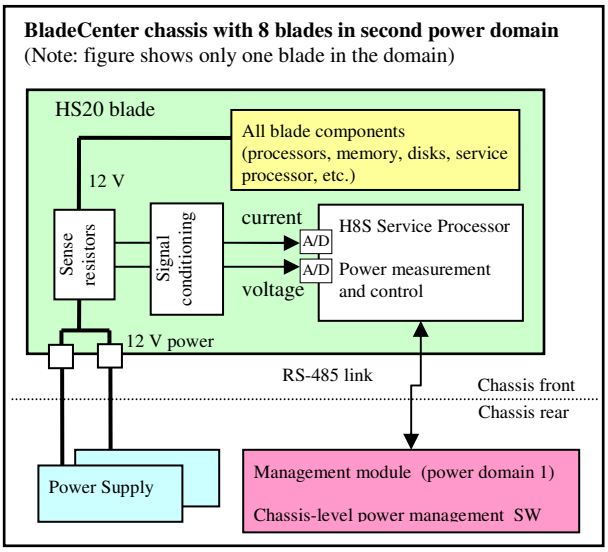

Figure 1: BladeCenter chassis.

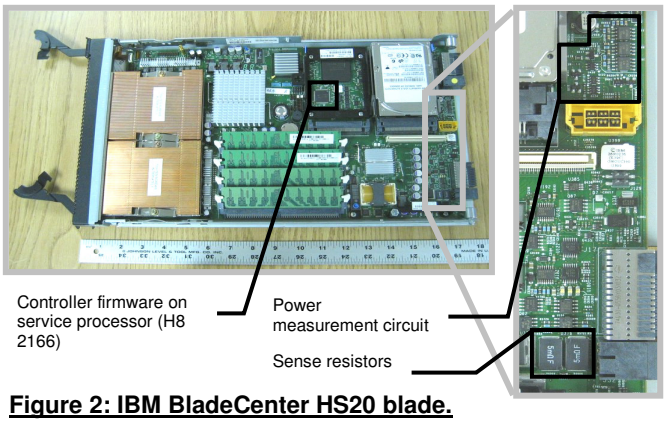

Figure 2: IBM BladeCenter HS20 blade.

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. We show that under a heavy power constraint, our controller can provide much better performance than simpler open-loop and ad-hoc techniques. Under light power constraints, our controller often runs workloads at full speed.

6. Our controller allows server designers to safely underprovision the power supply to lower costs while negligibly affecting performance of real-world workloads.

In the next section we 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 the controller based on feedback control theory in Section 4. Next, in Section 5, we analyze the control performance and show how to account for system variation. In Section 6, we describe the detailed implementation of each component in the feedback control loop. Our empirical results are presented in Section 7 and we draw conclusions in Section 8.

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. We use processor throttling as the actuator in our power controller.

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 servers [20] which measures power consumed by the entire server. 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 and implement it on a conventional server. For example, we are able to handle unexpected power demand from memory, disk, and I/O components.

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 with tighter margins.

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.

Brooks et al. [17] use ad-hoc control to limit processor temperature so cheaper heat-sinks can be used. In a similar way, one result of our work is that system designers are no longer required to use over-provisioned power supplies to survive worst-case scenarios.

Felter et al. [18] use open-loop control to shift power between processor and memory components to maintain a server power budget. In contrast, our solution can operate at smaller design margins because it uses precision measurement. Our controller could be added to such a system to provide tight guarantees on the system-level limit and provide a safe environment for power shifting between components that do not use measurement.

Foxton [19], which is not yet available in products, uses on-chip power measurement to control power in a single Itanium processor. Our technique is used outside the main application processor and is therefore applicable to a wider range of architectures. Our power measurement circuit has already been deployed across multiple server products spanning three processor architectures [20].

Ranganathan et al. propose using processor performance states to control blade server power [23]. However, they rely on ad-hoc control methods that do not guarantee stability across a variety of workloads.

This paper builds on a previously designed control system described in a technical report [15]. The previous controller had a power measurement precision of 1 W, while the controller presented here has a precision of 0.1 W. This not only improves the measurement quality, but raises the performance results slightly. We add a discussion of the modulation technique used by the system. We also add a comparison to a typical ad-hoc approach to control power and show why such a technique is generally not desirable for server power management.

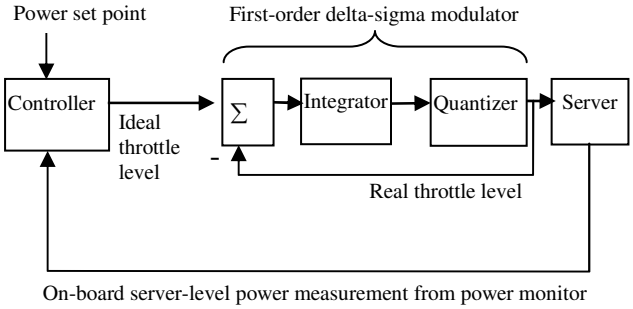

Figure 3: System diagram for power control

3. System-level power management

We present a description of the current power management solution in the BladeCenter as an example of requirements in conventional servers that the control loop must meet in order to satisfy power supply constraints.

3.1. BladeCenter test platform

Our test platform is a single 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 blade servers 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 blade servers. Our discussion and experiments focus on the second power domain because its blades have a stricter, lower power constraint.

BladeCenter adheres to a policy which specifies that the power supplies must not be in an overload situation (drawing more power than their rating) for more than 1 second [14]. Overload can happen when one of the power supplies fails and the load is shifted completely to the remaining supply. 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 practice, the one second target is conservative and the power supply can sustain a power overload for even longer periods of time. In this work, we design the controller to manage power at this one second time scale. Servers with different overload power constraints may have different requirements.

Our blade has a label power of 308 W. During overload conditions, the power must be reduced to 250 W. The mechanism to throttle blade power 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. 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. Developing additional power controller for non-processor components would further extend the power control range for the server. Femal et al. and Ranganathan et al. have adopted the approach of using only processor performance states to limit whole-server power consumption [9][23].

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 6. 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

A detailed description of the feedback control loop, including its derivation, can be found in our prior technical report [15]. Hence, this section will only reiterate the important features of the controller. Figure 3 shows the system diagram.

We first introduce the following notation:

T: The control period.

$p(k)$: The power consumption of the server in the k^{th} control period.

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.

$d(k)$: The difference between $t(k+1)$ and $t(k)$. Specifically $d(k)= t(k+1)-t(k)$.

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

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.

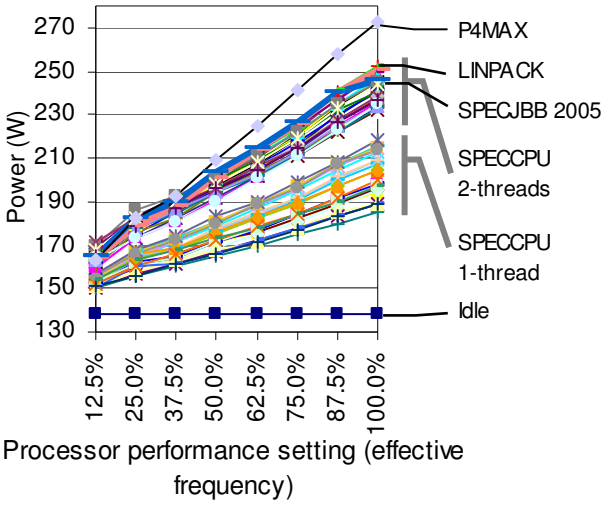

Figure 4: Maximum system-level power measurement for each processor performance state.

4.1. System Modeling

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 4 plots the relationship between the processor performance setting and the maximum 1 second power consumption. A linear model fits well ($R^2 > 99\%$) for all workloads. Hence, our system model of power consumption is:

$$p(k) = At(k) + B \quad (1)$$

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

$$p(k+1) = p(k) + Ad(k) \quad (2)$$

4.2. Controller Design

Following standard control theory, we design a *proportional* (P) controller [12]. We used a P controller instead of a more sophisticated PI controller because the actuator includes an integration step (as part of the first-order delta-sigma modulator) such that zero steady-state error can be achieved without resorting to a PI controller.

The time-domain form of our P controller is:

$$d(k) = \frac{1}{A}(P_s - p(k)) \quad (3)$$

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. Due to space limitations, we skip the detailed derivation which can be found in standard control textbooks [12].

The desired performance setting in period *k+1* is:

$$t(k+1) = t(k) + d(k) \quad (4)$$

5. Performance Analysis for Model Variation

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

$$p(k) = g_1 A t(k) + g_2 B \quad (5)$$

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 5) 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.

We investigate system stability when a controller designed based on the nominal model (1) is used to control the real system (5). The technical report [15] shows that the system will remain stable as long as $0 < g_1 < 2$. This established stability range is an important guideline for us to choose the control parameter *A*.

The technical report explains the reasoning for our selection of A. In brief, we calculate *A* as the average between the minimum and maximum slopes (Figure 4) to guarantee stability even in extreme cases. In this case, *A* is 78.85. and $0.406 < g_1 < 1.594$. The technical report also shows that the system has zero steady state error even under model variation. Hence, as long as the system is stable (i.e. $0<g_1<2$), we are guaranteed to achieve the desired power value.

```
// Controller code
error = setpoint - power_measurement;
ideal_throttle = throttle + (1/A) * error;

// Actuator code
// First-order delta-sigma modulation
throttle = truncate(ideal_throttle);
frac = ideal_throttle - throttle;
total_fraction = total_fraction + frac;
if (total_fraction > 1) {
   throttle = throttle + 1;
   total_fraction = total_fraction - 1;
}
// Actuator saturation handling
if (throttle > 7) throttle = 7;
if (throttle < 0) throttle = 0;
```

Figure 5: Pseudo code for P controller

In the technical report, we show that the system will settle to within 0.5 W of the power set point in 13 control intervals. Dividing 1 second by 13 periods tells us the control period should be less than 76.9 ms. For the P controller, we use a slightly more conservative interval of 64 ms.

Empirically, we found the minimum set point to be 170 W, which is the maximum power consumed by the workloads at the lowest performance setting. In other words, using a set point less than 170 W risks a violation of the power constraint for some workloads. Therefore, the practical range of the set point is from 170 W to 308 W (label power).

6. System Architecture and Implementation

The power control architecture in our server has three pieces. A *power monitor* (hardware and firmware) and a *controller* (firmware) are two new pieces added to the blade that measure power at 1000 samples per second and decide on a throttle setting for the processors every 64 ms. The *actuator* piece providing performance state selection is already available in processors today. We augment the actuator by modulating between the available performance states to effectively produce a finer range of performance states. The pseudo code used on the service processor for the controller and the actuator components is shown in Figure 5.

6.1. Power monitor

The power monitor measures the blade's power at its 12 V bulk power supply interface. The power supply interface is attached to sense resistors and a signal conditioning circuit to obtain the current and voltage levels. The conditioning circuit attaches to analog-to-digital converters on the service processor. Every millisecond, the power monitor firmware in the service processor (a 29 MHz Renesas H8) converts the current and voltage signals into a calibrated power measurement for the

Workload	OS	Notes
P4MAX	Windows	Run for 3 minutes 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 mode (2 users).
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.

Power budget	Open-loop processor performance setting	Improved Ad-hoc set point	P control set point
250 W	75%	238.9 W	245.0 W
240 W	62.5%	229.1 W	235.2 W
230 W	62.5%	219.3 W	225.4 W
220 W	50%	209.5 W	215.6 W
210 W	37.5%	199.7 W	205.8 W

Table 2: Controller set points used in application performance measurements. The open-loop performance setting is determined by the finding the highest performance setting that runs P4MAX without a violation of the power budget. P controller set point is calculated by reducing the power budget by 2% measurement error. Ad-hoc controller set point is 6.1 W lower than P controller set point to account for safety margin due to steady-state error in the ad-hoc controller.

entire blade. After every 64 readings, the power monitor calculates the average power over the previous 64 ms interval which is sent to the controller.

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 of 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 0.1 Watt digital resolution representation of the discrete power signal.

6.2. Controller

At each 64 ms control interval, the 64 ms average power is used to select the processor throttle level. The output of the controller is an ideal throttle value represented as a floating-point number. The value 0 represents the 12.5% performance state and 7 represents

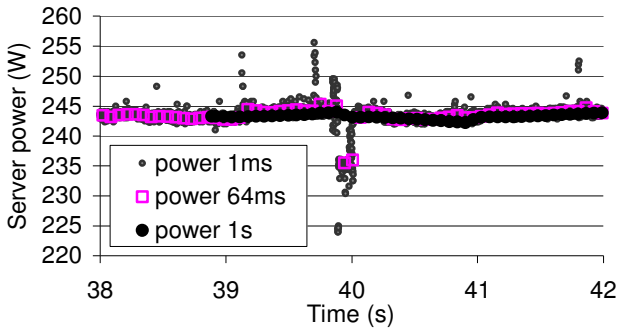

Figure 6: Linpack without power management. Graph shows 4 seconds of the Linpack benchmark after running for 38 seconds. Average power over 1 ms, 64 ms, and 1 s periods is plotted. The plot points mark the end of the measurement period. Linpack has almost constant power consumption with periodic dips in power (as seen around the 40 second mark).

the 100% performance state. It is possible to represent effective performance states that are not strictly available in the processor. For example, 6.2 represents a performance state of 90%. The actuator is responsible for approximating this value.

6.3. Actuator

Since the output of the controller is a floating-point value, the actuator code must resolve this to a series of discrete performance state settings to approximate the value. For example, to approximate 6.2, the modulator would output the sequence 6, 6, 6, 6, 7, 6, 6, 6, 6, 7, etc. To do this, we implement a first-order delta-sigma modulator [16], which is commonly used in analog-to-digital signal conversion.

If the modulator outputs a performance state that was different from the previous control interval, the service processor affects the actuation by activating a BIOS routine on the host processor. This routine sets the IA32_CLOCK_MODULATION register in the Xeon processor to invoke the performance state. All processors in the server are set to the same performance state.

In the worst case, the controller may actuate every control period. In our two processor server, the BIOS takes 40 microseconds to change the performance state. Therefore, the effect of actuation overhead on system performance is no more than 0.07% (40 microseconds/64ms). In situations where the performance state does not change (e.g. server requires less than the power constraint), there is no actuation overhead.

6.4. Power budget

In Section 5, we found the minimum value for the controller set point to be 170 W. Considering we have a 2% maximum error in power measurement, we must subtract the measurement error from the desired power budget to form the set point used in the controller. For example, if the desired power budget is 250.0 W, then we

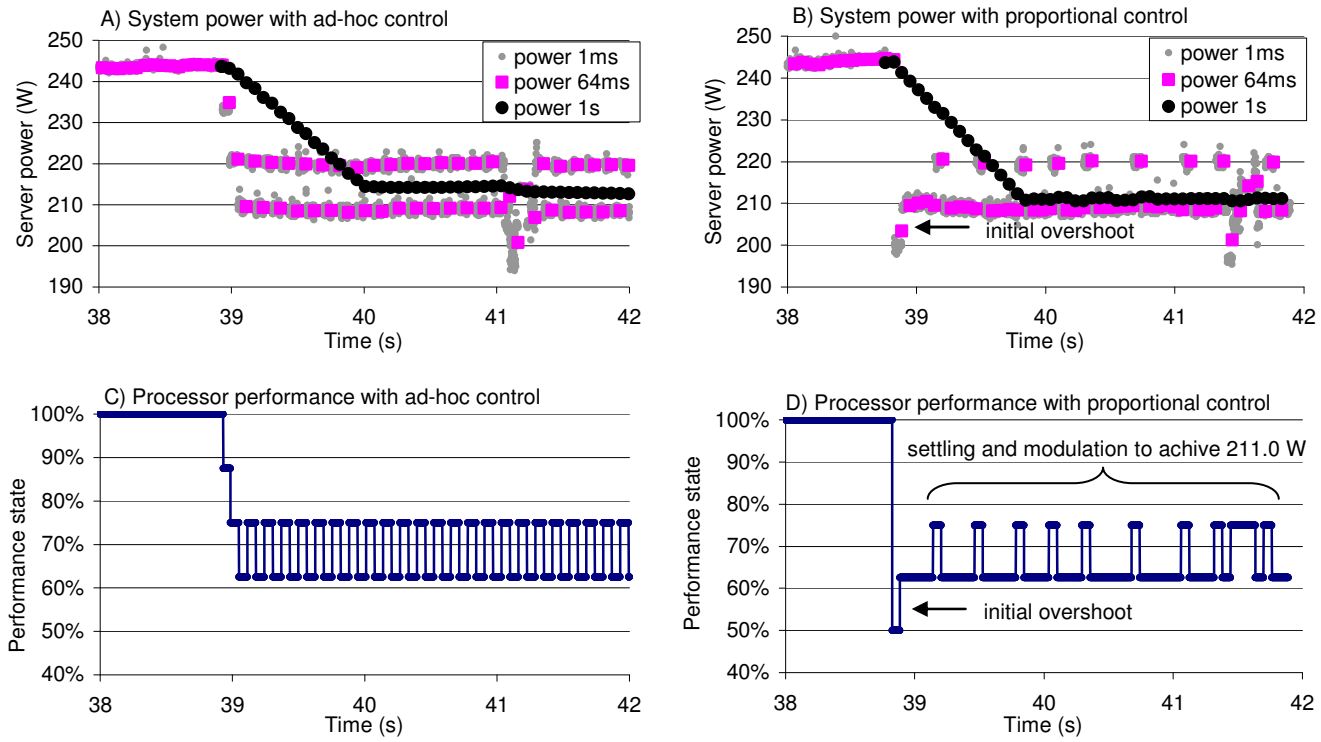

Figure 7: Example of ad-hoc controller and P controller. Linpack benchmark from Figure 6 is shown with the feedback controller turned on at about 39 seconds into the run. The set point for each controller is 211.0 W. In A) and C), the ad-hoc controller moves one performance state up or down depending on whether the 64 ms power is above or below the set point. After 1 second (around t=40s), the average 1 second power is 216.0 W, violating the set point by 5 W. The processor speed averages 68.8%. The power consumption never reaches the set point. In B) and D), the P controller shows more efficient use of the actuator and modulates it to achieve precisely the 211.0 W target. One second after the set point (around t=40s) the average power over 1 second is 210.7 W and is considered to be settled (by design, within 0.5 W of the set point). By t=41, the average power over 1 second measures 211.0 W and the processor speed averages 65.8%.

use 245.0 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 173.4 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.

7. Results

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

7.1. 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 compare our P controller to an open-loop controller and an ad-hoc controller that both represent common solutions found in industry. We evaluate each of these three power management policies using power budgets ranging from 210 W to 250 W. The 250 W budget corresponds to the case in which the BladeCenter has lost a single redundant 2000 W power supply. 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].

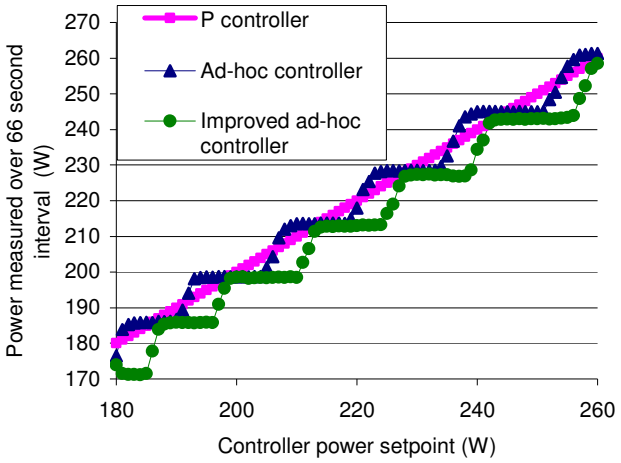

Figure 8: Steady-state error. This figure shows the error the controller experiences in adhering to a given set point value. P4MAX, which exhibits very steady power consumption, was run for set points from 180 W to 260 W in 1 W increments. The maximum power within a 66 second interval was recorded and the average of 3 runs is plotted. The P controller measurements matched the set point to 0.1 W precision (the limits of the measurement circuit). The ad-hoc controller showed long-term steady-state violations in the measured power by up to 6.1 W over the set point. The final series shows the improved ad-hoc controller run with a safety margin by subtracting 6.1 W from the set point before running the experiment.

7.2. Open-loop control

The open-loop controller, referred to as *open-loop,* selects a fixed performance setting for a given power budget. It assumes that the system could be running any workload and therefore must lower the performance state to the point that even the most power-consuming workload could be run. For this study we take the highest power load to be P4MAX running at normal machine room temperatures. The performance state used for a specific power budget is shown in Table 2 which comes from our power measurements of P4MAX in Figure 4.

7.3. Ad-hoc control

We also compare the P controller with an ad-hoc controller that is representative of typical industry solutions to control power and temperature. We use this to motivate our use of control theory and demonstrate that ad-hoc controllers do not generally have the desired properties that make them safe and reliable. Our ad-hoc controller actuates every 64 ms just like the P controller. However, it simply raises or lowers the performance state by one step depending on whether the measured power is lower or higher than the power set point.

A simple example of the P controller and ad-hoc controller shows how they are different from each other. First, consider the LINPACK benchmark shown in Figure 6 which runs at up to 245 W with no power management. In Figure 7, a power constraint is introduced by setting the power set point to 211.0 W at t=39 s.

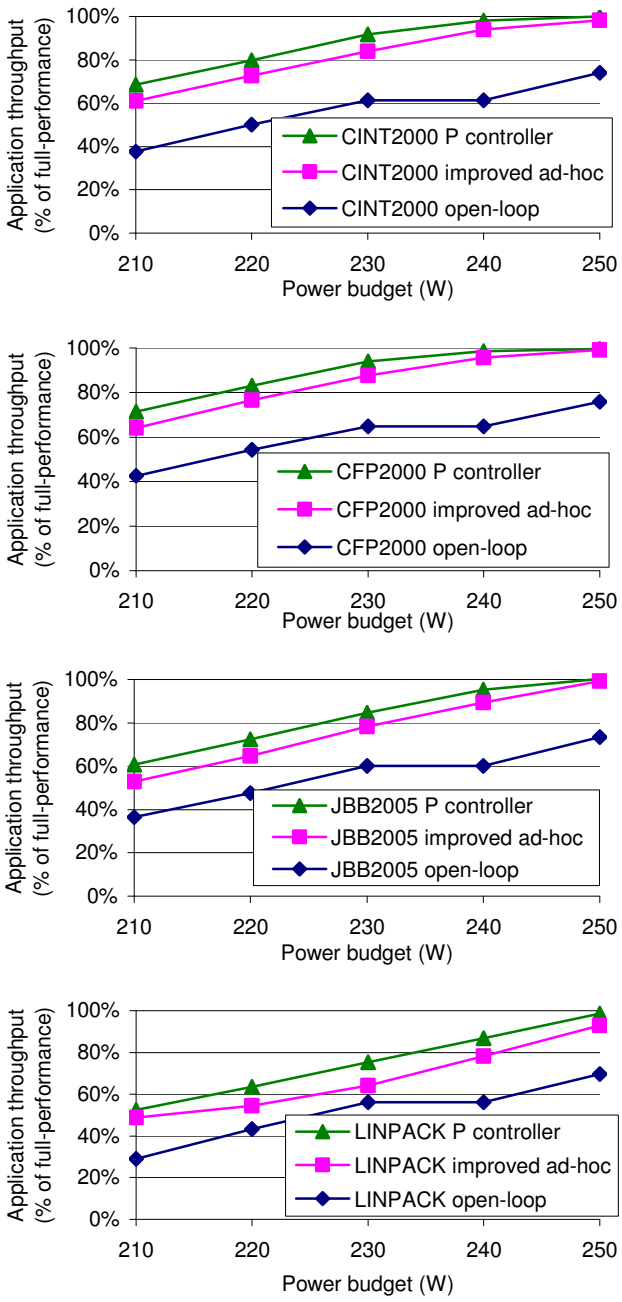

Figure 9: Application performance. The Y-axis shows throughput performance compared to the application running at full-performance (no power management).

The ad-hoc control responds by stepping down the performance state of the processors until the power is lower than the set point. Afterwards, the controller oscillates between the 62.5% and 75% performance states because the set point power is between the power consumption levels at these performance states for

LINPACK. The power consumption never settles to the set point and has a steady-state error of 5 W. Even if the ad-hoc controller used a shorter control period, it would still oscillate and have a steady-state error.

The P controller initially responds by lowering the performance state by several steps in the first control interval. One important benefit of proportional control is that it can react quicker than the ad-hoc method. It initially overshoots the set point, but then settles within 1 second as designed to the set point power. The first-order delta-sigma modulator in the P controller modulates performance states to run the processors at an effective frequency of 65.8% to meet this set point.

One is tempted to think that the delta-sigma modulator could easily be added to the ad-hoc controller to improve its steady-state error. However, it is difficult in practice. Imagine using an ad-hoc controller that uses smaller step sizes to change the ideal throttle level (e.g., 0.1 instead of 1.0). As the number of discrete performance steps available rises, the steady-state error would reduce, but at the cost of increased settling time. While it may be possible in some cases to design an ad-hoc controller that works well in practice, proportional control is preferred because there are established techniques to provide theoretical guarantees on the control performance and in the case of our P controller there is no steady-state error.

7.4. Improved ad-hoc control

The ad-hoc controller of the previous section can be improved to not have positive steady-state error. In Figure 8, we show the result of running both the P controller and the ad-hoc controller at many set points from 180 W to 260 W. The results are collected by running P4MAX and collecting the long-term steady-state error observed after a few minutes. The P controller is able to precisely meet the set point with 0.1 W precision. However, the ad-hoc controller shows steady-state error that is often above the set point. At most it is 6.1 W above the set point. An improved ad-hoc controller that always runs at or below the set point is created by subtracting 6.1 W from the set point used. The figure shows that the improved ad-hoc controller with the safety margin does not violate the set point. We use this improved version of the ad-hoc controller in the rest of the paper.

7.5. Application Performance

In this section, we investigate the impact of closed-loop power control on the performance of common microprocessor benchmarks. We use the improved ad-hoc controller with the safety margin of 6.1 W because this allows both controllers to run with the same power constraints so that application performance can be compared. Without the safety margin, the ad-hoc controller would violate the power constraint for some workloads and show better performance than the P controller.

We ran *open-loop*, (improved) *ad-hoc*, and *P controller* under each power budget and recorded the throughput achieved. In Figure 9, we present the benchmark performance as a percentage of the throughput at full-performance. For example, a measure of 100% means the application ran at the same rate as it would in the 100% performance state (no power management). A measure of 50% 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. CPU2000 is divided into CINT2000 and CFP2000 which consist of integer and floating-point benchmarks, respectively. The reported result is the average for all benchmarks in the category.

Over the entire power budget range, application performance with the *P controller* is 31% to 82% faster than *open-loop* and up to 17% faster than *ad-hoc*. The *open-loop* policy runs applications at 29% to 76% of full-performance. The slowdown is very high because open-loop does not use real-time measurement and must select a single static speed at which to run the processors. The *improved ad-hoc* policy does much better and runs applications at 49% to 99% of full-performance. However, the *P controller* can do even better due to quicker settling times in response to changing power levels and more efficient modulation of the performance states in the processors. The *P controller* achieves between 53% and 100% of full-performance on the workloads across power budgets from 210 W to 250 W. Budgets beyond 250 W for *P controller*, cause performance to quickly converge to full-performance for all workloads.

Figure 10 summarizes the speedup of the P controller over other methods. The speedup is calculated as the throughput of workload under the P controller divided by the throughput of the workload under the other control mechanism. In general, the largest improvements are made at the lowest power budgets where the power constraints are the greatest. At the highest power budgets (240 W and 250 W), the improved ad-hoc and P controller policies ran most benchmarks near full performance because the workloads often run below the power set point at these levels.

8. Conclusions

In this paper we present a control-theoretic peak power management system for servers. We show that a relatively simple closed-loop controller provides better application performance under a power constraint than by using 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 can increase application performance by up to 82%. A key factor in realizing this performance improvement is having accurate power measurement which

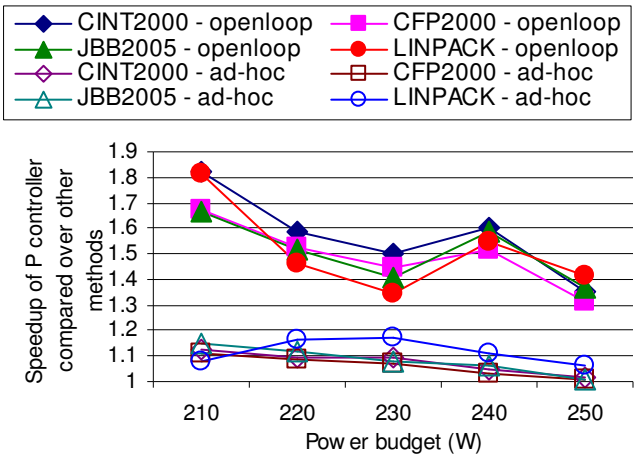

Figure 10: Speedup of P controller over other methods.

reduces controller design margins and utilizes the available power supply effectively.

We compared our controller to a widely used ad-hoc technique. In general our controller is superior because 1) it has no steady-state error, 2) it has much shorter settling time, 3) it has less actuation overhead, 4) it has guaranteed stability and predictable settling time even when the system model is not accurate, and 5) it provides a stability range which gives the designer confidence about the degree of variation that the control system can tolerate. The P controller runs applications at up to 17% faster than the ad-hoc controller.

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 changing power and thermal environments. In addition, it allows design-time safety margins to be reduced so that severs run closer to the limits of the available power supply constraints. In our blade, we could reduce label power from 308 W to 250 W with a minimal impact on the performance of real applications. We expect this technique will be applied to low-cost rack-mount and blade servers so that cost-effective power supplies with lower power ratings can be used. At another level, datacenter operators may use power control to match server power consumption to the available rack cooling capacity.

9. References

[1] C. Lefurgy et al., "Energy Management for Commercial Servers", *Computer*, vol. 36, no. 12, December, 2004.

[2] P. Bohrer et al., 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 et al., "Ecosystem: Managing energy as a first class operating system resource", *Int. Conf. on Architectural Support for Programming Languages and Operating Systems*, 2002.

[4] Y. H. Lu et al.. "Operating-system directed power reduction", *Int. Symp. on Low Power Electronics and Design*, 2000.

[5] J. Hellerstein et al., *Feedback Control of Computing Systems*, John Wiley & Sons, 2004.

[6] K. Skadron et al., "Control-Theoretic Techniques and Thermal-RC Modeling for Accurate and Localized Dynamic Thermal Management." In Proceedings of the Eighth International Symp. on High-Performance Computer Architecture, 2002.

[7] Q. Wu et al., 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), 2002.

[9] M. E. Femal and V. W. Freeh, "Boosting Data Center Performance Through Non-Uniform Power Allocation", In Proceedings of *2nd Intl. Conf. on Autonomic Computing*, 2005.

[10] V. Sharma et al., "Power-Aware QoS Management on Web Servers", In Proceedings of the 24th International Real-Time Systems Symposium (RTSS), Dec. 2003.

[11] Y. Chen et al., "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 et al., *Digital Control of Dynamic Systems*, 3rd ed., 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] X. Wang, C. Lefurgy, and M. Ware,"Managing Peak System-level Power with Feedback Control", IBM Research Technical Report RC23835, 2005.

[16] S. Norsworthy, R. Schreier, and G. Temes (Eds.), *Delta-Sigma Data Converters: Theory, Design, and Simulation,* Wiley-IEEE Press, 1996.

[17] D. Brooks and M Martonosi, "Dynamic Thermal Management for High-Performance Microprocessors", *Proceedings of the 7th Symp. on High Performance Computer Architecture* (HPCA-7), 2001.

[18] W. Felter et al., "A Performance-Conserving Approach for Reducing Peak Power Consumption in Server Systems", *Proceedings of the International Conf. on Supercomputing,* 2005.

[19] C. Poirier et al., "Power and Temperature Control on a 90nm Itanium-Family Processor", In proceedings of *Intl. Solid State Circuits Conf.*, 2005.

[20] IBM Systems, *IBM PowerExecutive 1.10 Installation and User's Guide Version 1.10*, 2nd ed., June, 2006.

[21] Intel, *Dual-Core Intel Xeon Processor 5100 Series Thermal/Mechanical Design Guide*, June, 2006.

[22] B. Colwell, "We May Need a New Box", *Computer*, March, 2004.

[23] P. Ranganathan et al., "Ensemble-level Power Management for Dense Blade Servers", *Proceedings of the 33rd Annual Intl. Symp. on Computer Architecture* (ISCA), 2006.