SHIP: Scalable Hierarchical Power Control for Large-Scale Data Centers*

Xiaorui Wang, Ming Chen
University of Tennessee, Knoxville, TN 37996
{*xwang, mchen11*}*@eecs.utk.edu*

Charles Lefurgy, Tom W. Keller
IBM Research, Austin, TX 78758
{*lefurgy, tkeller*}*@us.ibm.com*

Abstract

In today's data centers, precisely controlling server power consumption is an essential way to avoid system failures caused by power capacity overload or overheating due to increasingly high server density. While various power control strategies have been recently proposed, existing solutions are not scalable to control the power consumption of an entire large-scale data center, because these solutions are designed only for a single server or a rack enclosure. In a modern data center, however, power control needs to be enforced at three levels: rack enclosure, power distribution unit, and the entire data center, due to the physical and contractual power limits at each level. This paper presents SHIP, a highly scalable hierarchical power control architecture for large-scale data centers. SHIP is designed based on well-established control theory for analytical assurance of control accuracy and system stability. Empirical results on a physical testbed show that our control solution can provide precise power control, as well as power differentiations for optimized system performance. In addition, our extensive simulation results based on a real trace file demonstrate the efficacy of our control solution in large-scale data centers composed of thousands of servers.

1 Introduction

Power consumed by computer servers has become a serious concern in the design of large-scale enterprise data centers. In addition to high electricity bills and negative environmental implications, increased power consumption may lead to system failures caused by power capacity overload or system overheating, as data centers increasingly deploy new high-density servers (*e.g.,* blade servers), while their power distribution and cooling systems have already approached the peak capacities. The goal of power control (also called power capping) is to have runtime measurement and control of the power consumed by servers, so that we can achieve the highest system performance while keeping the power consumption lower than a given power budget, which can be determined by various factors such as the capacity of the power distribution system. Precise power control, combined with power differentiation based on server performance needs, can prevent system failures while allowing data centers to operate at peak efficiencies for a higher return on investment.

In today's data centers, power needs to be controlled at three levels: rack enclosure, Power Distribution Unit (PDU), and an entire data center, due to the physical and contractual power limits at each level [1]. For example, if the physical power limits are violated, overloading of electrical circuits may cause circuit breakers to trip, resulting in undesired outages. Even though data centers commonly rely on power provisioning, the actual power consumption of the IT equipment in a data center may still exceed the power distribution capacity of the facility. A real scenario that many data centers face is that business needs require deploying new servers rapidly while upgrades of the power and cooling systems lag far behind. In some geographies, it is either impossible or cost-prohibitive to provide more power from the utility company to the data centers. For example, the power consumption of National Security Agency (NSA) headquarters in 2006, which is greater than that of the city of Annapolis, reached the power limit of the facility [2]. The agency responded by turning off non-critical equipment. In 2007, the power constraint delayed deployment of new computing equipment and caused planned outages and rolling brownouts in the NSA data center. Similar incidents are expected to increasingly occur in the coming years as more data centers reach their power limits. Therefore, it is important to control the power consumption of an entire data center.

However, to date, most existing work on server power control focuses exclusively on controlling the power consumption of a single server. Only a few recently proposed control strategies are designed for the rack enclosure level [3, 4, 5]. These centralized solutions cannot be easily extended to control an entire large-scale data center due to several reasons. First, the worst-case computational complexity of a centralized controller is commonly proportional to the system size and thus cannot scale well for large-scale systems [6]. Second, since every server in the data center may need to communicate with the centralized controller in every control period, the controller may become a communication bottleneck. Furthermore, a centralized controller may have long communication delays in large-scale systems. Therefore, highly scalable control solutions need to be developed.

In addition, most existing power control solutions heavily rely on heuristics for decision making. In recent years, feed-

[*] The authors at the University of Tennessee are supported by NSF under a CSR grant CNS-0720663 and an NSF CAREER Award CNS-0845390, and by Microsoft Research under a power-aware computing award in 2008.

back control theory has been identified as an effective tool for power control due to its theoretically guaranteed control accuracy and system stability. Control theory also provides well-established controller design approaches, *e.g.,* standard ways to choose the right control parameters, such that exhaustive iterations of tuning and testing can be avoided. Furthermore, control theory can be applied to quantitatively analyze control performance (*e.g.,* stability, settling time) even when the system is suffering unpredictable workload variations. This rigorous design methodology is in sharp contrast to heuristic-based adaptive solutions that heavily rely on extensive manual tuning. For example, recent work [7, 4] has shown that control-theoretic power management outperforms commonly used heuristic solutions by having more accurate power control and better application performance.

There are several challenges in developing scalable power control algorithms. First, the global control problem (*i.e.,* power control for an entire data center) needs to be decomposed into a set of control subproblems for scalability. The decomposition strategy must comply with the data centers' power distribution hierarchy. Second, the local controller designed for each decomposed subproblem needs to achieve local stability and control accuracy despite significantly varying workloads. Third, each local controller needs to coordinate with other controllers at different levels for global stability and control accuracy. Finally, the system performance of the data center needs to be optimized based on optimal control theory, subject to various system constraints.

In this paper, we present SHIP, a highly scalable hierarchical power control architecture for large-scale data centers composed of thousands of servers. Our control architecture is designed based on control theory for theoretically guaranteed control accuracy and system stability. Specifically, the contributions of this paper are three-fold:

- We decompose the problem of power control for a data center into control subproblems at the three levels of the common power distribution hierarchy, and then model the power consumption of each level.
- We design and analyze Multi-Input-Multi-Output (MIMO) power control algorithms for the three levels based on Model Predictive Control (MPC) theory to optimize system performance, while controlling the total power to stay within the desired constraints.
- We present empirical results on a physical testbed to demonstrate that our solution can provide precise power control and desired power differentiation for optimized system performance. We also present simulation results based on a real trace file to show the effectiveness of our solution in large-scale data centers.

The rest of the paper is organized as follows. Section 2 introduces the overall architecture of our hierarchical power control solution. Section 3 describes the system modeling, controller design and analysis of the PDU-level power controller. Section 4 discusses the coordination among controllers at different levels. Section 5 provides the implementation details of our control architecture. Section 6 presents our empirical results on a physical testbed and simulation results. Section 7 highlights the distinction of our work by discussing the related work. Section 8 concludes the paper.

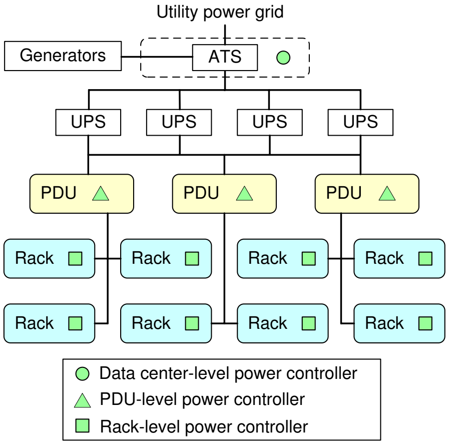

Utility power grid
Generators
ATS
UPS
UPS
UPS
UPS
PDU
PDU
PDU
Rack
Rack
Rack
Rack
Rack
Rack
Rack
Rack
Data center-level power controller
PDU-level power controller
Rack-level power controller

Figure 1. Simplified power distribution hierarchy in a typical Tier-2 data center.

2 Hierarchical Power Control Architecture

In this section, we first introduce the power distribution hierarchy used in many data centers. We then introduce the design of our hierarchical power control architecture.

2.1 Power Distribution Hierarchy

Today's data centers commonly have a three-level power distribution hierarchy to support hosted computer servers [1], though the exact distribution architecture may vary from site to site. Figure 1 shows a simplified illustration of the three-level hierarchy in a typical 1 MW data center. Power from the utility grid is fed to an Automatic Transfer Switch (ATS). The ATS connects to both the utility power grid and on-site power generators. From there, power is supplied to Uninterruptible Power Supplies (UPS) via multiple independent routes for fault tolerance. Each UPS supplies a series of Power Distribution Units (PDUs), which are rated on the order of 75 - 200 kW each. The PDUs further transform the voltage to support a group of server racks.

A typical data center may house ten or more PDUs. Each PDU can support approximately 20 to 60 racks while each rack can include about 10 to 80 computer servers. We assume that the power limit of the upper level (*e.g.,* the data center) is lower than the sum of the maximum power limits of all the lower-level units (*e.g.,* PDUs). This assumption is reasonable because many data centers are rapidly increasing their number of hosted servers to support new business in the short-term while infrastructure upgrades happen over much longer time scales.

2.2 Control Architecture of SHIP

In this section, we provide a high-level description of the SHIP power control architecture, which features a three-level power control solution, as shown in Figure 1. First, the rack-level power controller adaptively manages the power con-

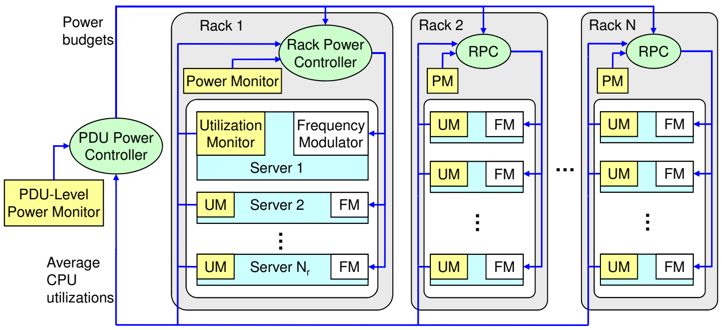


Figure 2. PDU-level and rack-level power control loops in the SHIP power control architecture.

sumption of a rack by manipulating the *CPU frequency* (e.g., via Dynamic Voltage and Frequency Scaling (DVFS)) of the processors of each server in the rack. Second, the PDU-level power controller manages the total power consumption of a PDU by manipulating the *power budget* of each rack in the PDU. Similar to the PDU-level controller, the data center-level controller manages the total power consumption of the entire data center by manipulating the *power budget* of each PDU. Our control architecture is directly applicable to data centers where applications (*e.g.,* scientific computing and background data processing) can allow degraded performance when power must be controlled to stay below a budget at runtime (*e.g.,* due to thermal emergency). For data centers where applications need to achieve specified service-level agreements (SLAs) (*e.g.,* response time), our solution can be integrated with application-level performance control solutions (*e.g.,* [8][9][10]) for simultaneous control of power and application performance.

There are several reasons for us to use processor frequency (and voltage) scaling as our actuation method at the rack level. First, processors commonly contribute a large portion of the total power consumption of a server [11]. As a result, the processor power difference between the highest and lowest power states is large enough to compensate for the power variation of other components and can thus provide an effective way for server power control. Second, frequency scaling has small overhead while some other actuation methods like turning on/off servers may lead to service interruption and undesired long delays. Finally, most today's processors support frequency scaling by DVFS or clock modulation [7], while there are still very few real disks or memory devices that are designed for servers and allow runtime transition among different *active* power modes. We plan to include other actuation methods in our future work.

As shown in Figure 2, the key components in a rack-level control loop include a *power controller* and a *power monitor* at the rack level, as well as a *CPU utilization monitor* and a *CPU frequency modulator* on each server. The control loop is invoked periodically and its period is chosen based on a trade-off between actuation overhead and system settling time. The following steps are invoked at the end of every control period: 1) The power monitor (*e.g.,* a power meter) measures the average value of the total power consumption of all the servers in the last control period and sends the value to the controller. The total power consumption is the *controlled variable* of the control loop. 2) The utilization monitor on each server sends its CPU utilization in the last control period to the controller. The utilization values can be used by the controller to optimize system performance by allowing servers with higher utilizations to run at higher CPU frequencies. Please note that application-level performance metrics such as response time and throughput can also be used in place of CPU utilization to optimize power allocation in our solution. 3) The controller computes the new CPU frequency level for the processors of each server, and then sends the level to the CPU frequency modulator on each server. The levels are the *manipulated variables* of the control loop. 4) The CPU frequency modulator on each server changes the CPU frequency (and voltage if using DVFS) of the processors accordingly. The rack-level power controller is designed based on the power control algorithm presented in [4]. The focus of this paper is on the power control loops at the PDU and data center levels and the coordination among controllers at different levels.

The key components in a PDU-level power control loop include a *power controller* and a *power monitor* at the PDU level, as well as the rack-level power controllers and the utilization monitors of all the racks located within the PDU. The control loop is invoked periodically to change the power budgets of the rack-level control loops of all the racks in the PDU. Therefore, to minimize the impact on the stability of a rack-level control loop, the control period of the PDU-level loop is selected to be longer than the settling time of the rack-level control loop. This guarantees that the rack-level control loop can always enter its steady state within one control period of the PDU-level loop, so that the two control loops are decoupled and can be designed independently. The following steps are invoked at the end of every control period of the PDU-level loop: 1) The PDU-level power controller receives the power consumption of the entire PDU in the last control period from the PDU-level power monitor. The power consumption is the *controlled variable* of this control loop. 2) The PDU-level controller also receives the average CPU utilization of each rack from the rack-level utilization monitor. The utilizations are used to optimize system performance by allocating higher power budgets to racks with higher utilizations. 3) The PDU-level controller then computes the power budget for each rack to have in the next control period based on control theory. The power budgets are the *manipulated variables* of the control loop. 4) The power budget of each rack is then sent to the rack-level power controller of that rack. Since the rack-level power controller is in its steady state at the end of each control period of the PDU-level controller, the desired power budget of each rack can be achieved by the rack-level controller by the end of the next control period of the PDU-level controller.

Similar to the PDU-level control loop, the data center-level power control loop controls the power consumption of the *entire* data center by manipulating the power budgets of the PDU-level power control loops of all the PDUs in the data center. The control period of the data center-level power control loop is selected in the same way to be longer than the settling time of each PDU-level control loop.

3 PDU-level Power Controller

In this section, we introduce the design and analysis of the PDU-level power controller. The data center-level controller is designed in the same way.

3.1 Problem Formulation

PDU-level power control can be formulated as a dynamic optimization problem. In this section, we analytically model the power consumption of a PDU. We first introduce the following notation. T_p is the control period. $pr_i(k)$ is the power consumption of Rack i in the k^{th} control period. $\Delta pr_i(k)$ is the power consumption change of Rack i, *i.e.,* $\Delta pr_i(k) = pr_i(k+1) - pr_i(k)$. $br_i(k)$ is the power budget of Rack i in the k^{th} control period. $\Delta br_i(k)$ is the power budget change of Rack i, *i.e.,* $\Delta br_i(k) = br_i(k+1) - br_i(k)$. $ur_i(k)$ is the average CPU utilization of all the servers in Rack i in the k^{th} control period. N is the total number of racks in the PDU. $pp(k)$ is the aggregated power consumption of the PDU. P_s is the power set point, *i.e.,* the desired power constraint of the PDU.

Given a control error, $pp(k) - P_s$, the control goal at the k^{th} control point (*i.e.,* time kT_p) is to dynamically choose a power budget change vector $\mathbf{\Delta br(k)} = [\Delta br_1(k) \ldots \Delta br_N(k)]^T$ to minimize the difference between the power consumption of the PDU in the next control period and the desired power set point:

$$\min_{\{\Delta br_j(k)|1 \leq j \leq N\}} (pp(k+1) - P_s)^2 \tag{1}$$

This optimization problem is subject to three constraints. First, the power budget of each rack should be within an allowed range, which is estimated based on the number of servers in that rack and the maximum and minimum possible power consumption of each server. This constraint is to prevent the controller from allocating a power budget that is infeasible for the rack-level power controller to achieve. Second, power differentiation can be enforced for two or more racks. For example, in some commercial data centers that host server racks for different clients, racks may have different priorities for power budget allocation. As power is directly related to application performance, the power budget allocated to one rack may be required to be n (*e.g.,* 1.2) times that allocated to another rack. This is referred to as *proportional power differentiation*. The differentiation is particularly important when the entire data center is experiencing temporary power budget reduction. In that case, with power differentiation, premium clients may have just slightly worse application performance while ordinary clients may suffer significant performance degradation. Finally, the total power consumption should not be higher than the desired power constraint. The three constraints are modeled as:

$$P_{min,j} \leq \Delta br_j(k) + br_j(k) \leq P_{max,j} \ (1 \leq j \leq N)$$
$$\Delta br_i(k) + br_i(k) = n(\Delta br_j(k) + br_j(k)) \ (1 \leq i \neq j \leq N)$$
$$pp(k+1) \leq P_s$$

where $P_{min,j}$ and $P_{max,j}$ are the *estimated* minimum and maximum power consumption of a rack. The two values are estimated based on the number of servers in the rack and the estimated maximum and minimum power consumption of a server when it is running a nominal workload. The two values may be different in a real system due to different server configurations and workloads, which could cause the controller to allocate a power budget that is infeasible (*e.g.,* too high or too low) for a rack-level controller to achieve. This uncertainty is modeled in the system model described in the next subsection. Therefore, PDU-level power management has been formulated as a constrained MIMO optimal control problem.

3.2 System Modeling

We now consider the total power consumption of a PDU. The total power consumption in the $(k+1)^{th}$ control period, $pp(k+1)$, is the result of the power consumption of the PDU in the previous control period, $pp(k)$, plus the sum of the power consumption changes of all the racks in the PDU.

$$pp(k+1) = pp(k) + \sum_{i=1}^{N} \Delta pr_i(k) \tag{2}$$

As introduced in Section 2, the control period of the PDU-level controller is longer than the settling time of the rack-level controller. As a result, at the end of each control period of the PDU-level controller, the desired power budget of each rack should have already been achieved by the corresponding rack-level controller, *i.e.,* the power consumption change $\Delta pr_i(k)$ should be equal to the power budget change $\Delta br_i(k)$. However, there could be situations that a rack may fail to achieve a given power budget because it is infeasible to do so. For example, a rack may fail to reach a given high power budget because its current workload is not as power-intensive as the nominal workload used to estimate the maximum power consumption of a rack used in constraint (2). As a result, the current workload may not be enough for the rack to achieve the given power budget even when all the servers in the rack are running at their highest frequencies. In that case, the power consumption change of the rack may become a function of the change of its assigned budget, *i.e.,* $\Delta pr_i(k) = g_i \Delta br_i(k)$, where g_i is the system gain, which is also called the *power change ratio*. Note that g_i is used to model the uncertainties of the PDU-level power controller and its value is unknown at design time. Our model is not limited to constant g_i. When g_i varies along time, we can identify a range of g_i for which there exists a common Lyapunov function for all g_is. As a result, our model can handle time varying g_i without any change. The literature has discussion for time-varying systems in more detail [6].

In general, the relationship between the power consumption of all the servers in a PDU and the power budget change of each rack in the PDU can be modeled as follows.

$$pp(k+1) = pp(k) + \mathbf{G}\mathbf{\Delta br(k)} \quad (3)$$

where $\mathbf{G} = \left[\begin{array}{ccc} g_1 & \dots & g_N \end{array}\right]$, and $\mathbf{\Delta br(k)} = \left[\begin{array}{ccc} \Delta br_1(k) & \dots & \Delta br_N(k) \end{array}\right]^T$.

3.3 Controller Design and Analysis

We apply *Model Predictive Control* (MPC) theory [12] to design the controller. MPC is an advanced control technique that can deal with MIMO control problems with constraints on the plant and the actuators. This characteristic makes MPC well suited for power control in data centers.

A model predictive controller optimizes a *cost function* defined over a time interval in the future. The controller uses a system model to predict the control behavior over P control periods, called the *prediction horizon*. The control objective is to select an input trajectory that minimizes the cost function while satisfying the constraints. An input trajectory includes the control inputs in the following M control periods, $\mathbf{\Delta br(k)}$, $\mathbf{\Delta br(k+1|k)}$, $\dots$ $\mathbf{\Delta br(k+M-1|k)}$, where M is called the *control horizon*. The notation $x(k+i|k)$ means that the value of variable x at time $(k+i)T_p$ depends on the conditions at time kT_p. Once the input trajectory is computed, only the first element $\mathbf{\Delta br(k)}$ is applied as the control input to the system. At the end of the next control period, the prediction horizon slides one control period and the input is computed again based on the feedback $pp(k)$ from the power monitor. Note that it is important to re-compute the control input because the original prediction may be incorrect due to uncertainties and inaccuracies in the system model used by the controller. MPC combines performance prediction, optimization, constraint satisfaction, and feedback control into a single algorithm.

The MPC controller includes a least squares solver, a cost function, a reference trajectory, and a system model. At the end of every control period, the controller computes the control input $\mathbf{\Delta br(k)}$ that minimizes the following cost function under constraints.

$$\begin{aligned} V(k) = \; & \sum_{i=1}^{P} \|pp(k+i|k) - ref(k+i|k)\|^2_{Q(i)} \\ & + \sum_{i=0}^{M-1} \|\mathbf{\Delta br(k+i|k)} + \mathbf{br(k+i|k)} - \mathbf{P_{max}}\|^2_{\mathbf{R(i)}} \end{aligned} \quad (4)$$

where P is the prediction horizon, and M is the control horizon. $Q(i)$ is the *tracking error weight*, and $\mathbf{R(i)}$ is the *control penalty weight vector*. The first term in the cost function represents the *tracking error*, *i.e.,* the difference between the total power $pp(k+i|k)$ and a reference trajectory $ref(k+i|k)$. The reference trajectory defines an ideal trajectory along which the total power $pp(k+i|k)$ should change from the current value $pp(k)$ to the set point P_s (*i.e.,* power budget of the PDU). Our controller is designed to track the following exponential reference trajectory so that the closed-loop system behaves like a linear system.

$$ref(k+i|k) = P_s - e^{-\frac{T_p}{T_{ref}}i}(P_s - pp(k)) \quad (5)$$

where T_{ref} is the time constant that specifies the speed of system response. A smaller T_{ref} causes the system to converge faster to the set point but may lead to larger overshoot. By minimizing the tracking error, the closed-loop system will converge to the power set point P_s if the system is stable.

The second term in the cost function (4) represents the *control penalty*. The control penalty term causes the controller to optimize system performance by minimizing the difference between the estimated maximum power consumptions, $\mathbf{P_{max}} = [P_{max,1} \dots P_{max,N}]^T$ and the new power budgets, $\mathbf{br(k+i+1|k)} = \mathbf{\Delta br(k+i|k)} + \mathbf{br(k+i|k)}$ along the control horizon. The control weight vector, $\mathbf{R(i)}$, can be tuned to represent preference among servers. For example, a higher weight may be assigned to a rack if it has heavier or more important workloads, so that the controller can give preference to increasing its power budget. As a result, the overall system performance can be optimized. In our experiments, we use the average CPU utilization of all the servers in each rack as an example weight to optimize system performance.

We have established a system model (3) for the PDU-level power consumption in Section 3.2. However, the model cannot be directly used by the controller because the system gains $\mathbf{G}$ are unknown at design time. In our controller design, we assume that $g_i = 1, (1 < i < N)$, *i.e.,* all the racks can achieve their desired power budget changes in the next control period. Hence, our controller solves the constrained optimization based on the following *estimated* system model:

$$pp(k+1) = pp(k) + [1 \dots 1]\, \mathbf{\Delta br(k)}. \quad (6)$$

In a real system that has different server configurations or is running a different workload, the *actual* value of g_i may become different than 1. As a result, the closed-loop system may behave differently. A fundamental benefit of the control-theoretic approach is that it gives us theoretical confidence for system stability, even when the estimated system model (6) may change for different workloads. In MPC, we say that a system is *stable* if the total power $pp(k)$ converges to the desired set point P_s, that is, $\lim_{k\to\infty} pp\,(k) = P_s$. In an extended version of this paper [13], we prove that a system controlled by the controller designed with the assumption $g_i = 1$ can remain stable as long as the actual system gain $0 < g_i < 14.8$. This range is established using stability analysis of the closed-loop system by considering the model variations. To handle systems with an actual g_i that is outside the established stability range, an online model estimator implemented in our previous work [14] can be adopted to dynamically correct the system model based on the real power measurements, such that the system stability can be guaranteed despite significant model variations.

This control problem is subject to the three constraints introduced in Section 3.1. The controller must minimize the cost function (4) under the three constraints. This constrained optimization problem can be transformed to a standard constrained least-squares problem. The transformation is similar to that in [15] and not shown due to space limitations. The controller can then use a standard least-squares solver to solve the optimization problem on-line. In our system, we implement the controller based on the `lsqlin` solver in Matlab. `lsqlin` uses an active set method similar to that described in [16]. The computational complexity of `lsqlin` is polynomial in the number of racks in the PDU and the control and prediction horizons. The overhead measurement of `lsqlin` can be found in [15].

4 Coordination with Rack-level Controller

As discussed in Section 2, to achieve global stability, the period of an upper-level (*e.g.,* PDU) control loop is preferred to be longer than the settling time of a lower-level (*e.g.,* rack) control loop. This guarantees that the lower-level loop can always enter its steady state within one control period of the upper-level control loop, so that the two control loops are decoupled and can be designed independently. As long as the two controllers are stable individually, the combined system is stable. Note that the configuration of settling time is a sufficient but *not* necessary condition for achieving global stability. In other words, global stability can be achieved in some cases even when the control period is shorter than the settling time of the lower-level control loop [17].

We now analyze the settling times of the PDU-level control loop and the rack-level control loop. The settling time analysis includes three general steps. First, we compute the feedback and feedforward matrices for the controller by solving the control input based on the system model (*e.g.,* (3)) of a specific system and the reference trajectory (*e.g.,* (5)). The analysis needs to consider the composite system consisting of the dynamics of the original system and the controller. Second, we derive the closed-loop model of the composite system by substituting the control inputs derived in the first step into the actual system model. Finally, we calculate the dominant pole (*i.e.,* the pole with the largest magnitude) of the closed-loop system. According to control theory, the dominant pole determines the system's transient response such as settling time.

As an example, we follow the above steps to analyze the settling times of the PDU-level controller and a rack-level controller used in our experiments. The PDU-level controller has a nominal gain vector $\mathbf{G} = [1, 1, 1]$. Our results show that the magnitude of the dominant pole of of the closed-loop system is 0.479. As a result, the number of control periods for the PDU-level loop to settle is 6. Similarly, the number of control periods for the rack-level loop to settle is 16.

Note that the selection of control periods is also related to the sampling period of the adopted power monitor. Since the shortest period for the power meters used in our testbed to sample power is 1 second, the control period of the rack-level control loops is set to 5 seconds to eliminate instantaneous reading errors by having averaged values. According to the settling time analysis, the control period of the PDU-level control loop is set to 80 seconds in our experiments. It is important to note that our control loops are not restricted to such long control periods. When equipped with high-precision power monitors that can sample power in a significantly shorter period (*e.g.,* 1 millisecond [7]), our control solution can quickly react to sudden power budget violations caused by unexpected demand spikes at different levels. In a real data center where such power monitors are equipped, the control periods should be derived to ensure that the settling times of the controllers are shorter than the manufacturer-specified time interval for the power supplies to sustain a power overload [7].

5 System Implementation

In this section, we first introduce the physical testbed and benchmark used in our experiments, as well as the implementation details of the control components. We then introduce our simulation environment.

5.1 Testbed

Our testbed includes 9 Linux servers to run workloads and a Linux machine to run the controllers. The 9 servers are divided into 3 groups with 3 servers in each group. Each group emulates a rack while the whole testbed emulates a PDU. Server 1 to Server 4 are equipped with 2.4GHz AMD Athlon 64 3800+ processors and run openSUSE 11.0 with kernel 2.6.25. Server 5 to Server 8 are equipped with 2.2GHz AMD Athlon 64 X2 4200+ processors and run openSUSE 10.3 with kernel 2.6.22. Server 9 is equipped with 2.3GHz AMD Athlon 64 X2 4400+ processors and runs openSUSE 10.3. All the servers have 1GB RAM and 512KB L2 cache. Rack 1 includes Server 1 to Server 3. Rack 2 includes Server 4 to Server 6. Rack 3 includes Server 7 to Server 9. The controller machine is equipped with 3.00GHz Intel Xeon Processor 5160 and 8GB RAM, and runs openSUSE 10.3. All the machines are connected via an internal Ethernet switch.

In our experiments on the testbed, we run the High Performance Computing Linpack Benchmark (HPL) (V1.0a) on each server as our workload. HPL is a software package that solves a (random) dense linear system in double precision (64 bits) arithmetic. The problem size of HPL is configured to be $10,000 \times 10,000$ and the block size is set as 64 in all experiments unless otherwise noted. We use HPL as our workload because it provides a standard way to quantify the performance improvement achieved by our control solution. We have tested our control architecture using other commercial benchmarks. The results are similar and can be found in the extended version [13].

We now introduce the implementation details of each component in our power control architecture.

Power Monitor. The power consumptions of the emulated PDU and three racks are measured with 4 WattsUp Pro power meters, which have an accuracy of 1.5% of the measured value. The power meters sample the power data every second and then send the readings to the 4 controllers through system files */dev/ttyUSB0* to *ttyUSB3*.

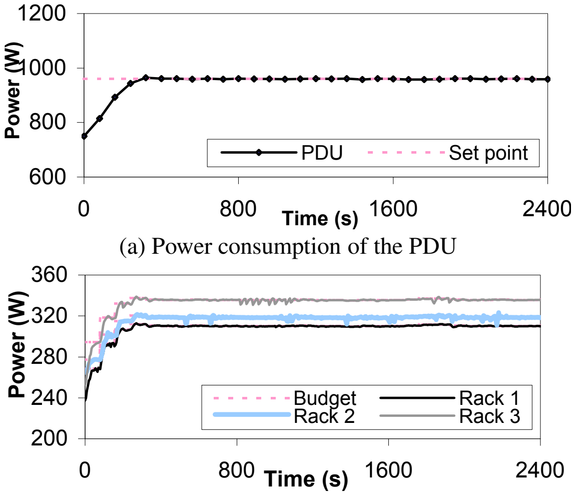
Utilization Monitor. The utilization monitor uses the */proc/stat* file in Linux to estimate the CPU utilization in each control period. The file records the number of jiffies (usually 10ms in Linux) when the CPU is in user mode, user mode with low priority (nice), system mode, and when used by the idle task, since the system starts. At the end of each sampling period, the utilization monitor reads the counters, and estimates the CPU utilization as 1 minus the number of jiffies used by the idle task divided by the total number of jiffies in the last control period.

Power Controllers. Each rack-level power controller receives the power reading of the corresponding rack and the CPU utilizations of all the servers in the rack in the last control period. The controller then executes the control algorithm presented in [4] to compute new CPU frequency levels for the servers. The new levels are then sent to the CPU frequency modulators on the servers. The PDU-level controller receives the power measurement of the PDU and the average CPU utilizations of all the racks in the PDU in the last control period. The PDU controller then executes the control algorithm presented in Section 3 to compute new power budgets for the racks. The budgets are then sent to the rack-level controllers. The MPC controller parameters used in the experiments include the prediction horizon as 8 and the control horizon as 2. The time constant T_{ref}/T_p used in (5) is set as 2 to avoid overshoot while having a relatively short settling time.

CPU Frequency Modulator. We use AMD's Cool'n'Quiet technology to enforce the new frequency (and voltage) level by DVFS. The AMD microprocessors have 4 or 5 discrete DVFS levels. To change CPU frequency, one needs to install the *cpufreq* package and then use the root privilege to write the new frequency level into the system file */sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed*. Since the new CPU frequency level received from the rack-level power controller is normally a fractional value, the modulator code must resolve this to a series of discrete frequency values to approximate the fractional value. For example, to approximate 3.2 during a control period, the modulator would output the sequence 3, 3, 3, 3, 4, 3, 3, 3, 3, 4, etc on a smaller time scale. The first-order delta-sigma modulator introduced in [7] is used to implement this. The actuation overhead is analyzed in [4]

5.2 Simulation Environment

To stress test the hierarchical control architecture in large-scale data centers, we have developed a C++ simulator that uses a trace file from real-world data centers to simulate the CPU utilization variations. The trace file includes the utilization data of $5,415$ servers from ten large companies covering manufacturing, telecommunications, financial, and retail sectors. The trace file records the average CPU utilization of each server in every 15 minutes from 00:00 on July 14th (Monday) to 23:45 on July 20th (Sunday) in 2008. We generate several data center configurations. In each configuration, we group the servers into 6 to 8 PDUs with each PDU including 20 to 60 racks and each rack including 10 to 30 servers. Based on the specifications of the real servers used in our testbed, each server is randomly configured to have a minimum power consumption between 90W and 110W, a maximum power consumption between 150W and 170W, and a lowest relative frequency level between 0.3 and 0.5. More details of the simualtor can be found in [13].



(a) Power consumption of the PDU

(b) Power consumptions of the three racks

Figure 3. A typical run of the SHIP hierarchical control solution on the physical testbed.

6 Experimentation

In this section, we first present our empirical results on the testbed. We then describe our simulation results in large-scale data centers based on the real trace file.

6.1 Empirical Results

We first demonstrate that the SHIP hierarchical control solution can provide precise power control for different power set points. We then examine the capability of SHIP to provide desired power differentiation.

6.1.1 Precise Power Control

In this experiment, we run the HPL benchmark on each of the 9 servers. The power set point of the PDU is 960W. Figure 3 shows a typical run of the SHIP hierarchical control solution. At the beginning of the run, the total power of the PDU is lower than the set point because all the servers are initially running at the lowest frequency levels. The PDU-level controller responds by giving more power budgets to all the three racks. The rack-level controllers then step up the servers' frequency levels to achieve the new power budgets within one control period of the PDU-level loop. After four control periods, the power consumption of the PDU has been precisely controlled at the desired set point, without causing an undesired overshoot. After the transient state, as shown in Figure 3(b), the power budget allocated to each rack is kept at a stable value with only minor variations. The power consumption of each rack has also been precisely controlled at their respective allocated budgets. As discussed in Section 3.3, the PDU controller tries to minimize the difference

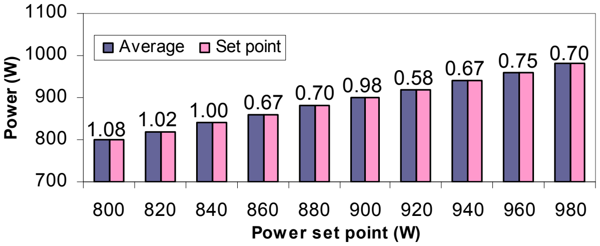


Figure 4. Average power consumption of the emulated PDU under different power set points (with standard deviations above the bars).

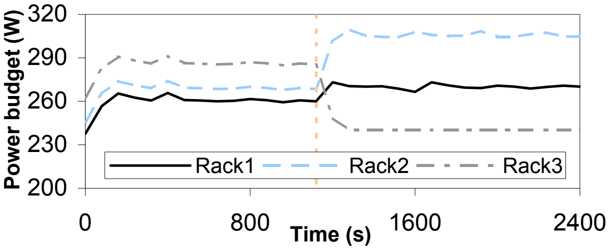


Figure 5. Power differentiation based on performance needs. Rack 3 has the lowest utilization.

between the estimated maximum power consumption (*i.e.,* $P_{max,j}$) and the allocated power budget for each rack in its cost function. Specifically, the maximum power consumption for Racks 1 to 3 is 339W, 347.5W, and 364.5W, respectively. Since all the racks have the same weight (100% CPU utilization), their budgets are allocated to have the same distance with their maximum power consumptions.

In a data center, a PDU may be given different power set points at different times. For example, a data center may need to deploy a new PDU before an upgrade of its power distribution capacity can be done. As a result, the power set points of all other PDUs need to be reduced to accommodate the new PDU. Therefore, it is important to precisely control power for different power set points. We test our control solution for different set points (from 800W to 980W). Figure 4 plots the average power consumption of the emulated PDU with the standard deviation on the top of each bar. Each value is the average of 20 power measurements of the PDU after the PDU-level controller enters its steady state. The maximum standard deviation is only 1.08W around the desired set point. This experiment demonstrates that SHIP can provide precise power control.

6.1.2 Power Differentiation

In data centers, it is commonly important for servers to have differentiated power budgets, especially when the available power resource is not enough for all the servers to run at their highest CPU frequency levels. For example, higher budgets can be given to servers running heavier workloads for improved overall system performance.

In this experiment, we differentiate server racks by giving higher weights (*i.e.,* $\mathbf{R}(\mathbf{i})$) in the controller's cost function (4) to racks that have heavier workloads. Specifically, the weights are assigned proportionally to the racks' average CPU utilizations. Since running HPL on a server always leads to a 100% CPU utilization, we slightly modify the original HPL workload by inserting a sleep function at the end of each iteration in its computation loop, such that we can achieve different utilizations such as 80%, 50% for different servers. In the modified version of HPL, the problem size is configured to be $4,000 \times 4,000$ and the block size is set as 1. Note that the modified HPL benchmark is used *only* in this experiment. The power set point of the PDU is set to 810W. At the beginning of the run, we use the original HPL on all the servers so that all the racks have an average CPU utilization of 100%. As a result, all the racks are given the same weight. At time 1120s, we dynamically change the workload only on the servers in Racks 1 and 3 to run the modified HPL, so that the average CPU utilizations of Racks 1 and 3 become approximately 80% and 50%, respectively. Figure 5 shows that the controller responds to the workload variations by giving a higher power budget to a rack with a higher average CPU utilization. Rack 3 has the lowest budget because it has the lowest average utilization (*i.e.,* 50%). Note that application-level performance metrics such as response time and throughput can also be used to optimize power allocation in our solution. The results demonstrate that SHIP can provide power differentiation for the consideration of overall system performance.

6.2 Simulation Results in Large-Scale Data Centers

In this section, we test SHIP in large-scale data centers using the trace file introduced in Section 5.2, which has the utilization data of $5,415$ servers from real-world data centers.

Figure 6 is a typical run of SHIP in a data center that is generated based on the method introduced in Section 5. This data center has 6 PDUs and 270 racks. The power set point of the data center is 750kW. As shown in Figure 6, the power of the data center precisely converges to the desired set point in two control periods of the data center-level control loop. Figure 7 plots the average power consumptions of three randomly generated data centers under a wide range of power set points from 600kW to 780kW. It is clear that SHIP can achieve the desired set point for the three large-scale data centers. The maximum standard deviation of all the data centers under all the power set points is only 0.72kW.

We then examine the capability of SHIP to differentiate PDUs based on the utilization data from the trace file. According to the controller design in Section 3.3, the data center-level controller tries to minimize the difference between the estimated maximum power consumption and the power budget for each PDU. Therefore, a PDU with a higher average CPU utilization should have a smaller difference because of its higher weight in the controller's cost function. Figure 9(a) shows the average CPU utilizations of the 6 PDUs in the experiment, while Figure 9(b) shows the difference (*i.e.,* the estimated maximum power consumption minus the power budget) for each PDU. We can see that the difference order of the PDUs is consistent with the order of

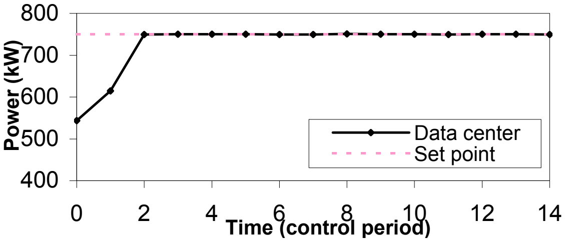


Figure 6. A typical run of SHIP in a simulated large-scale data center.

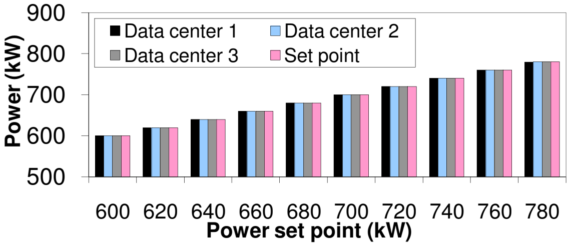


Figure 7. Average power consumptions under different data centers and power set points.

their average CPU utilizations. For example, PDU 2 has the highest average CPU utilization and thus the smallest difference. The results demonstrate that SHIP can effectively achieve the desired control objectives in large-scale data centers.

The key advantage of the SHIP hierarchical control solution is that it decomposes the global control problem into a set of control subproblems at the three levels of the power distribution hierarchy in a data center. As a result, the overhead of each individual controller is bounded by the maximum number of units possibly controlled by a controller, which is 60 (racks in a PDU) in our simulations. Figure 8 shows that the average execution time of a centralized MPC controller (in 3 randomly generated data centers) increases dramatically when the number of directly controlled servers increases. Given that the control period of a data center-level power controller usually should be shorter than several minutes, a centralized MPC controller can only control 500 or fewer servers at the data center level. In addition, a centralized controller normally has undesired long communication delays in large-scale systems, resulting in degraded control performance. Therefore, centralized control solutions are not suitable for large-scale data centers.

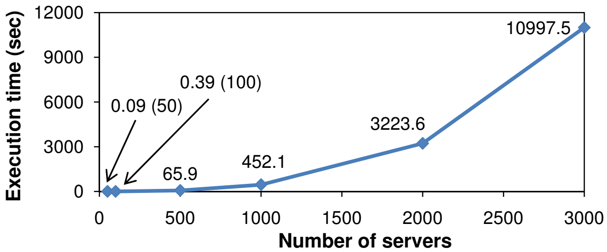


Figure 8. Average execution time of the MPC controller for different numbers of servers.

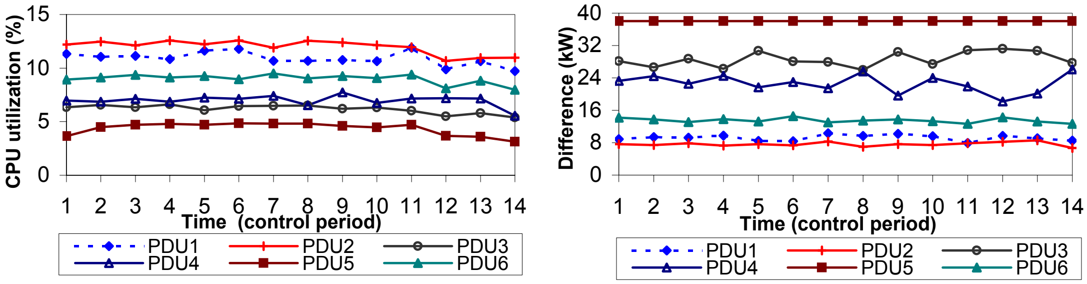
7 Related Work

Power is one of the most important design constraints for enterprise servers. Much of the prior work has attempted to reduce power consumption by improving the energy-efficiency of individual server components [18]. There has been some work on system-level power and thermal management [19, 20, 21]. For example, Nathuji et al. have proposed heuristic solutions for power budgeting in virtualized environments [22]. In contrast to existing work, which relies on heuristic-based control schemes, we adopt a rigorous design methodology that features a *control-theoretic* framework for systematically developing control strategies with analytical assurance of control accuracy and system stability.

Several research projects [23, 7, 24] have successfully applied control theory to explicitly control power or temperature of a single enterprise server. Some recent work has proposed heuristic-based control strategies at the rack level [3, 25]. Control-theoretic solutions have also been designed to control rack-level power consumption for optimized system performance [4]. However, those solutions cannot be directly applied to control a PDU or an entire data center because the overhead of their centralized control schemes becomes prohibitive when the system size increases to a certain extent. In contrast, our hierarchical control architecture is highly scalable for large-scale data centers.

A recent study [5] indicates the possibility of having a general group power manager that can be extended to control a data center. Our work is different in three aspects: 1) our control scheme is designed specifically based on data centers' three-level power supply hierarchy, 2) our solution features a MIMO control strategy with rigorous stability analysis, and 3) our work is evaluated on a physical testbed, while only simulation results are presented in [5]. In addition, we also present simulation results in large-scale data centers with a trace file of $5,415$ servers while only 180 servers are simulated in [5]. At the PDU level, Govindan et al. [26] propose statistical profiling-based techniques to provision servers under a power constraint. At the data center level, Fan et al. [1] investigate the aggregate power usage characteristics of a warehouse-sized data center. In contrast, we dynamically control the power consumption of an entire data center and optimize system performance by shifting power among racks and PDUs.

Some prior work has been proposed to use power as a tool for application-level performance requirements at the OS level. For example, Horvath et al. [8] use dynamic voltage scaling (DVS) to control end-to-end delay in multi-tier web servers. Sharma et al. [27] effectively apply control theory to control application-level quality of service requirements. Chen et al. [9] also present a feedback controller to manage the response time in a server cluster. Although they all use control theory to manage power consumption, power is only used as a knob to control application-level performance. 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 to ad-



(a) Average CPU utilization

(b) Difference between maximum power and budget

Figure 9. Power budget differentiation based on average CPU utilizations.

here to a given constraint. Our solution is complementary to OS-level power management schemes and can be combined for increased adaptation capability and simultaneous control of power and system performance.

8 Conclusions

Power control for an entire data center has become increasingly important. However, existing server power control solutions are not scalable for large-scale data centers because they are designed for a single server or a rack enclosure. In this paper, we presented SHIP, a highly scalable hierarchical control architecture that controls the total power consumption of a large-scale data center to stay within a constraint imposed by its power distribution capacity. The control architecture is designed based on rigorous control theory for analytical assurance of control accuracy and system stability. Empirical results on a physical testbed show that our control solution can provide precise power control, as well as power differentiations for optimized system performance. In addition, our extensive simulation results based on a real trace file demonstrate the efficacy of our control solution in large-scale data centers composed of thousands of servers.

References

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

[2] Siobhan Gorman, “Power supply still a vexation for the NSA,” The Baltimore Sun, June 2007.

[3] P. Ranganathan, P. Leech, D. Irwin, and J. S. Chase, “Ensemble-level power management for dense blade servers.” in *ISCA*, 2006.

[4] X. Wang and M. Chen, “Cluster-level feedback power control for performance optimization,” in *HPCA*, 2008.

[5] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, “No power struggles: Coordinated multi-level power management for the data center,” in *ASPLOS*, 2008.

[6] X. Wang, D. Jia, C. Lu, and X. Koutsoukos, “DEUCON: Decentralized end-to-end utilization control for distributed real-time systems,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 7, 2007.

[7] C. Lefurgy, X. Wang, and M. Ware, “Power capping: a prelude to power shifting,” *Cluster Computing*, vol. 11, no. 2, 2008.

[8] T. Horvath, T. Abdelzaher, K. Skadron, and X. Liu, “Dynamic voltage scaling in multi-tier web servers with end-to-end delay control,” *IEEE Transactions on Computers*, vol. 56, no. 4, pp. 444–458, 2007.

[9] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, “Managing server energy and operational costs in hosting centers,” in *SIGMETRICS*, 2005.

[10] Y. Wang, X. Wang, M. Chen, and X. Zhu, “Power-efficient response time guarantees for virtualized enterprise servers,” in *RTSS*, 2008.

[11] P. Bohrer, E. N. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony, “The case for power management in web servers,” *Power Aware Computing*, 2002.

[12] J. M. Maciejowski, *Predictive Control with Constraints*. Prentice Hall, 2002.

[13] X. Wang, M. Chen, C. Lefurgy, and T. W. Keller, “Hierarchical Power Control for Large-Scale Data Centers, Tech Report, EECS, University of Tennessee,” http://pacs.ece.utk.edu/techreports/tech0918.pdf, 2009.

[14] Y. Wang, K. Ma, and X. Wang, “Temperature-constrained power control for chip multiprocessors with online model estimation,” in *ISCA*, 2009.

[15] C. Lu, X. Wang, and X. Koutsoukos, “Feedback utilization control in distributed real-time systems with end-to-end tasks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 6, 2005.

[16] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*. Academic Press, London, UK, 1981.

[17] X. Fu, X. Wang, and E. Puster, “Dynamic thermal and timeliness guarantees for distributed real-time embedded systems,” in *RTCSA*, 2009.

[18] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. W. Keller, “Energy management for commercial servers,” *IEEE Computer*, vol. 36, no. 12, pp. 39–48, 2003.

[19] H. Zeng, C. S. Ellis, A. R. Lebeck, and A. Vahdat, “ECOSystem: managing energy as a first class operating system resource,” in *ASPLOS*, 2002.

[20] Y.-H. Lu, L. Benini, and G. D. Micheli, “Operating-system directed power reduction,” in *ISLPED*, 2000.

[21] D. Brooks and M. Martonosi, “Dynamic thermal management for high-performance microprocessors.” in *HPCA*, 2001.

[22] R. Nathuji and K. Schwan, “Vpm tokens: virtual machine-aware power budgeting in datacenters,” in *HPDC*, 2008.

[23] R. J. Minerick, V. W. Freeh, and P. M. Kogge, “Dynamic power management using feedback,” in *COLP*, Sep. 2002.

[24] K. Skadron, T. Abdelzaher, and M. R. Stan, “Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management,” in *HPCA*, 2002.

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

[26] S. Govindan, J. Choi, B. Urgaonkar, A. Sivasubramaniam, and A. Baldini, “Statistical profiling-based techniques for effective provisioning of power infrastructure in consolidated data centers,” Pennsylvania State University, Tech. Rep. CSE08-008, 2008.

[27] V. Sharma, A. Thomas, T. Abdelzaher, K. Skadron, and Z. Lu, “Power-aware QoS management in web servers,” in *RTSS*, 2003.