A Gray-box Feedback Control Approach for System-Level Peak Power Management

Jiayu Gong and Cheng-Zhong Xu
Department of Electrical and Computer Engineering
Wayne State University, Detroit, MI 48201
{jygong, czxu}@wayne.edu

Abstract

Power consumption has become one of the most important design considerations for modern density servers. To avoid system failures caused by power capacity overload or overheating, system-level power management is required to control power consumption precisely. Conventional solutions to this problem mostly rely on feedback controllers that treat the servers under consideration as a black-box. Such black-box approaches may not respond to the variation of system quickly since they are unaware of the system usage which determines the power consumption. This paper presents a gray-box strategy that assumes a model-predictive feedback controller based on a pre-built power model and a performance prediction model to constraint the peak power consumption of a server. In contrast to the existing strategies, this gray-box approach uses the performance events, which bring more insights of the behaviors and power consumption of a system, for the purpose of model prediction.

A prototype of this controller is implemented and evaluated using SPECweb2005 benchmark on a web server. This controller can settle the power consumption below the power cap within 2 control periods for more than 75% of the power overloading regardless of workload variations, outperforming black-box approaches. Meanwhile, the performance of application can be maximized with this controller.

1 Introduction

The increasing power consumption of servers poses a key challenge in enterprise data centers. It not only leads to a greater probability of thermal failover impacting the availability and reliability of systems, but also increases electricity costs dramatically. Thus, online power management is desirable.

Traditionally, servers are designed to provide for the worst-case scenario by over-provisioning which adds costs with only few benefits for the real environments. Most of the work in this context focused on reducing power consumption by either improving the energy-efficiency of individual server components [27] or providing an efficient power management strategy to balance the power and QoS guarantee [22][26]. In contrast, a "better-than-worst-case" design approach has been adopted by limiting power consumption (power capping) [10]. Enforcement of power limits can be physical which concerns about power outage due to overloading of electrical circuits, or contractual which considers the economic penalties for exceeding the negotiated load [15]. In addition, power capping is a key element for implementing power shifting, which is a dynamic setting of power budgets for individual servers so that a global power cap for a cluster can be maintained; see [33][15] for examples.

In general, power consumption increases with system processing, networking and storage usage. For web servers, it is difficult to predict the system usage in advance due to the nature of variability or even bursts of network traffic. This poses a challenge to achieve the goal of maximizing performance under power cap. A general approach is to employ feedback control that regulates system configuration dynamically in response to the error between measured power and reference power; see [31][12][28][39] for examples.

Though existing feedback controller for power management can assure stability and accuracy, the settling time is largely overlooked. In the context of power capping, the settling time can be interpreted as how long it will take for the system to operate within a certain range of power limit. To avoid power overloading, this term is more emphasized on the time for a system to decrease the power to the power cap when it is out of the limit. However, this settling time could be long for the existing work. For example, it was reported that a Proportional (P) controller for DVS (Dynamic Voltage/Frequency Scaling)-capable or clock-throttling-capable processor power management would require as many as 13 control periods to settle to the power cap [28]. While the length of control period is typically determined by the sampling rate of power measurement which is only 1Hz or 2Hz for most widely-used external power meter, this settling time remains too long in contrast to the tolerable overloading time which usually is one or several seconds. It could be even longer for a web server with highly variable traffics. To accelerate the process of settling, we design a Proportional-

Integral-Derivative (PID) controller in this paper. However, the settling time may still be long occasionally.

To shorten the long settling time, furthermore, we propose a gray-box approach for management of system-level peak power consumption of a server while maximizing system performance. Instead of relying on feedback control which is reactive to the measured power, we consider a proactive approach which can predict the optimal solution. The prediction module predicts the highest CPU speed at which a server can run within the power cap, with the help of performance events, including OS-level performance metrics and hardware performance counters. A feedback controller makes adjustment according to the summated errors. Then the selection of CPU speed is based on both the prediction and feedback control. The contributions of this paper are two folds in summary. The first is to construct a power estimation model and a performance impact model for a web server by using performance events. The second one is a model-predictive feedback controller to control system-level power consumption while maximizing performance. The experimental results show the gray-box approach can be more responsive in power control than the black-box feedback controllers by letting more than 75% of the power overloading settle to power cap within 2 control periods. Meanwhile, the degradation on system performance is minimal using this gray-box controller in contrast to the black-box controllers.

The rest of this paper is organized as follows. In Section 2, we introduce the basics of black-box feedback control in power management. We present our gray-box strategy to control system power in Section 3 and construct models in Section 4. The empirical results are shown in Section 5. Related work is discussed in Section 6. We draw conclusion in Section 7.

2 Black-box Feedback Control for Power Management

In this section, we first discuss the feedback control in the context of power management. Then we present the design of a PID controller for power capping.

2.1 Overview of Feedback Control

Feedback control is to use measurements of a system's outputs to adjust the system control inputs in order to achieve externally specified goals. In the context of power management of a system, the output is the measured power and the reference input is the power cap. The control input, also known as the control knob, mostly focuses on the power management techniques supported by processors. This control knob can be frequency and voltage (P-state) using Dynamic Voltage/Frequency Scaling (DVFS), or sleep states (C-state) that reduce power consumption when parts are idle, or the throttle state (T-state) [28]. In this paper, we use CPU frequency (P-state) as the control knob, called CPU speed. The power is measured and fed back for use in the control computation to get the control input for the next control period so that the measured power in the next control period can be settled to the power cap.

Applying feedback control in power capping, we need to consider the following properties. First is the system stability. In a server with highly variable traffic, the measured power oscillates due to the change of resources demand. Applying a feedback controller, this oscillation still exists since the dynamics of the system cannot be removed. This system is Bounded-Input-Bounded-Output (BIBO) stable since the output is bounded.

Second, the measured power can converge to the power cap with the controller, which is called accuracy. However, it is difficult for a system to always operate sufficiently close to the power cap due to the system dynamics. In this scenario, we would emphasize on that the measured power could be settled below the power cap since the main purpose of the power controller is to avoid power overloading.

Third, this convergence process should be short. The time taken for this process is the settling time. On one hand, fast settling to power cap when power consumption is below the cap can fully utilize server resources. On the other hand, responsive adjustment to satisfy power cap when power consumption is beyond the cap will reduce the possibilities of both outage and exceeding of negotiated load. For a web server, this property is of particular importance in the presence of highly variable traffic. The traffic variation requires different amounts of system resources over time which leads to fluctuation of system power consumption. In power capping, to avoid power overloading, the settling time would be focused on the time for a system to take to operate below the power cap when the power is beyond the cap.

Fourth, using the controller will not let the measured power exceed the power cap greatly.

The first two properties were widely studied in previous studies while the last two were largely overlooked. In our work, we focus on the property of settling time since it is closely related to the system performance and system failure due to power overloading.

Lefurgy et al. [28] designed a Proportional (P) controller with a first-order delta-sigma modulator to control peak power for a server, based on a linear model between maximal power consumption and CPU speed. This controller works well for systems with statical and predictable workloads, like Linpack benchmark [24]. In case of dynamic workloads, this controller may not perform well.

We applied this P controller on a web server running SPECweb2005 benchmark [4]. SPECweb2005 defines a representative workload cluster of applications for web servers. The system power consumption changes with workload which can be reflected by the number of clients. We used the power model for 1500 clients as the nominal system model and found this P controller theoretically can let power settle in 5 seconds (assuming the control period is 1 second) for the scenario of 900 clients. But from experiment results, it may

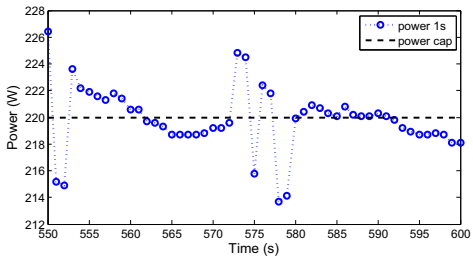


Figure 1. Example of a long settling time of P controller.

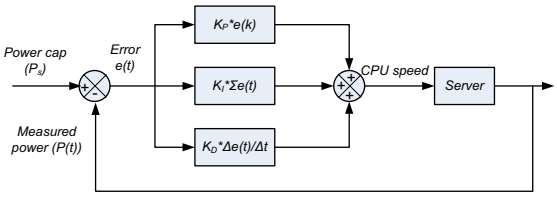


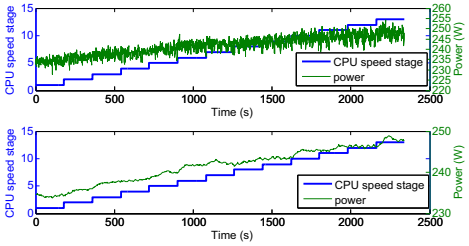
Figure 2. PID based power cap controller.

take as long as 8 seconds or even longer to settle power consumption below power cap, as shown in Fig. 1. This is due to the variability of Internet traffic that leads to large variability of power consumption in contrast to the static power consumption when running CPU-intensive workloads used in [28]. In this case, the controller should not be designed based on the maximal power model. Using the first-order delta-sigma modulator will scale the CPU speed frequently even within one control period, which is not expected in a stable system. Thus we design the controller to limit power consumption without this modulator.

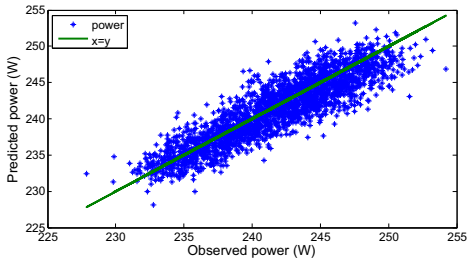
2.2 Design of the PID Controller

We design a PID controller, as illustrated in Fig. 2, based on the model that studies the relationship between current power consumption and CPU speeds. This relationship for a web server running SPECweb2005 with 1500 clients is depicted in Fig. 3(a). Fig. 3(a) plots the instant and average power (averaged over a 1-minute interval) under different CPU speed settings after ramp and warm-up phases. The CPU speed stage is increased by one stage every 180s.

Let $p(k)$ and $s(k)$ be the power consumption and CPU speed of the server in the k^{th} control period, respectively. In the k^{th} control period, given $p(k)$, the control goal is to choose a CPU speed $s(k)$ so that $p(k)$ can converge to the power cap in a finite number of control periods. We use the differences of $p(k)$ and $s(k)$ with their operating points to build a linear model. The control output of this model is $\hat{p}(k) = p(k) - \bar{p}$, the control input is $\hat{s}(k) = s(k) - \bar{s}$, and the desired value of the control output is $\hat{P}_s = P_s - \bar{p}$, where $\bar{p}$ and $\bar{s}$ are operating points, and P_s is the power cap, also



(a) Data used in system identification



(b) Model evaluation

Figure 3. Data for system identification and model evaluation.

known as set point. The operating points are desired steady values of CPU speed and power consumption in the power management context. The system model is:

$$\hat{p}(k) = a\hat{p}(k-1) + b\hat{s}(k-1), \quad (1)$$

where a and b are system parameters. We estimated these system parameters by system identification using the nomial model shown in Fig. 3(a). The middle value of a range of possible CPU speed settings is selected as the operating point of CPU speed, $\bar{s}$, while the average power consumption under this setting is $\bar{p}$. Using Least Square Method, we get the system parameters for Eq. (1), $a = 0.4093$ and $b = 0.6450$. Fig. 3(b) plots the measured and predicted power. To indicate how well the model fits, the coefficient of determination is a widely-used term, denoted as R^2. The R^2 value in this model is 0.84.

The time domain form of this PID controller is:

$$\hat{s}(k) = K_P e(k) + K_I \sum_{j=1}^{k} e(j) + K_D[e(k) - e(k-1)], \quad (2)$$

where $e(k) = \hat{P}_s - \hat{p}(k)$, K_P, K_I and K_D are control parameters. We use Root-Locus method [20] to design this PID controller. Constraining the settling time to be 1 control period can lead to the maximum overshoot of around 3%. By studying the root locus of this PID controller, we find the settling time can be 1 if the derivative term is very close to zero. So we set $K_D = 0$. Thus this PID controller is reduced to a PI controller with $K_P = 0.6322$ and $K_I = 1.5478$. The poles [20] that determine the stability of the closed-loop system are $0.00016 \pm 0.0241i$, which are inside the unit circle. Thus this closed-loop system is stable. The accuracy is guar-

anteed by the PI controller itself.

We analyzed the performance of this PI controller for the scenarios of 900 and 2100 clients as well. The stability and accuracy are guaranteed. The settling time is 7 for 900 clients and 2 for 2100 clients, close to the theoretical results using previous P controller. In practice, we can see this PI controller can lead to shorter settling time in contrast to the P controller in Section 5. However, it may still need more than 5 seconds occasionally. Thus we need a controller to control system power more responsively.

3 A Gray-box Approach

In this section, we will first present the architecture of the gray-box approach to manage peak power of a web server. Then a model-predictive feedback controller is developed based on this approach. We present the model prediction components of this controller by proposing two models.

3.1 Architecture

The architecture of this gray-box approach is shown in Fig. 4. It consists of two loops. One is a feedback loop that sums up the errors between the power cap and the measured power and adjusts the future CPU speed setting. The other is a model prediction loop that predicts the control input for the next control period.

The feedback loop employs a controller to sum up the error and adjust the future CPU speed. An external power meter is used to measure power. To predict the future CPU speed setting, this approach relies on gray-box monitoring to collect performance events, including both OS-level metrics and hardware performance counters. The collected performance events are used by a power model and a performance prediction model. The CPU speed for the next control interval is predicted by estimating power consumption at different speeds based on these two models. The one which can lead to the largest power consumption but below the power cap is the predicted speed. The feedback controller will adjust CPU speed according to both summated errors and the predicted speed. Because of the model prediction function, this approach is proactive, which means it is not only reactive on the measured power but also can predict the control input based on the usage statistics.

To distinguish the power controller based on this gray-box approach from other ones, we name it M-controller since it is a model predictive feedback controller.

3.2 Controller Design

Based on the results in Section 2, the derivative term can be omitted for the proposed PID controller. Thus, Eq. (2) can be re-written as:

$$s(k) = \bar{s} + K_P e(k) + K_I \sum_{j=1}^{k} e(j), \tag{3}$$

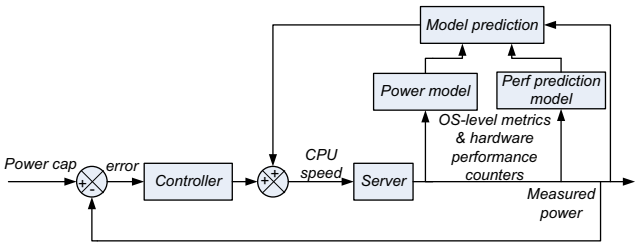


Figure 4. Power control loop.

where K_I and K_P determine the adjustment of the CPU speed corresponding to the current and past errors. Let f denote a model predictive CPU speed based on the power model and performance prediction model of a server. It follows that the CPU speed of the k^{th} control period is:

$$s(k) = f(k) + \alpha\hat{g}[K_P e(k) + K_I \sum_{j=1}^{k} e(j)], \tag{4}$$

where $\hat{g}$ is a model-predictive scaling factor of CPU speed with respect to the change of output error in the power consumption, and α is to tune the weight of this scaling factor. A controller with a large control factor can respond quickly to the errors. But it may cause oscillation. To reduce the overshoot, we prefer a smaller control factor by setting α to be a small value (e.g., $\alpha = 0.1$).

3.3 Model Prediction

The gray-box approach periodically predicts the future power consumption under different CPU speed settings to improve the controller's agility by monitoring performance events. This prediction is based on two models: power model and performance prediction model.

3.3.1 Power Model

We build a power model by correlating a few OS-level metrics and hardware performance counters with power consumption of a system.

Economou et al. [13] presented Mantis which provides a fast and accurate full-system power prediction model by low-overhead OS utilization metrics and performance counters. In our work, we adapt the approach in Mantis to construct power model with more analysis on the details of CPU utilization. We use the following OS-level metrics and performance counters to construct the power model:

- CPU utilization (u_{cpu}): It consists of utilizations occurred while executing at user level, at system level, at user lever with priority, and to wait for I/O (idle). It can be represented by $100\% - idle\%$.
- CPU I/O wait (u_{iowa}): When the CPU stalls due to an outstanding of disk I/O requests, the power consumption in the circuit will decrease. We distinguish this portion of CPU time, which is different from Mantis.

- L2 cache miss (u_{L2miss}): Memory will be accessed if L2 cache cannot hold the whole data set for a workload. This metric indicates the activity of memory.

- Disk read/write rate (u_{disk}): Existing research [7][15] showed the power consumption of disk is relatively small compared with overall full system power. This part of power consumption depends on the disk activity which can be represented by the read/write rate.

- Network rate (u_{net}): The power consumed on network depends on the network I/O rate.

We assume this power model is linear as follows:

$$P_{system} = C \bigotimes U + P_{leak}, \tag{5}$$

where $C = \left[c_{cpu}, c_{iowa}, c_{L2miss}, c_{disk}, c_{net}\right]$, $U = \left[u_{cpu}, u_{iowa}, u_{L2miss}, u_{disk}, u_{net}\right]^T$, and $\bigotimes$ is an inner product operator. C represents the coefficients of this linear model to be determined. Different CPU speeds have different power models. The measurements of the metrics in vector U will be discussed in Section 4.

3.3.2 Performance Prediction

The system behaviors would be affected due to CPU frequency scaling, which can be observed by performance events. We designed a model to predict the future OS-level metrics and hardware counters under different CPU frequencies.

Let $U(k)$ denote the vector of metrics in the k^{th} control period. Without loss of generality, we index the elements in $U(k)$ by an integer from 1, i.e., $U(k) = \left[u_1^k, \cdots, u_n^k\right]^T$. CPU speed stages are indexed by h_i, $i \in \{1, 2, \cdots\}$. The cardinality of $U(k)$ is determined by the number of metrics monitored. We use the following formula to represent the transition of metrics vectors if we change CPU speed from h_i to h_j:

$$U(k+1) = A^{i,j} \times U(k), \tag{6}$$

where $A^{i,j} = \begin{bmatrix} a_{11}^{i,j} & \cdots & a_{1n}^{i,j} \\ \vdots & \ddots & \vdots \\ a_{n1}^{i,j} & \cdots & a_{nn}^{i,j} \end{bmatrix}$. $A^{i,j}$ defines the effect on metrics if CPU speed is scaled from f_i to f_j, which can be obtained using Least Square Method (LSM) given $U(k+1)$ and $U(k)$.

The transition matrix $A^{i,j}$ $(i < j)$ can be calculated as:

$$A^{i,j} = A^{i,i+1} \times A^{i+1,i+2} \times \cdots \times A^{j-1,j}. \tag{7}$$

4 Model Construction

In this section, we first present the experimental environment for the construction of the models. Then we show the estimation of parameters of the models.

4.1 Experiment Environment

Experiments were conducted on a Dell PowerEdge1950 server with two Intel quad-core Xeon E5450 processors, 8GB memory, one 250GB SATA hard disk, and 1Gb Ethernet interface. The processors could be operated at four frequencies ranging from 2.0GHz to 3.0GHz. All the cores can run at different speed in pair. There are 13 possible CPU speed stages in total, indexing from 1 to 13 in an ascending order. To scale CPU speed, we can write the new frequency level in to a system file. A BIOS routine will periodically check this file and reset the CPU frequency accordingly. The average overhead for the BIOS to change frequency was less than 1ms according to our experimental results.

We patched Linux kernel 2.6.18 with perfctr 2.6.36 [3] so that we can read hardware performance counters on-line with relatively small overhead compared with OProfile [2]. We use SAR [5] to collect run-time OS-level metrics for CPU utilization, hard disk I/O, and network I/O. The power meter we use to obtain the power consumption value is WattsUp Pro [1]. This power meter has an accuracy of $\pm 1.5\%$ of the measured RMS power with a sampling rate of 1Hz (in practice, the power budget might be reduced by 2% measurement error). The workload was SPECweb2005 [4]. SPECweb2005 is a benchmark for evaluating the performance of World Wide Web Servers. The execution of requests in SPECweb2005 involves processor, disk I/O and network I/O for dynamic content generation and static image serving.

4.2 Model Parameters Estimation

While modern processors provide counters for measuring a large number of different events, the number of events that can be counted concurrently is typically quite small (2-4) if we don't use a multiplexing technique. In our experimental environment with Intel Xeon 5450 Quad core which is Intel Core 2 architecture, there are only two programmable counters. Since both the L2 cache miss and L1 cache miss can only be counted in a specific performance counter (pmc0) in the Intel Core 2 architecture, we use the number of L2 cache miss instead of L2 cache miss ratio.

In our experiment, SPECweb2005 benchmark will read from hard disk frequently and let it always be active. The variation of power consumption for hard disk is very small when it is active. In our power model, we can see the coefficient for this variable is almost 0. So we simplify the model (5) by removing u_{disk} from U and c_{disk} from C.

The related performance events with power consumption were collected during a set of runs under different CPU speed settings in order to obtain the coefficients of the power model. We use pace regression to solve this regression problem. Pace Regression [38] improves on classical ordinary least squares (OLS) regression by evaluating the effect of each variable and using a clustering analysis to improve the

statistical basis for estimating their contribution to the overall regression. WEKA [17] provides an API for user to draw the pace regression model from an input data set. The linear models fit well ($R^2 > 95\%$) for all CPU speed stages.

To obtain the transition matrices for power prediction model, we collected traces by scaling the CPU speed to its next step and scaling back to the original setting periodically in off-line training. The difference of performance events between two periods can be attributed to both the CPU frequency scaling and the change of workload. Due to the variability of the workload, the effect caused by the workload cannot be neglected. Thus we need to pre-process the collected traces using measured power. Since the power consumption will increase monotonically with scaling up CPU speed in an ideal scenario, we will remove the data that violate this principle. In addition, we filter the traces by checking the gap between the system power before scaling and that after scaling as well. If this gap is larger than a threshold, e.g. 5%, relative to before-scaling system power, we attribute this big gap to the significant variability of workloads and will not use the corresponding data for training model.

5 Evaluation

In this section, we present the experimental resultsfor the gray-box approach. We validate the power model and performance prediction model. We compare our approach against several other representative power controllers in terms of responsiveness and impact on application performance.

5.1 Experimental Methodology

The M-controller was implemented within the to-be-controlled server instead of using another dedicated machine to communicate with the controlled server periodically. The CPU utilization of running this controller is less than 0.5%.

We conducted experiments to show the responsiveness and impact on application performance of M-controller compared with the existing P, PID and an ad-hoc controller which is a representative power contoller in industry. We followed the way in [28] to design P controller. The PI controller was designed in Section 2. The basic idea of ad-hoc controller is to simply raise or lower CPU speed by one step depending on if the measured power is lower or higher than the power cap. All the controllers were designed based on the system identification for a run of SPECweb2005 with 1500 clients. The execution time for a run of SPECweb2005 was set to 5000 seconds with 180 seconds ramp-up time and 300 seconds warming time. All the controllers were evaluated in the scenarios of 900, 1500, and 2100 clients. The number of clients reflects the load and bursts of a server.

5.2 Model Validation

We used the power model to estimate power consumption for running SPECweb2005 with different number of clients.

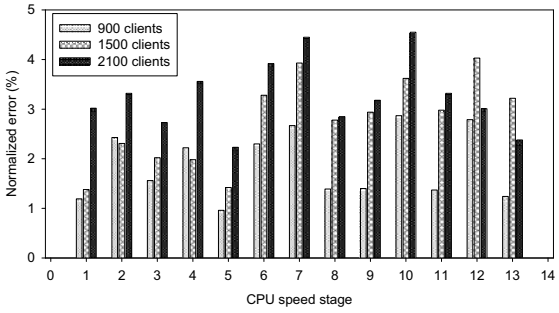


Figure 5. Average error of power model.

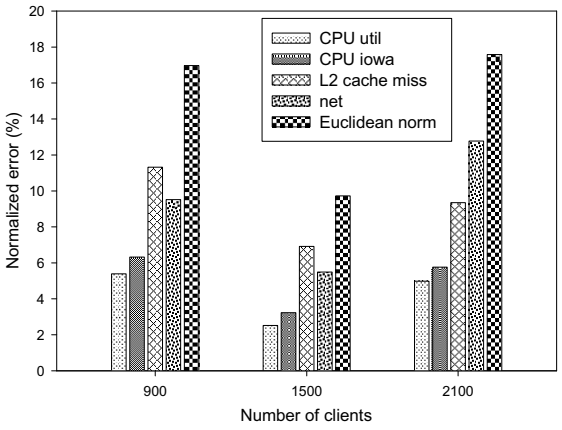


Figure 6. Average error of performance prediction model for CPU speed scaling from stage 5 to stage 6.

Fig. 5 shows the accuracy of our proposed power model. The average errors of power models under different CPU speed settings range from around 1% to 4.8%. The results show the high accuracy of this power model.

We evaluated the accuracy of the performance prediction model as well. Due to space limit, we only take one scenario, from speed stage 5 to 6, for example. Fig. 6 shows results of performance prediction. We can see the prediction errors on the metrics related to CPU range from around 2% to 6%. The prediction errors on other metrics can be larger than 10%. We use Euclidean norm (magnitude) as a collective metric to summarize the errors of all performance events since the magnitude can represent the length of a vector consisting of prediction errors of different metrics. The magnitude of normalized error for predicting a vector of metrics can range from 9% to 17%.

5.3 Controller Responsiveness

In this part, we investigated the responsiveness of M-controller in comparison with other controllers.

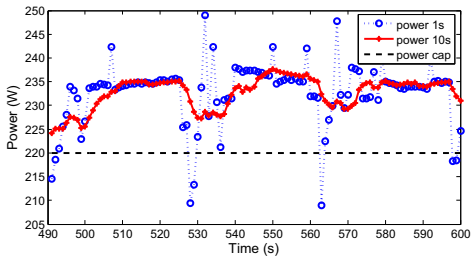


Figure 7. System power without power controller in the case of 900 clients.

The power caps were selected close to the average power of running workload at the middle CPU speed stage. The power caps are set to 220W, 240W, and 260W for 900, 1500, and 2100 clients, respectively. The data were collected after ramp-up and warming time. We take the scenario of 900 clients for example. The results for 1500 clients and 2100 clients are not presented for brevity since similar observations can be made.

The power consumption of a server running at a full speed may exceed the power cap for a long period, as shown in Fig. 7. Both the instant power consumption (1 second) and average power consumption (10 seconds) are plotted. The power consumption would fluctuate due to the high dynamics.

Fig. 8 plots the system power due to the deployment of different controllers. The controllers were enabled at the 500^{th} second. The peak power consumption of a system can be controlled as most time the average power is below the power cap. Because of the oscillations of power caused by the workload itself, the system power consumption cannot always stay at the power cap even with power controllers.

The ad-hoc controller raises or lowers CPU speed by one step at a time. Thus it is slow to respond to the power change. The average power is far below the power cap, which implies the ad-hoc controller is more conservative. The P controller could let the system operate close to power cap faster but it took a long period to settle, from 552s to 560s, as shown in Fig. 8(b). The PI controller has similar effect to the P controller. Using the M-controller can accelerate the process of settling to power cap while the oscillations still exist. The average power by using M-controller is close to the power cap and the instant power could be pulled back to the power cap more quickly when it exceeds the power cap in contrast to those using P and PI controller.

To quantify the responsiveness of these controllers, we analyzed the lengths of power violations after the controllers have been activated. Power violation is defined as that the measured power exceeds the power cap. Since the main purpose of these controllers is to limit the power usage, more responsiveness means shorter length of power violations. The effect of settling to power cap when measured power is below power cap can be reflected in the impact on performance,

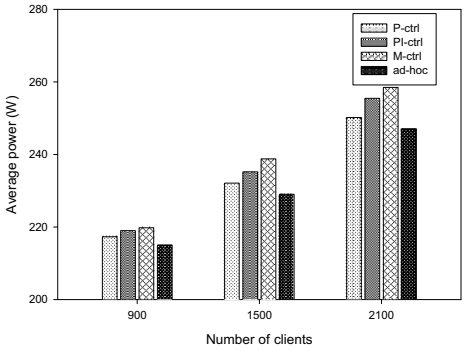


Figure 10. Average power.

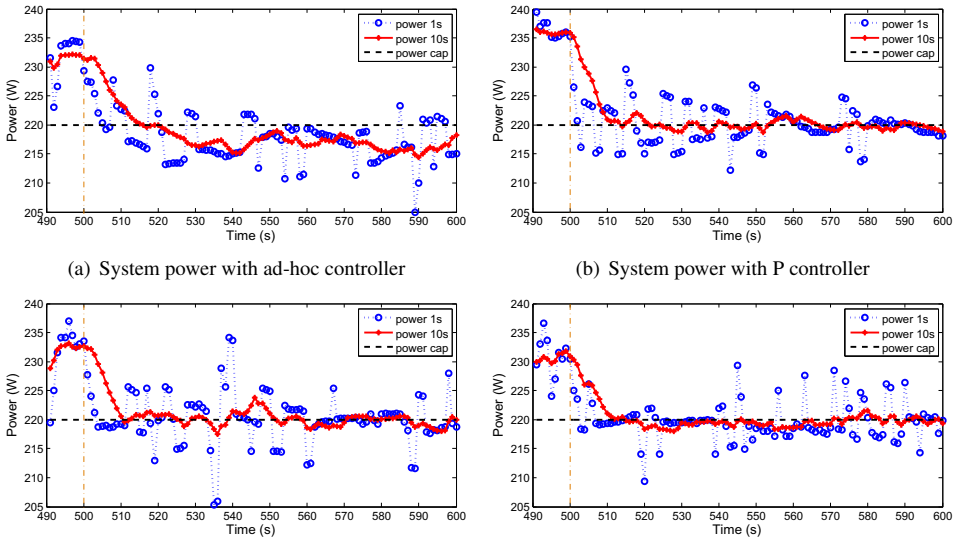
which is not discussed here. Fig. 9 shows the cumulative distribution function (CDF) of lengths of power violations for different settings. We can see more than 75% of power violations will have the length of no longer than 2 for M-controller, which means M-controller is responsive. In case of 900 and 2100 clients, M-controller is more responsive than the others and PI controller is more responsive than P controller. In case of 1500 clients, we can see the P and PI controllers are almost as responsive as, or more responsive than M-controller. It is because these two controllers are built based on system identification of a run of 1500 clients. Overall, M-controller is the more responsive than others.

5.4 Impact on Performance of Application

We investigated the impact of these controllers on the performance of running application from two perspectives. One is the measured performance directly. The other is the system power consumption.

Take the scenario of 900 clients for example. We changed the values of power cap from 215W to 225W in a step length of 2.5W. Fig. 11 plots the impact on application performance in terms of slowdown. Slowdown is defined as a percentage of the average response time increase compared with that at in a full-speed run. When the power budget constraint is stringent, the performance degradations for all controllers are remarkable, which are all over 14%. M-controller outperforms others in terms of slowdown by 5% when the power budget is 215W. The performance degradation for all controllers decreases with power budgets. Amongst all controllers, M-controller leads to the least performance degradation. When the power budget constraint is loose, the gap among the slowdown of P, PI and M-controller becomes marginal. Ad-hoc controller always has the largest performance loss. Similar results can be observed for the scenarios of 1500 and 2100 clients.

Fig. 10 shows the average power consumption for all controllers. A system with M-controller always has the largest average power. The results validate that more power lead to better performance. M-controller maximizes the power usage so it can lead to the least performance degradation. Ad-



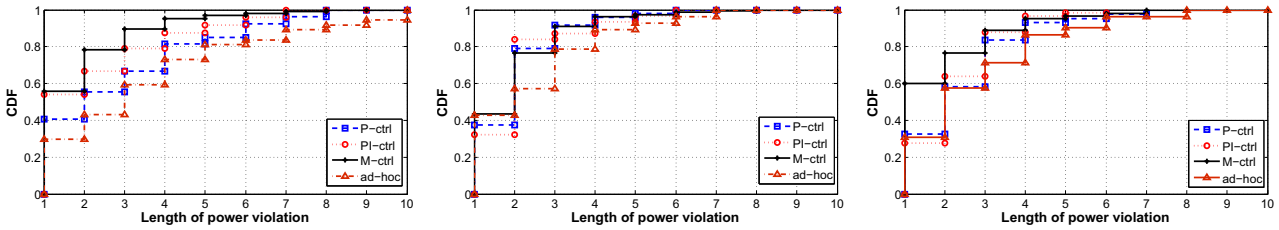
(a) System power with ad-hoc controller

(b) System power with P controller

(c) System power with PI controller

(d) System power with M controller

Figure 8. Performance of controllers in the case of 900 clients (turn on controllers at the 500^{th} second).



(a) 900 clients

(b) 1500 clients

(c) 2100 clients

Figure 9. CDF of length of power violations

hoc controller has the smallest average power which makes it have worst performance. P and PI controllers have close average power which explains why the performance degradation of these two controllers are very close.

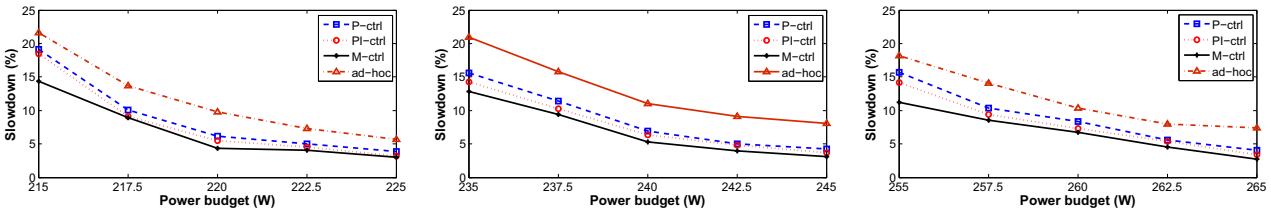
6 Related Work

A lot of existing working focused on how to reduce power consumption by improving the energy-efficiency of individual server components, from processors [25][21], to memory [14][32] and disk [46]. To address power management on system-level, Lu et al. [29] presented a power reduction technique at OS-level using task-based power management. Zeng et al. [43][42] proposed to build an explicitly energy-aware operating system by introducing a system-wide abstraction for the energy used. The purpose was to budget the energy available to individual processes. Different from these above work, our work is at system level without investigating individual task/process. Felter et al. [16] studied *power shifting*, an open-loop control to shift power between processor and memory components to maintain a server power budget. In contrast, our work uses the power measurement of the entire system for a closed-loop control.

Feedback control approach has been widely applied in power management. Lu et al. [30] described a formal feedback control algorithm combined with DVS for multimedia systems. Zhu and Muller [47] combined feedback control theory with DVS schemes for hard real-time systems with dynamic workloads. Both work are on the processor level using DVS while our work is to control system-level power.

Sharma et al. [36] applied control theory to control application-level quality of service requirements. Chen et al. [8] developed a controller to control SLA, which is response time, in a server cluster. Both of these work provide no guarantee to the power consumption of a computing system, which differs from the target of our work.

Minerick et al. developed a feedback controller to manage the average power consumption of a laptop to prolong battery lifetime regardless of current system resource usage [31]. Femal et al. [12] proposed a two-level framework for



(a) Slowdown: 900 clients (b) Slowdown: 1500 clients (c) Slowdown: 2100 clients

Figure 11. Slowdown of application performance with the reduction of power cap.

controlling cluster-wide power while applying the controller from [31] to limit the power consumption of each server.

Lefurgy et al. [28] presented a technique to control the peak power consumption of a server by a feedback controller using precise, system-level power measurement. As shown in our work, this controller may not settle the power quickly in a dynamic system. This work was extended to cluster-level [26] to control power for each server.

Model-predictive feedback controller can be employed for performance guarantee. Xu et al. [41] presented an end-to-end QoS provisioning framework to monitor and control user-perceived QoS. This work relies on queueing model. In contrast, our work is based on the models constructed by profiling. Similar profiling approach could be found for the purpose of system-wide energy optimization [44][45].

Performance monitoring counters (PMCs) are widely used to indicate the system performance [34][35]. Using performance monitoring counters (PMCs) for power management can be traced back to [6]. Contreras et al. [11] proposed a linear power estimation model using hardware performance counters to estimate run-time CPU and memory power consumption using Intel PXA255 processor. Because the number of counters used is more than that the processor can provide concurrently, this model is not feasible for online power estimation. Fan et al. [15] proposed a linear model considering CPU utilization and an empirical term which minimize squared root to improve accuracy. Heath et al. [19] presented a linear model using disk utilization in addition to CPU utilization. Both models ignore the impact on power consumption from other subsystems. Bircher and John [7] proposed power models for entire systems using PMCs. It needs to decompose the measurement of power for each subsystem physically. Economous et al. [13] presented a full-system power prediction model by adding CPU performance counters to the OS-reported CPU and disk utilization. We complement this approach by analyzing details of CPU utilization for different CPU speed.

There are a number of work using PMCs as a guide to predict performance impact due to CPU frequency scaling. Process cruise control [40] used PMCs to index a pre-computed table of frequency settings which led to a constant performance impact. A more flexible technique, using on-line regression to calculate the ratio of off-chip to on-chip cycles was proposed, in [9][23]. These work require computational overhead and can only work for processors with in-order single-issue pipelines. For processors with out-of-order design, which are used in server environments, on-chip execution can continue in the case of off-chip stalls. So the above approaches may not work for server environments. The work in [37][18] proposed execution time models to predict the performance impact due to frequency change. The performance was studied in terms of execution time. In contrast, our work is to find the impact on OS-level metrics and hardware counters due to CPU frequency scaling.

7 Conclusions

In this paper, we present a gray-box strategy to control the system-level power consumption for a web server while maximizing system performance. This approach employs a model prediction module to predict CPU speed and a feedback controller to adjust CPu speed. The prediction relies on a power model and a performance prediction model using both OS-level metrics and hardware performance counters.

We compared our proposed power controller with existing feedback controllers. It can achieve more responsive control on system power consumption with minimal workload performance degradation for web server environment.

However, this power controller requires more elaboration. The models need to be built with comparatively large cost of trace collection and off-line training. It requires periodical collection of performance events in contrast to the existing controllers which only need real-time power measurement.

The power model is independent of workloads but the performance prediction model is application-domain specific. Our power controller is proposed to control system power consumption for a web server. For different applications, different performance predictions model are needed. How to derive or adjust these two models on-the-fly without imposing costly overhead is a key challenge for the wide use of this gray-box strategy.

Acknowledgement

We would like to thank the anonymous reviewers for their constructive comments and suggestions. This research was supported in part by U.S. NSF grants CNS-0702488, CNS-0708232, CNS-0914330.

References

[1] Electronic educational devices inc., "watts up pro power meter". http://www.wattsupmeters.com.

[2] Oprofile. http://oprofile.sourceforge.net/.

[3] perfctr. http://user.it.uu.se/~mikpe/linux/perfctr/.

[4] Specweb2005. http://www.spec.org/web2005/.

[5] sysstat. http://pagesperso-orange.fr/sebastien.godard/.

[6] F. Bellosa. The benefits of event-driven energy accounting in power-sensitive systems. In *ACM SIGOPS European Workshop*, 2000.

[7] W. L. Bircher and L. K. John. Complete system power estimation: A trickle-down approach based on performance events. In *ISPASS*, 2007.

[8] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam. Managing server energy and operational costs in hosting centers. *SIGMETRICS Perform. Eval. Rev.*, 2005.

[9] K. Choi, R. Soma, and M. Pedram. Fine-grained dynamic voltage and frequency scaling for precise energy and performance trade-off based on the ratio of off-chip access to on-chip computation times. In *DATE*, 2004.

[10] B. Colwell. We may need a new box. *Computer*, 2004.

[11] G. Contreras and M. Martonosi. Power prediction for intel xscale®processors using performance monitoring unit events. In *ISLPED*, 2005.

[12] M. E. Femal and V. W. Freeh. Boosting data center performance through non-uniform power allocation. In *ICAC*, 2005.

[13] D. R. S. Economous, C. Kozyrakis, and P. Ranganathan. Full-system power analysis and modeling for server environments. In *MoBS*, 2006.

[14] X. Fan, C. Ellis, and A. Lebeck. Memory controller policies for dram power management. In *ISLPED*, 2001.

[15] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *ISCA*, 2007.

[16] W. Felter, K. Rajamani, T. Keller, and C. Rusu. A performance-conserving approach for reducing peak power consumption in server systems. In *ICS*, 2005.

[17] S. R. Garner. Weka: The waikato environment for knowledge analysis. In *the New Zealand Computer Science Research Students Conference*, 1995.

[18] R. Ge, X. Feng, W.-c. Feng, and K. W. Cameron. Cpu miser: A performance-directed, run-time system for power-aware clusters. In *ICPP*, 2007.

[19] T. Heath, B. Diniz, E. V. Carrera, W. M. Jr., and R. Bianchini. Energy conservation in heterogeneous server clusters. In *PPoPP*, 2005.

[20] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury. *Feedback Control of Computing Systems*. John Wiley & Sons, 2004.

[21] S. Herbert and D. Marculescu. Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In *ISLPED*, 2007.

[22] T. Horvath, T. Abdelzaher, K. Skadron, and X. Liu. Dynamic voltage scaling in multitier web servers with end-to-end delay control. *IEEE Trans. Comput.*, 2007.

[23] C.-H. Hsu and W. chun Feng. Effective dynamic voltage scaling through cpu-boundedness detection. In *PACS*, 2004.

[24] Intel. Intel math kernel library 10.1 - linpack. http://www.intel.com/cd/software/products/asmo-na/eng/266857.htm.

[25] C. Isci, G. Contreras, and M. Martonosi. Live, runtime phase monitoring and prediction on real systems with application to dynamic power management. In *MICRO*, 2006.

[26] J. O. Kephart, H. Chan, R. Das, D. W. Levine, G. Tesauro, F. Rawson, and C. Lefurgy. Coordinating multiple autonomic managers to achieve specified power-performance tradeoffs. In *ICAC*, 2007.

[27] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. W. Keller. Energy management for commercial servers. *Computer*, 2003.

[28] C. Lefurgy, X. Wang, and M. Ware. Server-level power control. In *ICAC*, 2007.

[29] Y.-H. Lu, L. Benini, and G. De Micheli. Operating-system directed power reduction. In *ISLPED*, 2000.

[30] Z. Lu, J. Hein, M. Humphrey, M. Stan, J. Lach, and K. Skadron. Control-theoretic dynamic frequency and voltage scaling for multimedia workloads. In *CASES*, 2002.

[31] R. J. Minerick, V. Freech, and P. M. Kogge. Dynamic power management using feedback. In *Workshop on COmpilers and Operating Systems for Low Power (COLP)*, 2002.

[32] V. Pandey, W. Jiang, Y. Zhou, and R. Bianchini. Dma-aware memory energy management. In *HPCA*, 2006.

[33] P. Ranganathan, P. Leech, D. Irwin, J. Chase, and H. Packard. Ensemble-level power management for dense blade servers. In *ISCA*, 2006.

[34] J. Rao and C.-Z. Xu. Cosl: A coordinated statistical learning approach to measuring the capacity of multi-tier websites. In *IPDPS*, 2008.

[35] J. Rao and C.-Z. Xu. Online measurement of the capacity of multi-tier websites using hardware performance counters. In *ICDCS*, 2008.

[36] V. Sharma, A. Thomas, T. Abdelzaher, and K. Skadron. Power-aware qos management in web servers. In *RTSS*, 2003.

[37] D. C. Snowdon, G. van der Linden, S. M. Petters, and G. Heiser. Accurate run-time prediction of performance degradation under frequency scaling. In *OSPERT*, 2007.

[38] Y. Wang and I. H. Witten. Pace regression. *Hamilton, New Zealand: University of Waikato, Department of Computer Science, 1999*.

[39] Z. Wang, C. McCarthy, X. Zhu, P. Ranganathan, and V. Talwar. Feedback control algorithms for power management of servers. In *FeBID*, 2009.

[40] A. Weissel and F. Bellosa. Process cruise control: event-driven clock scaling for dynamic power management. In *CASES*, 2002.

[41] C.-Z. Xu, B. Liu, and J. Wei. Model predictive feedback control for qos assurance in webservers. *Computer*, 2008.

[42] H. Zeng, C. S. Ellis, and A. R. Lebeck. Experiences in managing energy with ecosystem. *IEEE Pervasive Computing*, 2005.

[43] H. Zeng, C. S. Ellis, A. R. Lebeck, and A. Vahdat. Ecosystem: Managing energy as a first class operating system resource. In *ASPLOS*, 2002.

[44] X. Zhong and C.-Z. Xu. System-wide energy minimization for real-time tasks: Lower bound and approximation. In *ICCAD*, 2006.

[45] X. Zhong and C.-Z. Xu. Frequency-aware energy optimization for real-time periodic and aperiodic tasks. *ACM SIGPLAN Not.*, 2007.

[46] Q. Zhu, F. M. David, C. F. Devaraj, Z. Li, Y. Zhou, and P. Cao. Reducing energy consumption of disk storage using power-aware cache management. In *HPCA*, 2004.

[47] Y. Zhu and F. Mueller. Feedback edf scheduling exploiting dynamic voltage scaling. In *RTAS*, 2004.