RC23835 (W0512-086) December 19, 2005
Computer Science

IBM Research Report

Managing Peak System-level Power with Feedback Control

Xiaorui Wang
Washington University in St. Louis

Charles Lefurgy, Malcolm Ware
IBM Research Division
Austin Research Laboratory
11501 Burnet Road
Austin, TX 78758

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Managing Peak System-level Power with Feedback Control

Xiaorui Wang
Washington University in St. Louis
wang@cse.wustl.edu

Charles Lefurgy, and Malcolm Ware
IBM Research
{lefurgy,mware}@us.ibm.com

Abstract

Power consumption has arguably become the most important design consideration for modern, high-density servers, but current power management implementations have not evolved beyond primitive responses to thermal emergencies and do not manage to varying power and cooling constraints.

We present a technique that manages the peak power consumption of a high-density server by implementing a feedback controller that uses precise, system-level power measurement to periodically select the highest performance state while keeping the system within a fixed power constraint. A control theoretic methodology is applied to systematically design this control loop with analytic assurances of system stability and controller performance, despite unpredictable workloads and running environments.

This technique is particularly valuable when applied to servers with multiple power supplies, where a partial failure of the power supply subsystem can result in a loss of performance in order to meet a lower power constraint. Conventional servers use simple open-loop policies to set a safe performance level in order to limit peak power consumption. We show that closed-loop control can provide a more graceful degradation of service under these conditions and test this technique on an IBM BladeCenter HS20 server. Experimental results demonstrate that closed-loop control provides superior application performance compared to open-loop policies.

1. Introduction

As modern enterprise data centers continue to increase computing capabilities to meet their growing business requirements, high-density servers become more and more desirable due to space considerations and better system management features. However, the greatest immediate concerns about high-density servers are their power and cooling requirements, imposed by limited space inside the server chassis. The situation is even worse for servers which require redundant power supplies for increased reliability. In conventional products, system designers must provide for worst-case operating environments and workloads which result in servers with over-provisioned power supplies that can far exceed the power capacity required for the typical operation. This leads to some potential problems:

1. In the case of a partial power-supply failure, the system degrades performance assuming all the server components are consuming their worst-case power. This pessimistic estimation unnecessarily results in low application performance.
2. An over-provisioned power supply increases manufacturer cost for the server and may increase the footprint of the server.
3. In the case of blade servers, while a chassis is designed to last for several product generations, eventually new blade designs with higher power consumptions may exceed the older chassis' power capacity. Conservative solutions may be forced to use the new blades, but at a fixed, lower performance setting.
4. Data center operators must typically assume a cooling infrastructure that meets the label power of the power supplies, regardless of actual server power consumption, which adds cost to their operating expenses.

Each situation above would benefit from run-time measurement and control of power to adapt to a power constraint and reduce server performance only when actual power consumption exceeded the power constraint. This paper specifically focuses on the first problem to provide a graceful degradation of performance during a power supply fault.

Modern microprocessors commonly offer programmable performance states, such as clock throttling or dynamic voltage scaling (DVS), that can reduce server power consumption. In this paper, we propose to manage the peak system-level power consumption with a feedback controller that uses precise system-level power measurements to periodically select the highest performance state and yet keep the server within the desired power constraint. This allows fixed, design-time constraints to be relaxed because the system safely adheres to run-time constraints.

This paper makes the following contributions:

1. To our knowledge, we are the first to demonstrate managing the peak system power of a single server to a power constraint using precision measurement with a closed-loop control system. This differentiates our work from

previous solutions that manage average power, use ad-hoc control, or use estimations of power in place of real measurement.

2. We present a novel control design based on feedback control theory to manage system-level power with theoretic guarantees on accuracy and stability.
3. We demonstrate how to derive controller parameters such that the controlled system is guaranteed to achieve the desired controller performance in the presence of run-time variations that cause the system to behave differently from the control model.
4. We implement our control system directly in an IBM BladeCenter blade server and evaluate it using industry standard benchmarks.
5. For servers with multiple power supplies, we show that performance degrades much more gracefully under a partial power supply failure as compared to simpler open-loop solutions. In some situations, performance is not noticeably affected.

In the rest of the paper we first highlight the distinction of our work by discussing related work. In Section 3, we discuss system-level management of power in conventional systems and those with feedback controllers. We then demonstrate how we design and analyze the controller based on feedback control theory in Section 4. In Section 5, we describe the detailed implementation of each component in the feedback control loop. Extensive experimental results are presented in Section 6 and we draw conclusions in Section 7.

2. Related work

Power consumption is one of the most important design constraints for high-density servers. Much of the prior work has attempted to reduce power consumption by improving the energy-efficiency of individual server components [1]. In contrast, our paper is focused on providing an effective power management algorithm to control system-level power. Previous work [2] has shown that processors are often the dominant consumers of power in servers. This is particularly true in dense blade server environments. In this paper, we focus on controlling system-level power consumption by throttling processors.

Many researchers use expensive power measurement equipment to instrument servers for their studies [2]. In our work, we use an inexpensive, yet highly accurate, power measurement circuit built-in to recent IBM HS20 blade servers which measures power consumed by the entire blade. This enables our technique for power management to be used in ordinary, high-volume servers.

There has been much work done on system-level power management. Zeng et al. [3] and Lu et al. [4] have developed power management strategies for operating systems. In contrast, our work is at the system-architecture level. Our feedback controller in the service processor firmware directly controls the main host processors to keep the system-level power within a power constraint, while requiring no support from the OS or workloads running on the system and is operational during system boot. Thus, the power management is more robust and less susceptible to software errors or malicious threats.

Feedback control theory has proven to be an effective way in improving performance and robustness of computing systems [5]. Skadron et al. [6] use control theory to dynamically manage the temperature of microprocessors. Likewise, Wu et al. [7] manage power using dynamic voltage scaling by controlling the synchronizing queues in multi-clock-domain processors. In contrast to their work, we control peak power for a whole server instead of just the processors.

Minerick et al. [8] develop a feedback controller for managing the average power consumption of a laptop to prolong battery lifetime. Their study relies on experiments to find the best control parameters. In contrast, we derive parameters based on a systematically built control model. In addition, we not only design our controller based on feedback control theory, but also analytically model the possible system variations and provide corresponding theoretic guarantees. We believe our work is the first to provide such insightful analyses for system-level power management. As a result, our control method does not assume any knowledge about potential workloads and thus can be generally applied to any server system. In addition, our controller is designed to meet the tighter real-time constraints for the overload condition of server power supplies. Femal et al. [9] present a two-level framework for controlling cluster-wide power. The *Local Power Agent* (LPA) applies the controller from Minerick et al. to each server in order to limit the server-level power. The *Global Power Agent* dynamically re-allocates the power budgets between the local managers. Our blade server prototype could be used in place of the LPA to control cluster-wide power.

Sharma et al. [10] effectively apply control theory to control application-level quality of service requirements. Chen et al. [11] also develop a controller to manage the response time in a server cluster. Although they both use control theory to manage power consumption, power is only used as a knob to control application-level service metrics. As a result, they do not provide any absolute guarantee to the power consumption of a computing

system. In this paper, we explicitly control the power consumption itself to adhere to a given power constraint.

3. System-level power management

This section describes portions of the current power management solution in the BladeCenter and the requirements that the control loop must meet in order to satisfy power supply constraints.

3.1. BladeCenter test platform

Our test platform is an IBM BladeCenter HS20 blade server with Intel Xeon microprocessors. The power management architecture of BladeCenter is shown in Figure 1. A BladeCenter chassis has two power domains and is configured with four 2000 W power supplies total. Each power domain is redundantly connected to two of the power supplies so that in the event of a single supply failure, the domain continues operating with the remaining power supply. The first power domain provides power for six processor blades as well as supporting components shared by the blades including management modules, fans, the media tray, and network switches. The second power domain holds eight processor blades. Our discussion and experiments use the second power domain because its blades have a stricter power constraint.

The BladeCenter allows the user to specify the policy that is used when a single redundant power supply fails. One common policy is *redundant*. This allows the domain to consume at most 2000 W. When a power supply fails the remaining supply can provide 2000 W and the blades can consume full power and run at full performance. Another common policy setting is *recoverable oversubscription* [14], which is the focus of this discussion. This policy allows the user to install additional high-power blades which will use more power at full-performance than one power supply can provide, but less power than two power supplies can provide together. The blades run at full-performance as long as both power supplies function. When a single power supply fails, the blade's processors are throttled to a predetermined performance setting so their aggregate power is less than a single power supply. The blades return to full performance once the failed power supply is replaced. The BladeCenter management module determines the unique performance setting for each blade when the blade is initially installed in the chassis. The setting is based upon installed blade components and the capacity of the power supplies.

In BladeCenter, the *recoverable oversubscription* policy specifies that the maximum power that two functioning 2000 W power supplies can provide is 2500 W [15]. This is the power supply overload condition. During a single power supply failure, the power load shifts to the remaining supply. An engineering decision in BladeCenter is that the load must be reduced from a maximum of 2500 W down to 2000 W within one second

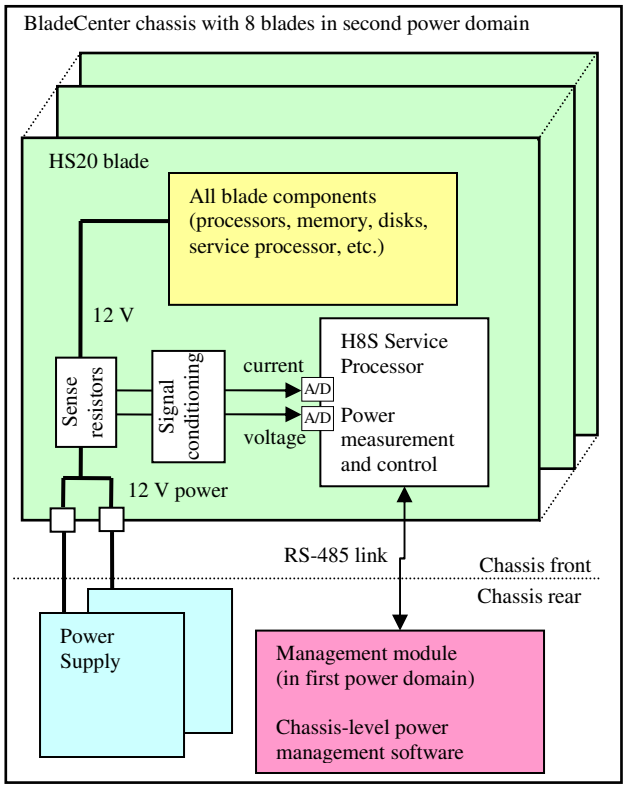


Figure 1: BladeCenter chassis.

[15]. If the load remains too high on the single supply for too long, then the remaining power supply may turn off and remove power from all blades in the domain. In reality, the one second target is conservative and the power supply can sustain a power overload for even longer periods of time.

The mechanism to throttle blade performance is processor clock modulation ("clock throttling") which lowers the effective frequency of the processors. There are 8 performance states which correspond to effective frequencies of 12.5%, 25%, 37.5%, 50%, 62.5%, 75%, 87.5%, and 100%.

3.2. Feedback control of power

We have developed a feedback control loop which adaptively controls the power consumption of the server by manipulating the processor clock modulation setting. Some form of clock modulation is commonly available across all microprocessor families used in servers, while DVS is not supported in all server products. Therefore, we developed a closed-loop controller using clock throttling, the more general mechanism, so that our technique can potentially be used across all server platforms. DVS will be a more energy-efficient mechanism to use for future blade power control. However, DVS requires a different design for the closed-loop controller, which is beyond the scope of this paper.

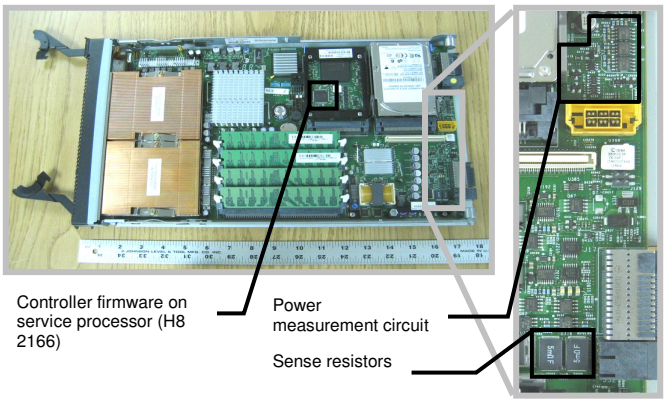


Figure 2: IBM BladeCenter HS20 blade.

There are two reasons for us to use processor throttling as our actuation method. First, processors typically have well-documented interfaces to adjust performance levels. Second, processors commonly contribute the majority of total power consumption of small form-factor servers. As a result, the processor power difference between the highest and lowest performance states is large enough to compensate for the power variation of other components and to provide an effective way to support a power budget reduction in the event of power supply failure. The control loop always sets all processors to the same performance state. Asymmetric throttling would violate the assumption made by many commercial operating systems that all processors run at the same speed.

During normal operation, the power budget for the control system on each blade is 312 W (2500 W / 8 blades). When a power supply fails, each blade simultaneously detects the event and changes its power budget to 250 W (2000 W / 8 blades). The control systems respond to this new budget and reduce the load on the remaining supply within one second.

The key components in the control loop include the monitor, the controller, and the actuator. The control loop is invoked periodically and its period is decided based on the trade-off between actuation overhead and system settling time. At each control period, a precision measurement of the real system-level power consumption is input to the controller. The controller computes the new performance state and sends it to the actuator. The actuator throttles the processors to the new performance state. A detailed description of each component is given in Section 5. The photo in Figure 2 shows our HS20 blade with the power measurement circuitry and sense resistors used for closed-loop control.

4. Controller Design and Analysis

The core of our feedback control loop is the controller. First, we mathematically model the system through the process of *system identification*. Based on the system model, the controller is then designed systematically using feedback control theory. Finally, the control performance of the model is analyzed and the impact of variation between the model and real systems is discussed.

We first introduce the following notation:

T: The control period

$p(k)$: The power consumption of the server in the k^{th} control period. Its Z-transform is $P(z)$.

P_s: The power set point of the server, namely, the desired power constraint.

$t(k)$: The performance state of the processors in the k^{th} control period. Its Z-transform is $T(z)$.

$d(k)$: The difference between $t(k+1)$ and $t(k)$. Specifically $d(k)= t(k+1)-t(k)$. Its Z-transform is $D(z)$.

The goal of the controller is to guarantee that $p(k)$ converges to P_s within a given *settling time*.

4.1. System Modeling

In order to have an effective controller design, it is crucial to model the dynamics of the controlled system, namely the relationship between the control input (i.e. performance state of the processors) and the control output (i.e. power consumption of the server). Therefore, we use a standard approach to this problem, called *system identification* [12]. We infer the relationship between the control input and control output by collecting data on a controlled system and establish a statistical model based on the measured data.

In order to ascertain the response of system power to a change in performance state, we run each of the workloads at every performance state and record the maximum power attained over a 1 second interval. Our workloads, described in Section 6.1, are P4MAX, LINPACK, SPECjbb2005, and SPEC CPU2000 (1 and 2 threads). In traditional system identification, the input is varied to find the relationship between the sequence of past input values and the output response. This is unnecessary here as we have observed the power consumption changes immediately (within a millisecond) as the performance state changes without regard to the previous performance state. That means the power consumption of the server for a given workload is determined exclusively by the performance setting and is independent of the power consumption in previous control periods. Although temperature also affects system-level power, it operates on a much slower timescale and can be modeled as a disturbance input to the controller. Figure 3 plots the relationship between the processor performance setting and the maximum 1 second power consumption. A linear model fits well ($R^2 > 99\%$)

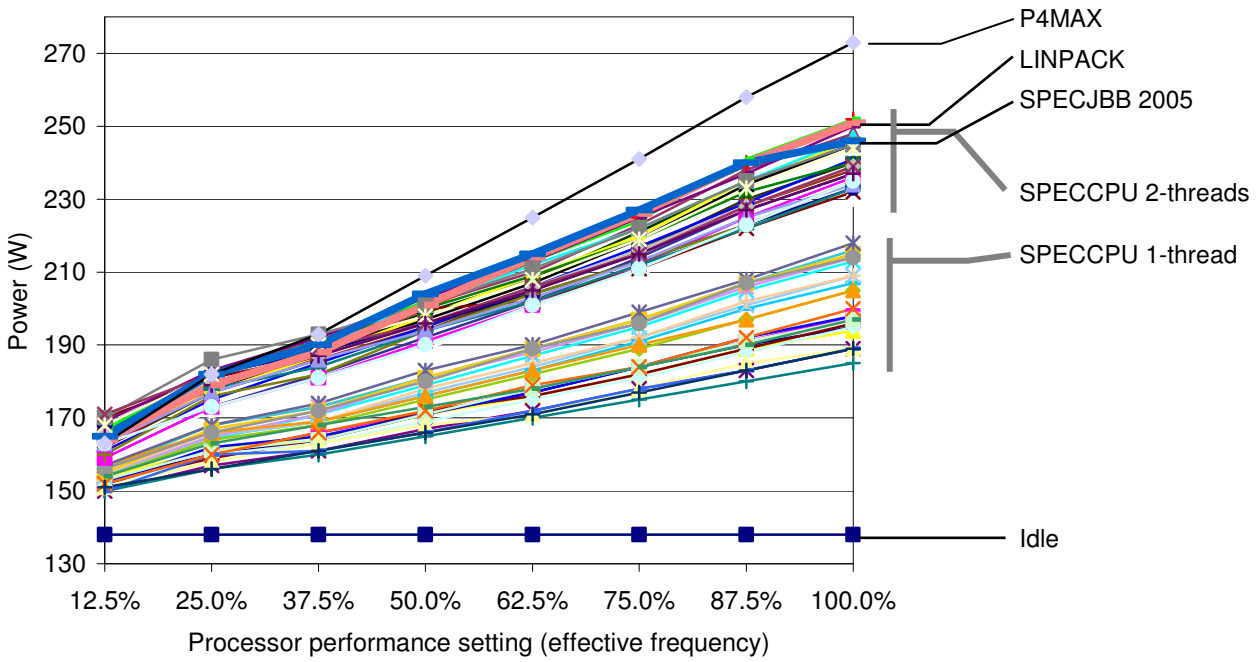


Figure 3: Maximum system-level power for each processor performance state.

for all workloads. Hence, our system model of power consumption is:

$$p(k) = At(k) + B \tag{1}$$

The dynamic model of the system as a difference equation is:

$$p(k+1) = p(k) + Ad(k) \tag{2}$$

4.2. Controller Design

We now apply feedback control theory [12] to design the controller based on the dynamic system model (Equation 2). The goal of the controller design is to meet the following requirements:

Stability: The power should finally settle into a bounded range in response to a bounded reference input.

Zero steady state error: The power should settle to the set point which is the power constraint.

Short settling time: The system should settle to the set point by a specified deadline.

Following standard control theory, we design a *proportional* (P) controller [12], which has a Z-transform of:

$$C(z) = \frac{1}{A} \tag{3}$$

It is easy to prove that the controller is stable and has zero steady state error. Satisfying these requirements means that when the power level or set point is changed, the controller will converge precisely to the desired set point. In addition, the settling time is one control period. Due to space limitations, we skip the detailed derivation which can be found in standard control textbooks [12].

The time-domain form of our proportional controller (Equation 3) is:

$$d(k) = \frac{1}{A}(P_s - p(k)) \tag{4}$$

Based on the definition of $d(k)$, the desired performance setting at the $(k+1)^{th}$ control period is:

$$t(k+1) = t(k) + d(k) \tag{5}$$

The pseudo code of the controller is given in Figure 5.

4.3. Control Performance Analysis

Our controller is designed to achieve the control performance specified in Section 4.2 when the system model is accurate. However, the real system model is usually different from the nominal model (Equation 1) we used to design the controller. This variation could have several causes. For example, the server may have different components and configurations from the modeled system, the workload could be different from the ones used in system identification, or manufacturing differences in the microprocessors may cause them to have different power levels. Since developing a different controller for every server and every workload is

infeasible, it is very important to analyze the impact of model variation to control performance, before we deliver any theoretical guarantees.

An important observation from our measurements is that the workloads always exhibit a linear relationship between power consumption and the performance state, even running on different servers. Based on this observation, we mathematically analyze the impact of model variation on control performance. Without loss of generality, we model the real system as

$$p(k) = g_1 A t(k) + g_2 B \tag{6}$$

where $g_1 = A'/A$ and $g_2 = B'/B$ are *system gains* and are used to model the variation between the real system model (Equation 6) and the nominal model (Equation 1). Since our controller is designed based on the difference equation (Equation 2) of the system model, g_2 has no effect on the performance of the controller. The closed-loop transfer function for the real system is

$$\frac{P(z)}{P_s z/(z-1)} = \frac{g_1}{z-(1-g_1)} \tag{7}$$

Now we investigate each control performance metric.

1. Stability

The closed-loop system pole in Equation 7 is $1- g_1$. In order for the system to be stable (i.e. converge to the desired set point), the pole must be within unit circle [12], namely $|1- g_1| < 1$. Hence the system will remain stable as long as $0 < g_1 < 2$. This result means that if the slope (i.e. A' or $g_1 A$ in Equation 6) of the real model is less than twice that of the nominal model, the system is still stable. The stability range serves as an important reference when applying our controller to different systems and running different workloads.

2. Steady state error

The steady state error of the real system can derived as

$$\lim_{z\to 1}(z-1)P(z) = \lim_{z\to 1}\left(\frac{g_1 z}{z-(1-g_1)} P_s\right) = P_s \tag{8}$$

Equation 8 means that as the system proceeds, the power will settle to P_s which is exactly the set point. Hence, as long as the system is stable (i.e. $0<g_1<2$), we can achieve the desired power value.

3. Settling time

By transforming the closed-loop response (Equation 7) to the time-domain, the power variation model becomes

$$p(k+1) = (1-g_1)\,p(k) + g_1 P_s \tag{9}$$

As commonly defined in control theory, the system settles when $p(k)$ converges into the 2% range around the desired set point P_s. Hence, the required number of sampling periods, k, for the system to settle can be calculated as:

$$k \geq \frac{\ln 0.02}{\ln|1-g_1|} \tag{10}$$

Based on our required settling time of 1 second from Section 3.1, we can use Equation 10 to derive a range of g_1. As long as g_1 is within this range, the system is guaranteed to achieve the required settling time.

4.4. Controller parameters

For this paper, we construct a control loop that can be used for our particular blade at nominal temperatures. Constructing a control loop for an actual product is similar, but involves taking measurements from many blades to account for manufacturing variation and taking the measurements of the BladeCenter under thermal stress to account for different machine room environments, which is beyond the scope of this paper.

The lower bound to which power can be controlled is constrained by the most power consuming benchmark, running at the lowest performance state. For our blade, the maximum power consumed by any workload at the 12.5% performance state was 170 W. This means that the power budget for the control loop cannot be set below 170 W, without risking a violation of the power constraint.

The value for A is chosen by considering the range for A' as shown in Figure 4. The maximum value for A' is the slope of P4MAX, from Figure 3, which is 125.7, the maximum of all workloads. The minimum value for A' depends upon the minimum set point value discussed above. We can estimate a safe lower bound for A' within which the control loop will work. For the minimum A' we take the slope of the imaginary line connecting (138 W, 0%) to (170W, 100%). The reasoning is that if the processor were to slow down to near 0% speed, then the power of the workload would be near that of the idle power of 138 W. Therefore, a workload that can go beyond 170W must have a slope greater than 32. Workloads that have slopes less than 32 cannot reach 170 W and therefore, always run at full-performance as the control loop selects the 100% performance state in an attempt to raise the power to the set point. We calculate A as the average between the minimum and maximum slopes to guarantee stability even in extreme cases. Therefore, A is 78.85 and $0.406 < g_1 < 1.594$.

Our goal is for system power to settle within 1 second to the set point power. If we use the conventional 2% target for the set point in Equation 10, then power could still be several Watts away from the set point, given that the maximum power measured by P4MAX is 273 W. Therefore, we modify Equation 10 to consider how many intervals are required for the power to settle within 0.5 W:

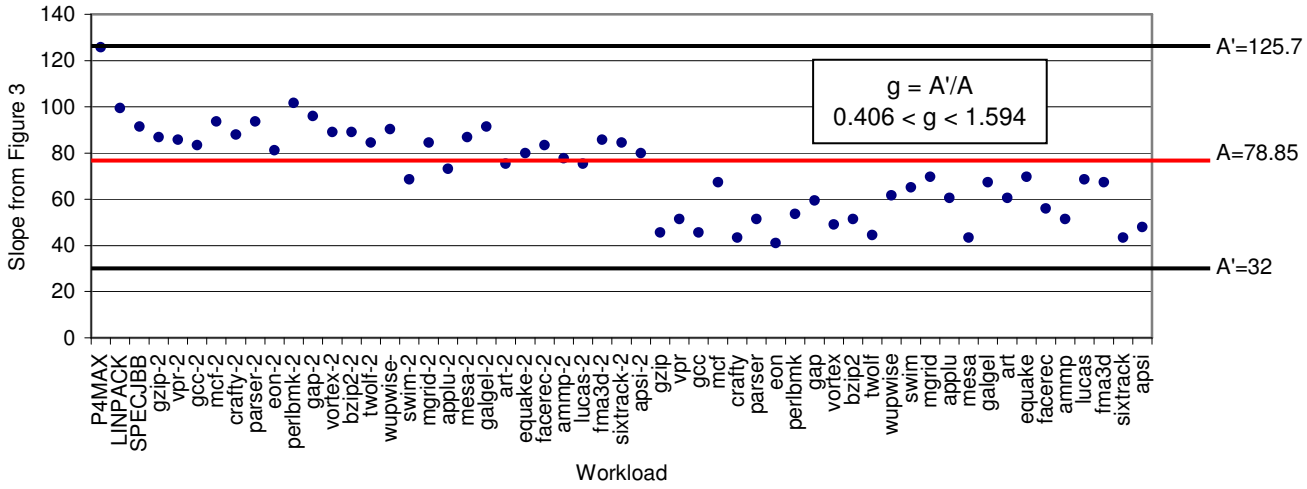


Figure 4: Selection of A and g1. "2" denotes 2 threads. Y-axis is the slope of the lines from Figure 3.

$$k \geq \frac{\ln \frac{0.5}{273}}{\ln|1 - g_1|} \tag{11}$$

Equation 11 uses 0.5 W out of 273 W to calculate the minimum percentage of the set point to which we need to converge. We calculate that k is at least 12.1 which means the power will settle in 13 periods. Dividing 1 second by 13 periods tells us the control period should be less than 76.9 ms. For this set of experiments, we use an interval of 64ms.

5. System Implementation

This section describes each component in our feedback control loop.

5.1. Power monitor

The HS20 blade, contains a signal conditioning circuit attached to sense resistors that allows for monitoring of the current and voltage of the power supply's 12V interface to the blade. This conditioning circuit attaches to analog-to-digital converters on the service processor. The service processor (a 29 MHz Renesas H8S) converts the signals into a calibrated power measurement that represents the power consumed by the entire blade as a function of time.

The absolute measurement is accurate to within 2% due to a calibration feature realized between the hardware and service processor firmware and due to the 1% accuracy rating of the sense resistors. The calibration step reduces a number of additional circuit thermal, aging, and precision issues that would otherwise have led to measurements that varied by 5% or worse as temperatures changed inside the chassis and as a blade's components aged over time. It is fundamental to the entire server system to build its power measurement and management around precision dynamic measurements. The quality of the power measurement is constrained by the cost the measurement circuit. For a high-volume, low-cost server, we use a low-cost circuit that meets a 2% maximum error goal and a 1 Watt digital resolution representation of the discrete power signal.

5.2. Controller

We sample the system power at a high frequency and average it over the control period to provide power values for 64 ms intervals to the P controller. In control theory, this is equivalent to having a low pass filter in front of the controller, which effectively reduces the random noise of the system. Shorter control periods could be used to cover a wider range of system gain at the cost of actuating more often.

5.3. Actuator

The pseudo code including the controller and the saturation handling is shown in Figure 5. The service processor affects the actuation by activating a BIOS routine on the host processor which sets the IA32_CLOCK_MODULATION register in the Xeon processor to set the performance state. In the worst case, the controller may actuate every control period. In our system it takes less than 1 ms to change the performance state. Therefore, the effect of actuation overhead on system performance is no more than 1.5% (1ms/64ms). In situations where the performance state does not change (e.g. system requires less than the power constraint), there is no actuation overhead.

5.4. Power budget

In Section 4.4, we derived the minimum value for the controller set point to be 170 W. Considering we have a 2% maximum error in power measurement, we must

```
//controller code
error = Ps – current power output;
ideal_throttle = throttle + (1/A) * error;
throttle = truncate(ideal_throttle);

//actuator saturation handling
 if (throttle > MAX_throttle)
   throttle = MAX_throttle;
 if (throttle < MIN_throttle)
   throttle = MIN_throttle;
```

Figure 5: Pseudo code of the control loop.

subtract 2% from the desired power budget to form the set point used in the controller. For example, if the desired power budget is 250 W, then we use 245 W as the controller set point to ensure that the real power is below the budget even with the worst case measurement error. Accounting for the worst-case measurement error means the lowest power budget we can guarantee is 174 W. When the server power consumption is below the set point the controller saturates at the highest performance state which allows the system to operate at full performance. Selection of the highest performance state is desired because we want the system to run at full performance in normal situations.

6. Results

In this section, we present the experimental results of using closed-loop control of power on an IBM BladeCenter blade. We first describe the experimental environment and benchmarks used in our experiments. Then we introduce the baseline open-loop controller to compare with our P controller. Finally we present results evaluating common benchmarks under fixed power budgets.

6.1. Experimental Environment

Our test environment is an IBM BladeCenter HS20 blade which was introduced in Section 3.1. This server is fully populated with two 3.6GHz Intel Xeon Irwindale SMP processors with hyper-threading, 8GB memory, two 36 GB SCSI hard-disks, dual 1 Gb Ethernet interfaces, and a Fibre Channel daughter card.

We evaluate each power management policy under two budgets. The first budget is 250 W, which corresponds to the case in which the BladeCenter has lost a single redundant 2000 W power supply. The second budget is 230 W, which is similar to the loss of a 1875 W power supply, which was used by the previous generation of blade servers. Each measurement presented is the average value of three runs.

Our evaluation workloads are listed in Table 1. Some of the workloads are run under SUSE Linux Enterprise Server 9 SP 2 and others are run under Windows Server 2003 Enterprise x64 Edition. In our evaluation, we do not show results for single thread SPEC CPU2000 because the power consumption is typically below the power budgets we evaluate and would result in no application slowdown. The P4MAX workload is a program designed to produce the maximum power consumption on the Intel Xeon microprocessors [13].

Workload	Environment	Notes
P4MAX	Windows	Run for 1 minute on both processors using 100% setting (4 threads total).
SPEC CPU2000	Linux	Compiled with Intel Compiler 9.0 (32-bit). Performance results are only shown for rate (2 users) mode.
SPECjbb2005	Windows	JVM is BEA JRockit JRE 5.0 Update 3 (RR25.2.0-28). Run 4 warehouses only.
Intel Optimized LINPACK	Linux	Version 2.1.2. Run with two threads. 15000x15000 matrix.

Table 1: Workloads.

6.2. Baseline

Our baseline is an open-loop policy, referred to as *open-loop* which selects a fixed performance setting for a given power budget. It is designed to be suitable for typical system conditions and is derived from our power measurements of P4MAX. The baseline assumes that under the loss of a power supply, the system could be running any workload and therefore must lower the performance setting to the point that even the most power-consuming workload could be run. For the 230 W budget, *open-loop* uses the 62.5% performance state and at 250 W budget it uses the 75% performance state. The BladeCenter product currently uses an open-loop policy, but instead estimates the proper performance setting based on blades' run-time component configurations [14]. Since we limit our evaluation of the closed-loop controller to nominal conditions, the BladeCenter open-loop control is not suitable for comparison in this study.

It is interesting to consider how much more our *open-loop* policy could be improved. As a thought experiment, we also consider the *best-open* policy which is an open loop that knows a priori the highest performance state for each workload that does not violate the power budget. This policy is impossible to implement in practice because it

requires foreknowledge of the workload to be run, its power demand on the processors, and the thermal environment of the facility. We consider this policy to represent an upper bound for any open-loop controller that could be developed and show that our closed-loop controller can still make impressive performance gains over it.

6.3. Experiment I: Constant power-level workload

First we examine the ability of the controllers to keep a workload with a steady power load at a fixed power budget. We run the P4MAX workload on each controller at each power budget for one minute and show the results in Table 2.

Controller	Statistic	230 W budget	250 W budget
Open-loop or **Best-open**	fixed speed	62.5%	75.0%
	avg. power	224.5 W	240.1 W
	max power 1s	225 W	241 W
	max power 8s	225 W	240 W
P controller set point power		225 W	245 W
P controller	avg. speed	64.7%	81.2%
	avg. power	225.4 W	245.5 W
	max power 1s	226 W	246 W
	max power 8s	225 W	245 W

Table 2: Results for P4MAX.

Since the *open-loop* policy is based on P4MAX, it allows the processors to run at the same speed as the *best-open* policy. A 230 W power budget causes the processors to be statically set to a performance state of 62.5%. A 250 W power budget causes the processors to be statically set to a performance state of 75%. For the 250 W budget, the open-loop maximum power over a 1 second interval is several Watts lower than the power budget, even when accounting for measurement error. This illustrates the wasted opportunity to further increase the processor speed without violating the power constraint.

The P controller keeps the workload's average power and the maximum power over an 8 second interval at precisely the controller set point. The maximum power over a 1 second interval is 1 W above the set point of 225 W because the power occasionally goes over the set point before the controller takes action and corrects it. For all workloads we studied, the maximum power in a 1 second interval never exceeds the set point by 2 W. These small, short excursions above the set point are negligible to the BladeCenter power supply, which can handle much higher overload conditions for longer periods of time [15]. These excursions could be reduced by either shortening the control period at the cost of more actuation overhead, or lowering the set point. At the 230 W budget, the P controller has a small performance advantage over the open-loop controllers. At the 250 W budget, which corresponds to a single power supply failure, the P controller runs the processors at 8% faster than the *open-loop* or *best-open* policy. Therefore, the P controller can make performance improvements over even the best open-loop controllers.

Accurate power measurement is a vital part of our closed-loop controller and is required to achieve good performance. For instance, if the measurement error was 10% instead of 2%, then the P controller would need to use a 225 W set point for the 250 W budget case. This is the same set point we used for the 230 W budget case with a 2% measurement error. Therefore, a 10% measurement error would cause the processors for the 250 W budget case to run at 64.7% speed instead of 81.2% speed. Additionally, it is clear that in some cases an open-loop controller can achieve a higher performance than a close-loop controller when its measurements are more accurate. For this reason, we assume the same measurement accuracy of 2% for all controllers to make fair comparisons.

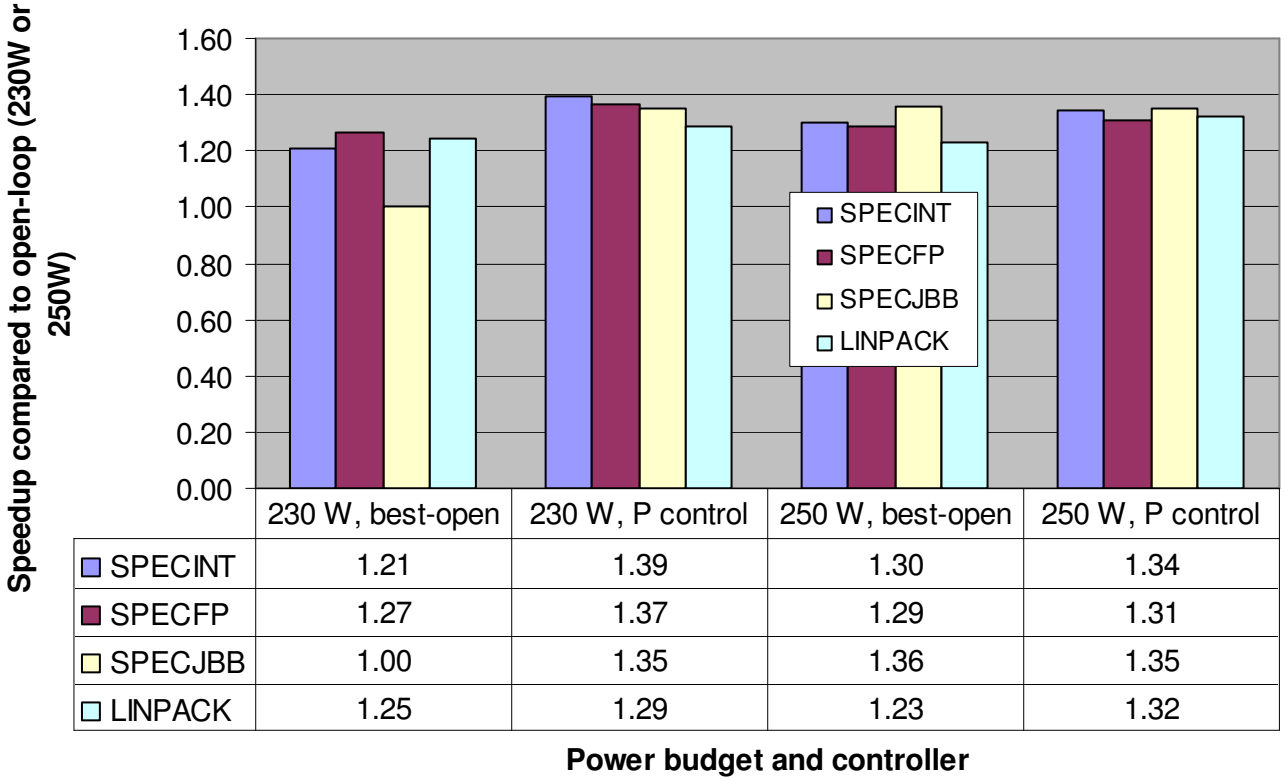
6.4. Experiment II: Application Performance

In this section, we investigate the impact of closed-loop power control on the performance on common microprocessor benchmarks. We ran the *open-loop*, *best-open*, and P controller under each power budget and recorded the throughput achieved. In Table 3, we present the benchmark performance as a fraction of the throughput at full-performance. For example, a measure of 1.0 means the application ran at the same rate as it would in the 100% performance state. A measure of 0.5 means that the workload achieved half of the throughput as it would the 100% performance state. The throughput for LINPACK is measured in GFLOP/S, the throughput for SPECjbb2005 is measured in business operations per second, and SPEC CPU2000 is run with 2 user threads and recorded as number of runs per second.

Workload	Power budget and control policy									
	230 W budget					250 W budget				
	Open-loop	Best-open		P controller		Open-loop	Best-open		P controller	
	Perf.	Fixed speed	Perf.	Avg. Speed	Perf.	Perf.	Fixed speed	Perf.	Avg. Speed	Perf.
164.gzip	0.60	75.0%	0.74	80.3%	0.83	0.74	100.0%	1.00	99.2%	1.00
175.vpr	0.62	75.0%	0.74	88.9%	0.88	0.74	100.0%	1.00	100.0%	1.00
176.gcc	0.61	75.0%	0.74	89.6%	0.83	0.74	100.0%	1.00	100.0%	1.00
181.mcf	0.64	62.5%	0.64	88.8%	0.89	0.76	87.5%	0.89	99.0%	1.00
186.crafty	0.61	75.0%	0.73	93.6%	0.88	0.73	100.0%	1.00	100.0%	1.00
197.parser	0.62	75.0%	0.74	80.7%	0.82	0.74	100.0%	1.00	100.0%	1.00
252.eon	0.61	75.0%	0.73	95.4%	0.95	0.73	100.0%	1.00	100.0%	1.00
253.perlbmk	0.61	75.0%	0.74	81.1%	0.77	0.74	87.5%	0.88	97.3%	0.97
254.gap	0.63	75.0%	0.75	78.7%	0.77	0.75	87.5%	0.89	94.7%	0.95
255.vortex	0.61	75.0%	0.74	90.3%	0.89	0.74	100.0%	1.00	100.0%	1.00
256.bzip2	0.58	75.0%	0.75	79.8%	0.83	0.75	87.5%	0.89	99.6%	1.00
300.twolf	0.61	87.5%	0.87	91.5%	0.91	0.73	100.0%	1.00	100.0%	1.00
168.wupwise	0.65	75.0%	0.76	79.9%	0.80	0.76	87.5%	0.89	98.6%	0.99
171.swim	0.70	87.5%	0.91	92.3%	0.96	0.79	100.0%	1.00	100.0%	1.00
172.mgrid	0.68	75.0%	0.78	85.5%	0.86	0.78	100.0%	1.00	99.9%	1.00
173.applu	0.67	87.5%	0.90	90.9%	0.93	0.77	100.0%	1.00	100.0%	1.00
177.mesa	0.61	87.5%	0.88	92.4%	0.92	0.74	100.0%	1.00	100.0%	1.00
178.galgel	0.62	75.0%	0.73	83.5%	0.78	0.73	87.5%	0.83	96.9%	0.96
179.art	0.66	87.5%	0.88	94.1%	0.94	0.76	100.0%	1.00	100.0%	0.99
183.equake	0.67	75.0%	0.78	85.8%	0.88	0.78	100.0%	1.00	100.0%	1.00
187.facerec	0.69	75.0%	0.79	85.6%	0.87	0.79	100.0%	1.00	97.3%	0.98
188.ammp	0.61	87.5%	0.88	92.9%	0.97	0.73	100.0%	1.00	100.0%	0.99
189.lucas	0.67	75.0%	0.78	90.0%	0.90	0.78	100.0%	1.00	100.0%	1.00
191.fma3d	0.66	75.0%	0.77	85.6%	0.88	0.77	100.0%	1.00	99.1%	1.00
200.sixtrack	0.61	87.5%	0.88	95.3%	0.91	0.74	100.0%	1.00	100.0%	1.00
301.apsi	0.62	87.5%	0.88	86.7%	0.87	0.74	100.0%	1.00	99.0%	1.00
SPEC CPU2000 INT (rate 2) avg.	**0.61**	**75.0%**	**0.74**	**86.8%**	**0.85**	**0.74**	**95.8%**	**0.96**	**98.9%**	**0.99**
SPEC CPU2000 FP (rate 2) avg.	**0.65**	**81.2%**	**0.83**	**88.6%**	**0.89**	**0.76**	**98.2%**	**0.98**	**99.3%**	**0.99**
SPECJBB	**0.60**	**62.5%**	**0.60**	**81.8%**	**0.81**	**0.73**	**100.0%**	**1.00**	**98.7%**	**0.99**
LINPACK	**0.56**	**75.0%**	**0.70**	**78.8%**	**0.72**	**0.70**	**87.5%**	**0.86**	**95.8%**	**0.92**

Table 3: Application performance as a fraction of full-performance throughput. *Perf* columns show the fraction of full-performance throughput of the workload (1=full performance). *Fixed speed* indicates the processor speed used by the best-open controller. *Avg speed* indicates the effective speed of the processors as the P controller changes performance state over the workload. *Open-loop* always runs at 62.5% speed at 230 W and 75% speed at 250 W.

For the 250 W budget, *open-loop* achieved 70%-79% of full performance throughput, while *best-open* achieved 83%-100% and P controller achieved 92%-100%. For the 230 W budget, *open-loop* ran at 56%-70%, *best-open* ran at 60%-91% and P controller ran at 72%-97% of full throughput. For some runs, the best-open policy outperforms the P controller by 1% or 2%. We believe this is due to the 1.5% actuation overhead. At the 250 W budget, the P controller ran most benchmarks near full performance. Not surprisingly, LINPACK, which is one of the hottest workloads, experiences the most slowdown under all controllers. Figure 6 shows the P controller has large speedups from 1.2 to 1.4 over the open-loop. At the 250 W budget, the P controller achieves performance similar to *best-open*, which is not surprising, given that few workloads are power-constrained at this level. The difference is more dramatic at the 230 W budget, where the P controller shows 3% to 35% better application performance. Although *best-open*, which uses an ideal fixed performance state, can sometimes approach the application performance of the P controller, it is important to remember that it cannot be implemented in practice.



	230 W, best-open	230 W, P control	250 W, best-open	250 W, P control
SPECINT	1.21	1.39	1.30	1.34
SPECFP	1.27	1.37	1.29	1.31
SPECJBB	1.00	1.35	1.36	1.35
LINPACK	1.25	1.29	1.23	1.32

Figure 6: Speedup of controllers compared to open-loop. 1.0 = open-loop throughput at either 230 W or 250 W budget.

The P controller's ability to use measure power and adapt to the workload as it changes allows it to outperform fixed performance state policies in most situations.

7. Conclusions

In this paper we present a control-theoretic power management system for managing peak power. As an advantage of theoretical design, we can quantitatively analyze system performance and choose the best control parameters, even when the system model has significant variations. Specifically, we determine the change in power consumption caused by workload variation in order to select the controller gain parameter. We use the gain parameter to select a control period that corresponds to the requirements of the overload condition of a power supply. This guarantees that when a power supply fails, under any workload, the power drawn by the remaining supply can be reduced to a nominal level within 1 second.

Feedback control of power is useful for improving server reliability. Specifically, closed-loop control provides less performance loss under a partial power supply failure than by using simpler open-loop solutions found in conventional servers. Since the closed-loop controller measures the actual power the system consumes, it can react to workload changes and adapt the performance state to meet the requested power budget. This results in a 1.2 to 1.4 speedup in application performance. A key factor in realizing this performance improvement is having accurate power measurement which reduces controller design margins and utilizes the available power supply effectively.

Feedback control of power has many implications for the future design and operation of servers. Enforcing a run-time power constraint with closed-loop control, rather than a design-time power constraint with open-loop control, will allow servers to flexibly adapt to their power and thermal environments. System designers could save cost by using power supplies that have just enough capacity to run important applications at full-performance for a target market, but are under-provisioned for worst-case power benchmarks with hardware components that consume worst-case power. New blade servers would be more likely to run at full-performance within the power capacity of the prior generation of blade server chassis. Data center administrators could slightly reduce the power consumption of their servers to match individual rack-level power and cooling requirements, with little impact on application performance. In general, feedback control power allows servers to run at the highest

performance level possible under a given system-level power constraint.

8. References

[1] Charles Lefurgy, Karthick Rajamani, Freeman Rawson, Wes Felter, Mike Kistler and Tom W. Keller, "Energy Management for Commercial Servers", Computer, volume 36, number 12, December, 2004, pages 39-48.

[2] P. Bohrer, M. Elnozahy, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony. The Case for Power Management in Web Servers. In R. Graybill and R. Melhem, editors, Power Aware Computing. Kluwer Academic Publishers, 2002.

[3] H. Zeng, X. Fan, C. Ellis, A. Lebeck, and A. Vahdat. Ecosystem: Managing energy as a first class operating system resource. In International Conference on Architectural Support for Programming Languages and Operating Systems, 2002.

[4] Y. H. Lu, L. Benini, and G. De Micheli. Operating-system directed power reduction. In International Symposium on Low Power Electronics and Design (ISLPED), 2000, pages 37–42.

[5] J. Hellerstein, Y. Diao, S. Parech, and D. Tilbury, Feedback Control of Computing Systems, John Wiley & Sons, 2004.

[6] K. Skadron, T. Abdelzaher, and M. Stan. "Control-Theoretic Techniques and Thermal-RC Modeling for Accurate and Localized Dynamic Thermal Management." In Proceedings of the Eighth International Symposium on High-Performance Computer Architecture, Feb. 2002, pages 17-28.

[7] Q. Wu, P. Juang, M. Martonosi, L. Peh, and D. W. Clark. Formal control techniques for power-performance management. IEEE Micro, 25(5):52-62, 2005.

[8] R. J. Minerick, V. W. Freeh, and P. M. Kogge, "Dynamic Power Management Using Feedback", In Proceedings of *Workshop on Compilers and Operating Systems for Low Power* (COLP), September, 2002, pages 6-1—6-10.

[9] M. E. Femal and V. W. Freeh, "Boosting Data Center Performance Through Non-Uniform Power Allocation", In Proceedings of *Second International Conference on Autonomic Computing* (ICAC), June, 2005, pp 250-262.

[10] V. Sharma, A. Thomas, T. Abdelzaher, Z. Lu, and K. Skadron. Power-Aware QoS Management on Web Servers. the 24th International Real-Time Systems Symposium (RTSS), Dec. 2003.

[11] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, N. Gautam. Managing Server Energy and Operational Costs in Hosting Centers. In Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, June 2005.

[12] G. F. Franklin, J. D. Powell, and M. Workman. Digital Control of Dynamic Systems, 3rd edition. Addition-Wesley, 1997.

[13] Intel, *Maximum Power Program User Guide Version 2.0 for Nocona/Irwindale Processor*, 2004.

[14] T. Brey et al., "BladeCenter Chassis Management", *IBM J. Res. & Dev.*, vol. 49, no. 6, November, 2005.

[15] J. Hughes et al., "BladeCenter Processor Blades, I/O Expansion Adapters, and Units", *IBM J. Res. & Dev.*, vol. 49, no. 6, November, 2005.