlement, more frequently than other groups. The PRM interface allows an administrator to specify the group to which a user belongs. All processes in a group share the resources assigned to that group. Within a group, standard HP-UX resource management policies are applied.

The HP-UX Workload Manager enhances PRM in two ways. First, by giving each workload only the resources that are needed to meet its goal, excess capacity is shared efficiently across workloads. Second, a higher priority workload gets the resources it needs, before the needs for lower priority workloads are met. This priority is distinct from the UNIX process priority. WLM uses this priority to resolve resource entitlement conflicts, when the resource needs of all the workloads cannot be met. A PRM group with higher CPU entitlement is selected to run more frequently than other groups. The standard HP-UX scheduler employs the UNIX process priority to schedule processes from the selected PRM group.

3. System Overview

The key to developing a successful workload manager is recognizing the fact that it is part of a closed feedback loop control system [6]. The output of the control system is the new set of resources (*entitlement value*) that must be allocated for the workload. To set the new entitlement value, the workload manager enlists PRM. Inputs to the WLM controller are the SLO, priority, and the measured performance. The system administrator specifies the goals and priorities for the workloads on the system, in a simple text file (the WLM configuration file). To obtain performance measurements, the system administrator specifies a performance monitor program. Each workload with a SLO goal, must have a performance monitor associated with it. The WLM daemon starts up this performance monitor process, which communicates subsequently with WLM through a simple API. The WLM controller wakes up periodically, subtracts the goal value from the most recent performance number, multiplies this result by a proportionality constant (`cntl_kp`), and adjusts the entitlement by this amount.

For the workload's performance to converge towards the goal, we require that the relationship between entitlement and performance be monotonic. We choose to ignore a variety of other factors that might influence performance. However, this oversimplification allows us to analyze the characteristics of the workload in useful ways by focusing on the average behavior.

4. Tuning the System

To achieve effective automatic control of workload performance with WLM, three tunable parameters are supported: `wlm_interval`, `cntl_kp`, and `cntl_margin`. `Wlm_interval` controls how often WLM wakes up, checks performance data, and makes entitlement adjustments. WLM makes the change request based only on the latest value reported by the performance monitor. `Cntl_kp` controls how big an adjustment is made (to the entitlement) in response to a deviation from the service level goal. This tunable is discussed in detail in the next paragraph. `Cntl_margin` is used to specify a safety margin around the service level. A safety margin is used to decide when operators are to be notified.

Eight factors help determine a suitable starting value for `cntl_kp`. These factors and their values represent the workload characteristics and the desirable rate of convergence to the workload's performance goal. For this discussion, we consider a transaction processing workload, where the performance goal is that the average completion time for a transaction be under `t` seconds. Four of the factors that determine `cntl_kp` are upper bound (U seconds) and lower bound (L seconds) on service levels, and the corresponding resource entitlements Eu (% CPU) and El (% CPU) respectively. The fifth factor is G (seconds) , the goal value for SLO. We also need to consider the rate of convergence, that is, how long we are willing to wait to get within a certain range of the goal. The range A is expressed as a fraction of G, and T (seconds) specifies the time to get within this range. The eighth factor is J – the average number of seconds between entitlement changes for the workload. Given these parameters, a reasonable initial estimate for `cntl_kp` can be obtained. The expression for calculating `cntl_kp` written using the natural logarithm (base `e`):

$$\texttt{cntl\_kp} = -ln(\frac{AG}{\|U-L\|})\|\frac{J(Eu-El)}{T(U-L)}\|$$

Because of the assumptions made in the calculations that led to this formula, this initial value of `cntl_kp` may need fine-tuning to produce the desired behavior. However, it serves as a reasonable starting point. This initial value must be adjusted based on actual measurements of the time it takes to converge to the response time goal.

5. Summary

The HP-UX Workload Manager employs a feedback control system to dynamically adjust CPU resources allocated to a workload. A future version will handle other resources including memory and I/O. The workload manager uses the existing infrastructure of resource control, performance instrumentation, and event monitoring tools.

References

[1] J. Aman, C. Eilbert, D. Emmes, B. Yocom, and D. Dillenberger. Adaptive algorithms for managing distributed data processing workload. *IBM Systems Journal*, 36(2), 1997.

[2] J. Gray and A. Reuter. *Transaction Processing: Concepts and Techniques*. Number ISBN 1-55860-190-2. Morgan Kaufmann Publishers, San Francisco, CA, 1993.

[3] Hewlett-Packard Company. *HP-UX Process Resource Manager*, May 1999. Technical White Paper.

[4] Hewlett-Packard Company. *HP-UX Process Resource Manager User's Guide*, December 1999.

[5] International Business Machines Corporation. *AIX 4.3.3 Workload Manager*, February 2000. Technical White Paper.

[6] B. C. Kuo. *Automatic Control Systems*. Number ISBN 0-13-304759-8. Prentice Hall, Upper Saddle River, NJ, 1995.

[7] Markus-Sinnwell and A. C. Konig. Managing distributed memory to meet multiclass workload response time goals. In *15th. International Conference on Data Engineering*, March 1999.

[8] E. Rahm. Goal-oriented performance control for transaction processing. In *9th. ITG/GI MMB97 Conference*. VDE-Verlag, 1997.

[9] Sun Microsystems. *Solaris Resource Manager [tm] 1.0: Controlling System Resources Effectively*, 2000. http://www.sun.com/software/white-papers/wp-srm.