

Power Consumption Prediction and Power-Aware Packing in Consolidated Environments

Jeonghwan Choi, Sriram Govindan, Jinkyu Jeong, Bhuvan Urgaonkar, and Anand Sivasubramaniam, *Senior Member, IEEE*

Abstract—Consolidation of workloads has emerged as a key mechanism to dampen the rapidly growing energy expenditure within enterprise-scale data centers. To gainfully utilize consolidation-based techniques, we must be able to characterize the power consumption of groups of colocated applications. Such characterization is crucial for effective prediction and enforcement of appropriate limits on power consumption—*power budgets*—within the data center. We identify two kinds of power budgets: 1) an *average budget* to capture an upper bound on long-term energy consumption within that level and 2) a *sustained budget* to capture any restrictions on sustained draw of current above a certain threshold. Using a simple measurement infrastructure, we derive *power profiles*—statistical descriptions of the power consumption of applications. Based on insights gained from detailed profiling of several applications—both individual and consolidated—we develop models for predicting average and sustained power consumption of consolidated applications. We conduct an experimental evaluation of our techniques on a Xen-based server that consolidates applications drawn from a diverse pool. For a variety of consolidation scenarios, we are able to predict average power consumption within five percent error margin and sustained power within 10 percent error margin. Using prediction techniques allows us to ensure safe yet efficient system operation—in a representative case, we are able to improve the number of applications consolidated on a server from two to three (compared to existing baseline techniques) by choosing the appropriate power state that satisfies the power budgets associated with the server.

Index Terms—Power-aware systems, reliability, server consolidation.



1 INTRODUCTION AND MOTIVATION

To accommodate modern resource-intensive high-performance applications, large-scale data centers have grown at a rapid pace in a variety of domains ranging from research labs and academic groups to industry. The fast-growing power consumption of these platforms is a major concern due to its implications on the cost and efficiency of these platforms as well as the well-being of our environment. Trends from such platforms suggest that the power consumption in data centers accounts for 1.2 percent of the overall electricity consumption in the US [30]. More alarmingly, if current practices for the design and operation of these platforms continue, their power consumption is projected to keep growing at 18 percent every year. These observations have spurred great interest among providers of data centers to explore ways to dampen the growth rate of servers by doing better *consolidation*. For example, as workload conditions change, it may be desirable to pack

hosted applications on to different subsets of racks/servers within the data center and turn off machines that are not needed [6], [7], [31]. Another major concern for data centers is the increase in power density of the servers which are reaching the limits of the power delivery and cooling infrastructure of these platforms, thereby raising reliability concerns. In response to this, existing research has attempted to reduce the peak power consumption both at the server level [15] as well as at the cluster level [33]. Consolidation further increases the power density of the servers, aggravating the reliability concerns of the facility.

Power budgets—upper bounds on power consumption—are a useful abstraction employed by several recently proposed techniques to address these energy and reliability related concerns in data centers [15], [33], [32]. Such power budgets need to be enforced at different levels of the spatial hierarchy. Previous work has looked at mechanisms to enforce power budgets both at the server level [26] and the cluster level [47]. Typically, power budgets are enforced either by throttling resource usage [47], [26] and/or migrating workloads [27], [43]. In our opinion, these techniques often operate with inadequate information about the power consumption behavior of the hosted applications. As a result, when consolidating applications, such techniques could cause poor performance or less effective enforcement of power budgets. Solutions for consolidation in data centers have benefited from detailed studies of the resource needs of individual applications [46]. Insights gained from these studies have been utilized to build models for predicting the performance and resource usage

- J. Choi is with the School of Information and Communication Engineering, Sungkyunkwan University. E-mail: jechoi@skku.edu.
- S. Govindan, B. Urgaonkar, and A. Sivasubramaniam are with the Department of Computer Science and Engineering, The Pennsylvania State University, University Park. E-mail: {sgovinda, bhuvan, anand}@cse.psu.edu.
- J. Jeong is with the Department of Computer Science, KAIST. E-mail: jinkyu@calab.kaist.ac.kr.

Manuscript received 3 Mar. 2009; revised 30 Nov. 2009; accepted 21 Jan. 2010; published online 14 Apr. 2010.

Recommended for acceptance by V. Barbosa.

For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2009-03-0099. Digital Object Identifier no. 10.1109/TC.2010.91.

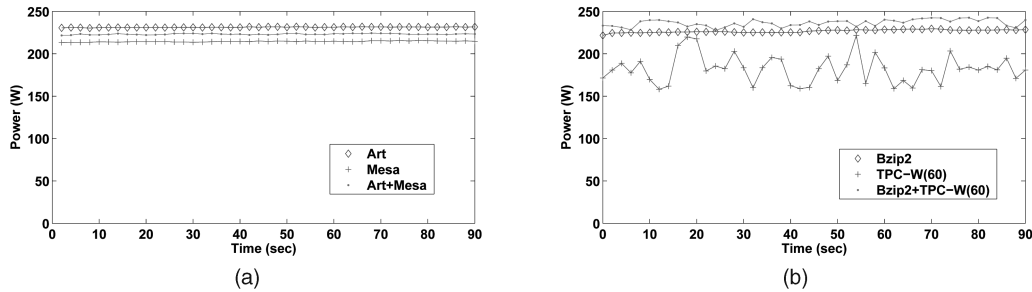


Fig. 1. Measured power consumption behaves differently when CPU-saturating applications are colocated from when CPU-saturating and non-CPU-saturating applications are colocated. The graph provides a representative (median) sample for every two second interval. TPC-W is running 60 simultaneous sessions. (a) Two CPU-saturating applications: Art and Mesa. (b) A CPU-saturating (Bzip2) and a non-CPU-saturating (TPC-W) application.

behavior of consolidated applications [45]. Similar research on the power consumption of consolidated applications, however, has received much less attention. Such research would be useful to an energy-friendly operation and management of consolidated platforms in a variety of ways. First, it will facilitate the prediction and control of energy consumption in consolidated environments. Second, in combination with existing research on workload characterization and application modeling, it will facilitate meaningful trade-offs between energy costs and application performance. Third, it will enable data centers to operate in regimes that are profitable yet safe from power surges likely to be induced by aggressive consolidation. Finally, ongoing efforts to develop power benchmarks would also benefit from such characterization [38].

Consolidation may occur at multiple spatial granularity, ranging from collocation of multiple applications on a single server to diversion of workloads to a subset of the server racks or rooms. Correspondingly, characterization of power consumption is desirable at each of these levels. Two aspects of power consumption are of particular significance at all these levels. First, the long-term *average power consumption* (several minutes to hours) within a subsystem dictates the energy costs involved in operating it. Second, the possibility of *sustained power consumption* above thresholds associated with fuses/circuit breakers (typically, a few seconds or even subsecond durations) critically affects the safe operation of devices protected by these elements. Thermal effects can also raise the need for both these budgets. At coarse spatial granularity (such as a room), average power may need to be curtailed to avoid excess heating. For smaller components (such as chips), power consumption must be controlled at finer time scales. In this paper, we characterize the power requirement of individual applications and use these characteristics to predict average and sustained power requirement of consolidated applications.

Characterizing the properties of power consumption within a given consolidation hierarchy presents with problems that are significantly different from those encountered in characterizing performance and resource usage. As a motivating example, consider the comparison of power consumption for two different consolidation scenarios, each packing a pair of applications on the same server, as shown in Fig. 1. In each case, we compare the power consumptions of individual applications with that when they were colocated. Power consumption was sampled once every two msec, and

we report the median sample over successive two second intervals as representative of the power samples obtained during that interval. When two CPU-saturating applications (Art and Mesa, two applications from the SPEC CPU2000 suite [37]) are colocated (Fig. 1a), the cumulative power consumption of the server appears to be an average of their individual power consumptions. When a CPU-saturating application (Bzip2, also from the SPEC CPU2000 suite) is colocated with a non-CPU-saturating application (TPC-W [44], an E-Commerce benchmark that spends significant time blocked on I/O), the aggregate power consumed appears to exceed their individual consumptions (Fig. 1b). Roughly speaking, this can be explained as follows: In Fig. 1a, the two applications, both whose power consumption was attributable almost entirely to the CPU, share this resource. On the other hand, in Fig. 1b, the aggregate power consumption depends on the resource usage characteristics of the individual applications including the variance in CPU and I/O usage. While traditional resource usage-aware consolidation is likely to favor scenario Fig. 1b over Fig. 1a, from power consumption perspective Fig. 1a may be considered better than Fig. 1b.

More generally, both average and sustained power consumption in a consolidated environment depend in nontrivial ways on the power consumption as well as resource usage patterns of the individual applications. Prediction of power consumption requires us to accurately identify these dependencies. Furthermore, the success of such prediction also depends on the methodology used to measure and characterize individual consumption. The design of measurement techniques and prediction models that can address these concerns is the focus of this paper.

1.1 Research Contributions

Using a simple combination of hardware and software tools, we design an offline technique to measure the power usage of individual applications. These *power profiles* are converted into convenient statistical descriptions of power usage. Similar profiles are obtained for resource usage of the applications. These profiles are used to build predictive models for average and sustained power consumption of consolidated applications. Two key insights behind these models are: 1) identifying crucial dependencies between the power consumption of individual applications and their resource usage patterns and 2) identifying how key power-consuming resources would be multiplexed among a given set of consolidated applications. Our profiling and prediction

techniques are general enough to be useful for a wide variety of applications and consolidation scenarios.

We conduct an empirical evaluation of our techniques using a prototype server running the Xen virtual machine monitor [2]. This server is capable of consolidating multiple applications, each encapsulated within its own virtual machine. Our evaluation employs a wide variety of applications with diverse power and resource usage behavior to demonstrate the utility and general applicability of our models. Our offline profiling yields crucial insights into the power usage of applications and its relationship with their resource usage. Our predictive models, built upon these insights, appear promising. For a variety of consolidation scenarios, we are able to predict average power consumptions with in a five percent error margin and sustained power consumption within a 10 percent error margin. Finally, we demonstrate how these techniques could be employed to improve system utilization without compromising the safety of its operation.

1.2 Road Map

The rest of this paper is organized as follows: In Section 2, we provide necessary background on power consumption in enterprise-scale environments and formalize the notions of average and sustained power consumption. In Section 3, we develop an offline measurement technique for deriving power profiles of applications. In Sections 4 and 5, we develop and evaluate techniques for predicting average and sustained power consumption, respectively. In Section 6, we evaluate the utility of our prediction techniques in packing applications in Xen-based consolidated settings. We discuss related work in Section 7. Finally, we present concluding remarks in Section 8.

2 BACKGROUND

2.1 Power Consumption in Data Centers

In a typical data center, a primary switch gear distributes power among several uninterrupted power supply substations (UPS; 1,000 KW) that, in turn, supply power to collections of power distribution units (PDUs; 200 KW). A PDU is associated with a collection of server racks (up to 50). Each rack has several chassis that host the individual servers. Power supply could be either at the server-level (as in rack-mounted systems) or at the chassis-level (as in blade servers). Throughout the hierarchy of data center, fuses/circuit breakers are used to ensure that every entity is protected from surges in current drawn.

We focus on characterizing power consumption at these different hierarchies of a data center: 1) at the lowest level, multiple applications are consolidated on a physical server and two) at the higher levels, multiple servers are consolidated within a Power Delivery Unit (PDU). The power supplied to a server is utilized by its various components, including the CPU, memory banks, buses, hard disks, network interface cards, etc. We view server power as composed of *idle* power, *active/busy* power, and I/O power. Idle power is the power consumed by the server when none of the applications are running on it (just running operating-system-related processes). We refer to this as *idle CPU power*. “*CPU power*” is the power consumed by the server when applications are running on it. It is referred to as *active CPU power* in this paper. Note the difference between what we

call idle power and static/leakage power of the CPUs. We call the dynamic power contributed by the I/O devices (disks, NICs, etc.) as *I/O power*. Note that the static power of the I/O devices when they are not doing any activities is included in the *idle* power. Modern CPUs are capable of running in multiple power modes/states including *Dynamic Voltage and Frequency Scaling (DVFS)* states and *Clock throttling* states. DVFS states are determined by the different voltages and frequencies the CPUs can employ. Clock throttling states are typically represented as percentages, which determine the effective duty cycle of the processor. I/O devices including disks and NICs have similar power states. Since I/O power in our environment is significantly smaller than CPU power, we do not break it down among individual I/O devices.

2.2 Average and Sustained Power Budgets

Two aspects of the power consumption within each level of the spatial hierarchy described above play an important role in the safe and profitable operation of the data center. At time scales over which consolidation decisions are made (which, in turn, may be related to the time scales at which workload characteristics change), it may be desirable to limit the energy consumption within the level to values that yield acceptable trade-offs between application performance/revenue and energy costs. Such decision making, likely to be done once every several minutes or hours (we will simply refer to time scales of this order as *long term*), might involve solving complex optimization problems to balance the performance/revenue yielded by operating a subset of the resources against the costs expended toward maintenance, operational power, and cooling. Regardless of the nuances of this decision making, it necessitates mechanisms to enforce limits on the long-term energy expenditure within various levels. We refer to such a limit for a level as the *average power budget* apportioned to it by the consolidation technique.

A second kind of budget, called the *sustained power budget*, results from the reliability needs of various hardware components in the data center defined by fuses¹ or circuit breakers associated with that component. In literature, the phrase *peak power* is sometimes used for this aspect of power usage [15].

The sustained power budget of a hardware component in a data center is represented by the time-current characteristics curve of the circuit breaker deployed in that component. The time-current characteristics curve of circuit breakers is available as part of their spec sheet. Fig. 2 shows the time-current characteristics curve of a typical one Amp circuit breaker, representative of commercial medium-delay circuit breakers [10]. A sustained power budget is represented by a tuple (S, L) , which specifies a bound on the maximum current S that could be sustained over any interval of length L . As shown in the figure, this curve is composed of multiple such sustained power tuples. Tuple A in the figure represents a current of two Amperes passing through the circuit continuously for a period of one second. Similarly, tuple B represents a current of 10 Amperes passing

1. Fuse is a metal wire that melts when heated by a prescribed current, opening the underlying circuit, and thereby, protecting the circuit from overcurrent situation. Circuit breaker is similar to fuse in its function except that it could be reused after a overcurrent situation.

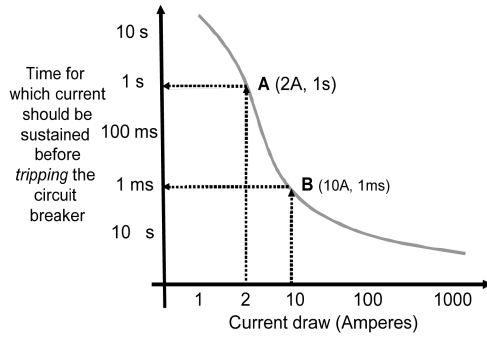


Fig. 2. Time-current characteristics curve of a one Amp circuit breaker. Current drawn from the circuit is provided in the x-axis and the amount of time the current will be sustained before tripping the circuit breaker is provided in the y-axis.

through the circuit continuously for a period of one millisecond. Both the tuples A and B cause the circuit breaker to trip. For simplicity, in this paper, we refer to a single sustained power tuple to represent the sustained power budget of a circuit. Since sustained power consumption corresponds to the maximum power that was continuously sustained over an interval, we find it appropriate to use the effective power consumption of that interval to denote the sustained power consumption. But our prediction technique is general enough to incorporate other statistical bound on power consumption over the specified interval including maximum and average power consumption.

3 POWER PROFILES: MEASUREMENT AND CHARACTERIZATION

In this section, we develop techniques to measure and characterize the power consumption of individual applications. Borrowing techniques from existing research, we also derive characterizations of their resource usage. Finally, we measure and characterize the power and resource usage consumption when these applications are consolidated. Taken together, these measurements set the background for techniques we develop in subsequent sections for predicting useful properties of power consumption in consolidated settings.

3.1 Empirical Derivation of Power and Resource Usage Profiles

Our approach for characterizing the power and resource usage of an application employs an offline profiling technique. The profiling technique involves running the application on an isolated server. By isolated, we mean that the server runs only the system services necessary for executing the application and no other applications are run on the server during the profiling process—such isolation is necessary to minimize interference from unrelated tasks when determining the application’s power and resource usage. The application is then subjected to a realistic workload and a combination of hardware and software monitoring infrastructure is used to track its power and resource usage.

Profiling power consumption. We connect a multimeter to the server used for our offline profiling and use it to measure the power consumption of the server once every

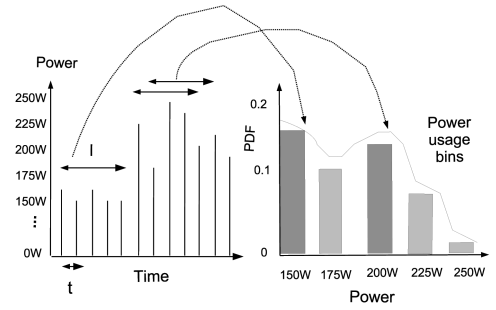


Fig. 3. Illustration of the derivation of power usage distribution from a power profile for $t_p = t$ and $I_p = I$.

t_p time units. We refer to the resulting time series of (instantaneous) power consumption samples as the *power profile* of the application. We find it useful to convert these power profiles into *power usage distributions*. Let $w_A^{I_p}$ be a random variable that represents the average power consumption of the application A over durations of I_p time units, where $I_p = k \cdot t_p$ (k is a positive integer). Note that $w_A^{I_p}$ represents the average consumption over *any* consecutive interval of size I_p . It is estimated by shifting a time window of size I_p over the power profile, and then, constructing a distribution from these values. Fig. 3 illustrates the process of converting a power profile into a power usage distribution. Note that our technique takes each power sample in a power profile to be the power consumption *throughout* the t_p time units preceding it. Clearly, the inaccuracies due to this assumption grow with t_p . As part of our profiling, we also profile the idle power of the server running the applications (approximately 156 W for our server).

Profiling resource usage. We use measurement techniques similar to those existing in research [46] to record resource scheduling events of interest. By recording CPU scheduling/descheduling instants for the virtual machine running our application, we derive its CPU usage profile, an ON-OFF time series of its CPU usage. Similarly, packet transmission/reception times and lengths yield its network bandwidth usage profile. We also record time series of memory consumption and disk I/O requests made by the application. Similar to power measurements, we find it useful to construct resource usage distributions from these profiles. Finally, we also record application-specific performance metrics (e.g., response time and throughput).

3.2 Discussion on Our Profiling Technique

The efficacy of our prediction technique depends crucially on the credibility as well as the feasibility of our offline profiling.

- *On the feasibility of collecting profiles:* The workload used during profiling should be both realistic and representative of real-world workloads. There are a number of ways to ensure this, implying that offline profiling is not unreasonable to assume. While techniques for generating such workloads are orthogonal to our current research, we note that a number of different well-regarded workload-generation techniques exist, ranging from the use of synthetic workload generators to the use of well-known benchmarks, and from trace replay of actual workloads to running the application in a “live” setting. Any such

TABLE 1
Specifications of the Server Used for Profiling

Dell PowerEdge SC1450 Features [11]	
Processor	Two(2) Intel(R) Xeon 64bit 3.4 GHz
Processor Power	80W/110W(Thermal Power)
DVFS states (4)	3.4 GHz, 3.2 Ghz 3.0 Ghz, 2.8 Ghz
Clock throttling states (8)	100%, 87.5% up to 12.5%
Main Memory	2GB
L2 Cache	2MB
Hard Disk	WD Caviar 40GB 7200rpm
Hard Disk Power	7W/1W (Active/Standby)
Network Interface	Dual embedded Intel Gigabit2 NICs
Power Supply	450Wx1

technique suffices for our purpose as long as it realistically emulates real-world conditions. In fact (with regard to running an application in a live setting), many data center applications do start in isolation. Consequently, profiling can be done during the early stages of application deployment, similar to that proposed in current research [46], [41]. Furthermore, workload patterns are often repetitive over some time granularity (such as daily cycles [21]), providing opportunities to incorporate increased confidence into gathered profiles by conducting multiple measurements.

- *Dealing with varying resource/power usage:* Implicit in the power/resource profiles described above is an assumption of stationarity of power/resource usage behavior. Executions of realistic applications are likely to exhibit “phases” across which their power and resource usage behavior change significantly. An example of this is the change in resource needs (and hence, power consumption) of a Web server whose workload exhibits the well-known “time-of-day” variation [21]. Similarly, many scientific applications alternate between doing significant amounts of I/O (when reading in parameters from files or dumping results to them) and computation. Clearly, the utility of our power profiles depends on effectively determining such phases. Power and resource profiles could then be derived separately for every such phase. Enhancing our techniques to deal with these issues is part of our future work.
- *Measurement-infrastructure-related considerations:* While collecting the power profiles of the applications, we ensure that all dynamic power saving daemons are disable. Note that in the actual deployment setting, these daemons are likely to be enabled, and therefore, our power estimate is a conservative estimate.
- *On application modeling:* We do not concern ourselves with identifying relationships between application’s performance metrics (such as response time) and resource usage. This is a well-studied area in itself [40], [4], [9], [12], [45]. We borrow from this literature whenever it is easily done. Generally, we make simplifying assumptions about these dependencies that we expect not to affect the nature of our findings.

TABLE 2
Details of the Multimeter Used in Our Profiling

Signametrics SM2040 Features [36]	
Digits of Resolution	6-1/2
Measurement Rates	0.2/sec - 1000/sec
Measurement Range (AC current)	2.5A
Interface	PCI

3.3 Profiling Applications: Experimental Results

In this section, we profile a diverse set of applications to illustrate the process of deriving an application’s power consumption behavior. We also present selected information about resource usage and performance. These experiments provide us with a number of key insights into: 1) how such profiling should be done; 2) the relationship between an application’s power consumption and its usage of various resources; and 3) the extent of variability in the power consumption of these applications.

Our testbed consists of several Dell PowerEdge servers (details appear in Table 1). We use one of these servers for running the applications that we profile. We connect a Signametrics SM2040 multimeter (details appear in Table 2) in series to the power supply of this server. The multimeter sits on the PCI bus of another server which is solely used for logging purposes. This multimeter is capable of recording power consumption as frequently as once every millisecond. Every recorded power consumption is an effective (or root mean square) power for every millisecond.

Applications that we profile are encapsulated within separate virtual machines and are run on top of the Xen VMM. (The profiles that we obtain include the contribution of the Xen VMM but this is not a problem since it will anyway be present when the applications are consolidated). The server running the application is connected to the multimeter and we use the remaining servers to generate the workload for the application. We ensure that all the machines are lightly loaded and that all nonessential system services are turned off to prevent interference during profiling. We profile a wide variety of applications. In this paper, we report our observations for the representative applications listed in Table 3. In our environment, CPU is the largest contributor to power consumption, so we find it useful to classify these applications based on their CPU usage. Applications in the SPEC CPU suite are *CPU-saturating* in that they are ready to use the CPU at all times. The remaining applications (non-CPU-saturating) alternate between using the CPU and being blocked (e.g., on I/O, synchronization activities, etc.) and their CPU utilization depends on the workload they are offered. We profile these

TABLE 3
Salient Properties of Our Applications

Applications	
TPC-W [44]	3-tiered NYU implementation of the TPC-W transactional Web-based E-commerce benchmark
Streaming Media	Home-grown UDP streaming server, runs with specified no. of clients and data rate
SPEC CPU2000 [37]	SPEC CPU2000 suite (Art, Bzip2, Mcf, Mesa)

TPC-W and Streaming server are non-CPU-saturating, whereas applications in the SPEC CPU2000 suite are CPU-saturating.

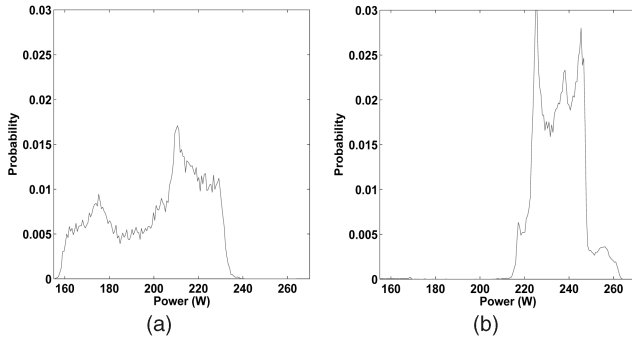


Fig. 4. Power distributions of Streaming server and Mcf compared. (a) Streaming, 60 clients (SM(60)). (b) Mcf.

non-CPU-saturating applications at different workload intensities. TPC-W is profiled with the number of simultaneous Web sessions varying from 10 to 100 in increments of 10. For experiments involving TPC-W, we represent the workload intensity TPC-W(x), where “ x ” is the number of sessions. For experiments involving Streaming Media Server, 3 Mbps is used as the streaming rate; Streaming(x) represents a workload of “ x ” clients.

We now present key results from our profiling study. Throughout this section, we have $t_p = I_p = 2$ msec (sampling interval). We begin by observing the power distributions of our applications and comparing key properties. We present a subset of these results in Fig. 4.

Given that CPU consumes significantly more power than I/O devices in our environment, not surprisingly, power distributions for non-CPU-saturating application (Figs. 4a) are found to exhibit higher variance than CPU-saturating application (Figs. 4b). Specifically, the variance of the TPC-W and Streaming profiles were 92 and 84 W^2 compared with only 47 and 59 W^2 for Bzip2 and Mcf, respectively. For all our applications, we find that their power consumption, when running on the CPU, does not vary a lot over time (that is, our chosen application does not exhibit multiple phases with respect to power consumption as was mentioned in Section 3.2). Even an application like Mcf, that is known to exhibit significant temporal variations in its memory access patterns, is found to not show excessive temporal variation in its power consumption. Consequently, the power distribution of Mcf is similar to that of Bzip2 that does not exhibit such variations. This observation leads us to realize that one primary contributor to variance in the power usage of an application is a *change in its CPU scheduling state*. The CPU profile of a non-CPU-saturating application exhibits an ON-OFF behavior, corresponding to the application being in running and blocked states, respectively. When such an application blocks, its power consumption corresponds to the server’s idle power. This ON-OFF CPU usage contributes to the higher variance in its power consumption.

Observation 1. Typically, power distributions of non-CPU-saturating applications exhibit higher variance (and longer tails) than those of CPU-saturating applications.

Next, we study the impact of changing CPU power states (DVFS and clock throttling) on the power consumption of

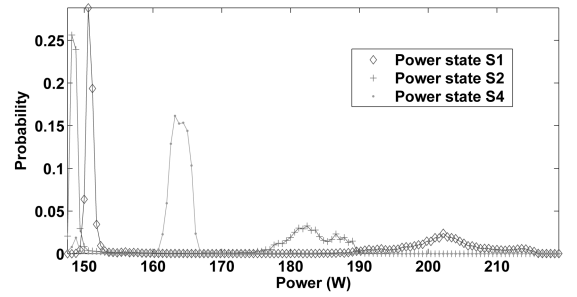


Fig. 5. Power distribution of TCP-W(60) collected at different CPU power states.

applications. We represent the power states of the CPU as a tuple (p, q) , where p represents the DVFS state and q represents the percentage duty cycle due to clock throttling. CPU power states are represented as S1: (3.4 GHz, 100 percent), S2: (3.2 GHz, 100 percent), S3: (3.0 GHz, 100 percent), and S4: (2.8 GHz, 50 percent). As a representative result, Fig. 5 shows the power PDF of TPC-W(60) when it was run with the CPU at three different CPU power states; Table 4 presents its CPU usage profile and performance during each of these executions. When a non-CPU-saturating application is run at a lower CPU power state, the fraction of time the CPU remains idle decreases. The peak power consumption reduces upon running an application at a lower power state. However, this reduced peak is sustained for a longer period of time than in the high-power state.

Observation 2. For non-CPU-saturating applications, CPU utilization increases and power distribution becomes less bursty at lower power states.

Finally, we observe the trade-off between power consumption and application performance as the CPU state is varied. Table 5 presents this comparison for a CPU-saturating application (Bzip2) and non-CPU-saturating application (TPC-W(20)). The performance metric for Bzip2 was program completion time; the metric for TPC-W was average response time. As seen in Tables 5 and 4, changes in

TABLE 4
CPU Usage of TPC-W(60) Operating at Three Different DVFS States

Power state	CPU Utilization				Normalized Performance degradation	Power (Watt)
	Average	95th	99th	Peak		
S1	0.41	0.92	0.93	0.95	1	185.6
S2	0.44	0.93	0.95	0.98	1.18	175.3
S4	0.92	0.97	0.98	0.99	15.69	173.2

TABLE 5
Impact of DVFS States on Power Consumption and Performance of Applications

Power state	Bzip2			TPC-W(20)		
	Power (W)	CPU (frac.)	Norm. degrad.	Power (W)	CPU (frac.)	Norm. degrad.
S1	224.9	0.98	1	164.7	0.14	1
S2	200.1	0.99	1.10	161.9	0.15	1.07
S3	189.2	0.99	1.19	160.5	0.16	1.12
S4	172.1	0.99	2.19	161.3	0.33	2.02

TABLE 6
Impact of DVFS States on Power Consumption and Application Performance Observed on Newer Generation Intel Quadcore Nehalem Processor

Power state	RUBiS(1000)			Bzip2		
	Power (W)	CPU (frac.)	Norm. degrad.	Power (W)	CPU (frac.)	Norm. degrad.
P1	127.01	0.22	1.00	177.79	0.99	1.00
P3	124.53	0.23	1.11	169.92	0.99	1.11
P5	123.05	0.26	1.22	159.53	0.99	1.23
P7	122.37	0.30	1.44	150.30	0.99	1.38
P9	122.46	0.33	1.56	143.79	0.99	1.60

average power usage are more pronounced for CPU-saturating applications, reduction of 50 W (between the two extreme power states in the table) for Bzip2, compared with a reduction of only 15 W for TPC-W(60) and negligible reduction for TPC-W(20). Non-CPU-saturating applications spend significant amounts of time idling on the CPU, and therefore, a change to CPU power state has less effect on their average power consumption. Their average power usage is dominated by idle power, unless they are subjected to high-intensity workloads requiring them to consume significant CPU cycles. However, while the performance degradation is easy to predict for CPU-saturating applications (directly proportional to the ratio of clock rates), it can depend in nontrivial ways on the workloads of non-CPU-saturating applications. For example, with a higher load (60 simultaneous browsing sessions), TPC-W(60) exhibits 15.69 times longer response time, while it suffers only 2.02 times response time degradation with 20 simultaneous sessions. The higher performance degradation for TPC-W(60) is due to its very high CPU utilization (92 percent—almost close to saturation). This results in the TPC-W threads spending lot of time waiting in the run queue, further delaying the time for receiving the next I/O request.

Observation 3. Power performance trade-offs when operating at different CPU power states differ significantly for CPU-saturating and non-CPU-saturating applications. While it is easy to predict for CPU-saturating applications, it can depend on nontrivial ways for non-CPU-saturating applications.

To illustrate the generality of our observations on newer generation processors, which feature a much richer set of DVFS states with low idle power component, we present power/performance result of applications running on a Intel quad-core 2.6 Ghz nehalem processor in Table 6. Our nehalem server (Xeon X5550, Dell PowerEdge R610) features nine DVFS power states, represented as P1-P9 in Table 6, each power state differing in frequency by 133 MHz. Table 6, apart from Bzip2, also presents results for RUBiS [35], a popular three-tier e-commerce benchmark loosely modeled after eBay.com. We configured RUBiS to handle 1,000 simultaneous client sessions. Our results on the nehalem processor corroborate with our earlier observations two and three.

3.4 Power Profiles of Consolidated Applications: Experimental Results

Next, we present our salient observations on power consumption of consolidated applications.

TABLE 7
Average Power Consumption of CPU-Saturating Applications and Non-CPU-Saturating Applications Behave Differently

Applications Consolidated	Art+Mesa		Bzip2+TPC-W(60)	
	Art	Mesa	Bzip2	TPC-W(60)
Power (W)	227.1	217	224.1	185.6
Consolidated Power (W)	226.1		224.6	

We had seen in Fig. 1, the server power consumption in two different consolidation scenarios, each colocating a pair of applications. In Table 7, we present the observed average power consumption for two sets of consolidation. The power consumed when two CPU-saturating applications (Art and Mesa) are colocated was close to the average of individual power usages. This happened because the sole significant power consuming resource—the CPU—was *time-shared* by these two applications. When a non-CPU-saturating application (TPC-W(60)) was colocated with a CPU-saturating application (Bzip2), however, the aggregate power consumption seems to exceed the average of the power consumption of the individual applications. Roughly, this happens because this pair of applications exercises both CPU and I/O devices concurrently.

Generalizing the above observation, for predicting power consumption of consolidated applications, it is important to separate out the significant components of power (e.g., CPU versus I/O power) consumed by individual applications. Furthermore, these components need to be considered along with usage patterns of the relevant resources.

Observation 4. There exist significant differences in average power consumption when colocating CPU-saturating applications versus when colocating CPU-saturating and non-CPU-saturating applications.

Similar observations apply to the behavior of sustained power consumed by a set of colocated applications. Fig. 6 plots the effective power usage seen during nonoverlapping intervals of length two sec each for two consolidation settings: 1) when CPU-saturating applications are colocated and 2) when a CPU-saturating and a non-CPU-saturating application are colocated. The effective power usage during each interval is its sustained power consumption. In each case, we present the sustained power usage for individual applications as well as upon consolidation.

Observation 5. Sustained power consumption for consolidated applications behaves significantly differently from the average power consumption.

4 AVERAGE POWER PREDICTION

In this section, we develop techniques to predict the average power consumption of a server which consolidates multiple applications. The inputs to our prediction algorithm are the power and resource usage distributions of individual applications.

4.1 Baseline Prediction

We begin with the following simple *baseline* predictor for average power consumption of a server on which

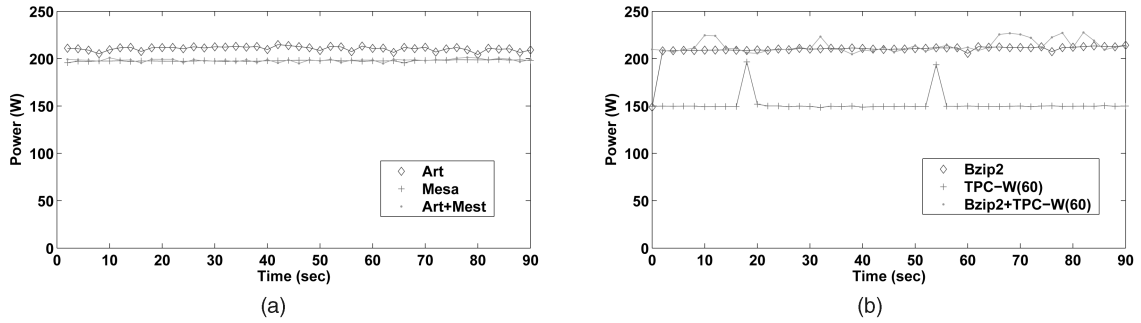


Fig. 6. Sustained power when CPU-saturating applications are colocated behaves differently from when CPU-saturating and non-CPU-saturating applications are colocated. (a) Two CPU-saturating applications: Art and Mesa. (b) A CPU-saturating, Bzip2, and a non-CPU-saturating, TPC-W(60) application.

n applications A_1, \dots, A_n are consolidated. Our predictor employs a sum of the average power consumptions of the individual applications, weighted by their respective CPU utilization:

$$\bar{P}_{A_1, \dots, A_n} = \sum_{i=1}^n (\bar{P}_{A_i} \cdot R_{A_i}^{cpu}) + \bar{P}^{idle} \cdot \left(1 - \sum_{i=1}^n R_{A_i}^{cpu}\right), \quad (1)$$

where \bar{P}_{A_i} is the average of the power distribution of the application A_i (obtained from the offline profile); \bar{P}^{idle} is the power consumption when CPU is idle; and $R_{A_i}^{cpu}$ ($0 \leq R_{A_i}^{cpu} \leq 1$) is the CPU allocation for it.² Note that \bar{P}_{A_i} is the average of the total system power measured when application A_i alone is running on the server and this includes the power consumed by the applications in all the components of the server. The first term captures the power dissipation of the server when the CPU is busy, whereas the second term is for when it is idle. We present in Table 8 the efficacy of baseline prediction in three consolidation settings, each colocating a pair of applications.

In the first consolidation setting, two CPU-saturating applications, Art and Mesa, time-share the CPU equally, and our baseline approach proves to be an excellent predictor of average power. In the latter two consolidation settings, where we have non-CPU-saturating applications, we observe increasing error margins in the baseline prediction. The main reason for these inaccuracies is that the first quantity in (1), \bar{P}_{A_i} , represents the average of the *entire* power distribution for application A_i including the durations when the CPU was idle (when A_i is blocked on some event). Non-CPU-saturating applications can have significant such idle periods. Upon consolidation, however, these idle durations are likely to be occupied by other colocated applications. Therefore, we need to employ the average power consumption by the application *only* over durations when it was using the CPU.

4.2 Improving Prediction of Average Power

We are interested in determining the average power consumed by an application only over durations when it was scheduled to run on the CPU. This can be estimated by considering the power distribution beyond its $100 \cdot (1 - U_A^{cpu})^{th}$ percentile, where U_A^{cpu} is the average CPU usage for A as obtained from its offline CPU usage profile

2. These CPU allocations for the applications are set by the administrator in the consolidated setting. Figuring out these allocations for the applications is done using well-studied techniques for application modeling [45].

(note that U_A^{cpu} should not be confused with the allocation R_A^{cpu} that we encountered above). The average power for this subset of the entire distribution corresponds exactly to the power consumed when the application was running on the CPU. Fig. 7 shows this process for TPC-W(60) whose CPU utilization was 40 percent. We denote this quantity by $P_{A_i}^{busy}$ and replace P_{A_i} in (1) with it. In this example, P_{tpcw}^{busy} was found to be 225 W, while P_{tpcw} was 185 W—note the difference.

Based on observations from several consolidation scenarios, we find that this enhancement results in improved predictions of average power. As specific examples, recall the prediction of average power for a server hosting: 1) Art and TPC-W(60) and 2) TPC-W(60) and TPC-W(10) (Table 8). With this enhancement, our estimates of average power improved from 209 to 226 W for 1), and from 167 to 188 W for 2) which reduces the error margins to 1.76-1.3 percent, respectively. In Fig. 8, we present the performance of our baseline and enhanced predictors for a variety of consolidation scenarios. Our enhanced technique is able to predict within five percent of the observed power, while the baseline approach has up to 20 percent error margin.

TABLE 8
Baseline Predictor of Average Power Consumption

Applications consolidated	Baseline prediction (W)	Observed average (W)	Error (%)
Art + Mesa	222.1	226.1	1.7
Art + TPC-W(60)	209.3	224.6	6.8
TPC-W(10) + TPC-W(60)	167.4	190.1	11.9

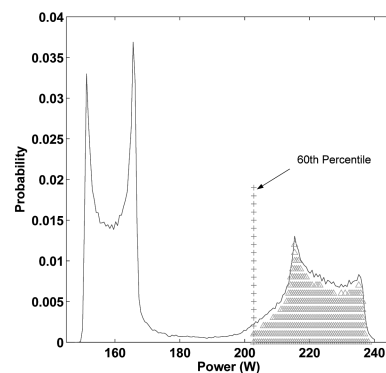


Fig. 7. Capturing the nonidle power portion for TPC-W(60).

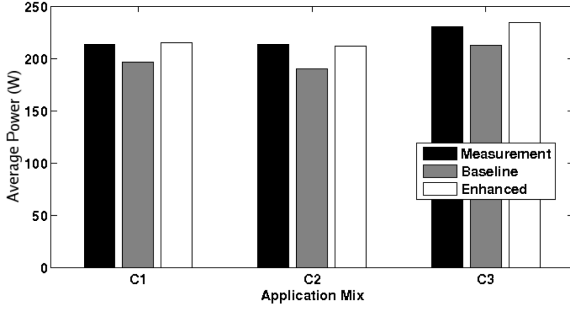


Fig. 8. Comparison of our predictors for a variety of consolidation scenarios. In C1 and C2, the applications are consolidated on the same physical processor and the second processor is idle. In C3, two applications each are consolidated on each processor. C1: TPC-W(60) + Bzip2 + SM(100), C2: TPC-W(60) + Bzip2 + SM(100) + SM(100), C3: Processor1: TPC-W(60)+SM(100), Processor2: SM(100) + Bzip2.SM(x) represents the streaming server with x clients.

Our prediction technique assumes that the total system power is in proportion with the CPU utilization of the application. Though this turns out to be a good assumption for our environment (where CPU is the dominant consumer of power), this may not be the case where more power-consuming I/O components (e.g., a graphics card, networking equipment, etc.) are deployed.

5 SUSTAINED POWER PREDICTION

Next, we turn our attention to sustained power budgets in consolidated servers. Recall that sustained power budgets are enforced at different levels in a data center to ensure that power consumption of the applications consolidated under that level does not cause the capacity of the corresponding PDU to be exceeded. In Section 5.1, we develop mechanisms to predict the sustained power consumption of applications consolidated on a server. At a single server level, sustained power prediction boils down to finding the possibility of a single server consuming enough power to reach the limit of its power supply. Though this may seem unlikely given the fact that the capacity of a server-level power supply is typically much higher than its peak power consumption [14], this method will later be extended to predict the possibility of applications consolidated on a set of servers reaching the capacity of their PDU which is very much a concern. Also, taking a cue from the aforementioned overprovisioning present in existing power supplies, recent research [26] has suggested using lower capacity (and cheaper) power supplies for servers to cut costs. Such an approach could make use of effective prediction techniques to determine the possibility of the limit of a power supply being violated. In Section 5.2, we extend our approach to predict sustained power violation at the PDU or rack level (across a set of servers).

5.1 Sustained Power Prediction for a Server

Recall that our goal is to predict the probability $Pr_{A_1, \dots, A_n}(S, L)$, upon consolidating n applications A_1, \dots, A_n on a server, of S or more units of power being consumed for any L consecutive time units. We will refer to Pr_{\dots} as the *probability of violating the sustained power budget*. Note the

trivial feasibility requirement on the sustained power budget that it always be higher than the idle power—if this does not hold, the server would be inoperable. I/O power contribution for our set of applications and servers is very low, and therefore, we do not consider it for predicting sustained power. Note that we are not ignoring power contributed by I/O components. The entire server power (from all the components in the server) is assumed to be proportional to the CPU utilization.

5.1.1 Baseline Prediction

To bring out the difficulties in predicting the probability of a given sustained power consumption, we evaluate a simple baseline approach that operates as follows: It first estimates the number of slots m_i , each of length t_p (recall that t_p is our power measurement granularity), during which the application A_i is expected to occupy the CPU over a duration of L time units:

$$m_i = \frac{L \cdot R_{A_i}^{cpu}}{t_p},$$

where $R_{A_i}^{cpu}$ is the CPU allocation for application A_i in the consolidated setting. Assuming stationarity of power consumption across durations of length t_p for each of the applications, the probability of violation is estimated as

$$Pr_{A_1, \dots, A_n}(S, L) = \prod_{i=1}^n \{Pr(w_{A_i}^{t_p} \geq S)\}^{m_i}, \quad (2)$$

where $Pr(w_{A_i}^{t_p} \geq S)$ is the probability that application A_i 's power consumption obtained from its power distribution at the granularity of t_p time units exceeds the sustained power limit S . Note that the above assumes independence among the execution patterns of colocated applications—a reasonable assumption in our settings. We make this assumption throughout this section. There are three key shortcomings in the baseline approach.

Shortcoming (A) due to assuming saturated CPU: First, the baseline approach does not capture the likelihood that the CPU could be idle for some portion of given durations of length L . Any such duration should not qualify as one where a violation of sustained power budget occurs (recall we assume that the sustained budget is greater than idle power).

Shortcoming (B) due to stationarity assumption: Second, the assumption of stationarity of power consumption at the granularity of t_p time units holds only for a very selected type of applications. Among our set of applications, this applies only to some of the CPU-saturating applications which do not exhibit large temporal variation in their power consumption. An example of an application that does exhibit such variations is the CPU-saturating Mcf—it exhibits temporal variations in its memory access patterns resulting in variations in its power consumption measured at the granularity of t_p units. We have already seen that all our non-CPU-saturating applications exhibit significant variations in power usage.

Shortcoming (C) due to ignoring CPU usage variation: Finally, the baseline approach assumes that the CPU usage of each of the colocated applications would be *precisely* equal to their CPU allocations ($R_{A_i}^{cpu}$ for application A_i) over

any period of length L . Again, while this assumption is fine for a set of colocated CPU-saturating applications whose CPU usage patterns do not exhibit variability, it introduces inaccuracies when there is even one application that does not adhere to such behavior. In particular, it becomes inaccurate when predicting the sustained power behavior of a set consisting of one or more non-CPU-saturating applications (when these applications are blocked on I/O activities, those idle periods will likely be used by other colocated applications resulting in a different CPU allocation than specified by $R_{A_i}^{cpu}$ for the applications).

In the rest of this section, we will address these three shortcomings.

5.1.2 Improving Prediction of Sustained Power

Addressing shortcoming (A). We had encountered a similar problem when dealing with prediction of average power. The idle periods included in the CPU profiles of non-CPU-saturating applications must be handled correctly because upon consolidation, they are likely to be occupied by other colocated applications. Exactly as in Section 4.2, we remove this idle portion by only considering the power distribution beyond its $100(1 - U_A^{cpu})^{th}$ percentile, where U_A^{cpu} is the average CPU usage for A as obtained from its CPU usage profile.

Addressing shortcoming (B). This problem arose because we used the distribution of power over t_p units to predict the probability that an application A_i consumes power above S units for $T_{m_i} = m_i \cdot t_p$ consecutive time units during which it occupies the CPU. We can improve this prediction by using the power profile for A_i to explicitly find this probability, rather than relying on the power distribution over t_p time units. Equivalently, given m_i , we are interested in the power distribution over $m_i \cdot t_p$ time units. This distribution is easily derived from the power profile by moving a window the size of $m_i \cdot t_p$ time units, shifting it by t_p units. We find the maximum sustained power of the application by choosing the window having maximum sustained power. The sustained power of each window is calculated by finding the *effective* power from m_i samples in each window.³ The samples obtained from these windows are converted into a distribution of sustained power over $m_i \cdot t_p$ time units. With this modification, our predictor takes the following form (compare with (2)):

$$Pr_{A_1, \dots, A_n}(S, L) = \prod_{i=1}^n \{Pr(w_{A_i}^{T_{m_i}} \geq S)\}. \quad (3)$$

We provide an example to illustrate the seriousness of this problem. For TPC-W, the probability of violating a budget of (200 W, 2 ms) as obtained from TPC-W power profile collected at 2 ms is 99.64 percent. The probability of violating 200 W for a period of one second would be approximated as $(0.9964)^{500}$ which is 16.47 percent. But when we compute the actual sustained power by moving

3. An each power sample can be derived to Q by $P = Q/t$, where Q is the electric charge. Accordingly, the effective power of each window is $\frac{\sum Q}{T_{m_i}}$.

window of size one sec over TPC-W time series, we find this probability to be 21.55 percent.

Addressing shortcoming (C). We find this to be the most interesting and challenging shortcoming to address. Recall that this shortcoming arose because we assumed that the applications would always consume the CPU exactly in accordance with their CPU allocations ($R_{A_i}^{cpu}$ for application A_i). We attempt to alleviate this problem by incorporating into our prediction the multiple ways in which the CPU could be shared among the applications over durations of length L . We derive this using the individual applications' CPU profiles collected over duration of length L . Let $Pr\{U_{(A_1, \dots, A_n)}^L = (c_1, \dots, c_n)\}$ be the probability that (c_1, \dots, c_n) are the fractional CPU consumptions of applications (A_1, \dots, A_n) , respectively, over all intervals of length L in our consolidated setting. We estimate this for all possible combinations of (c_1, \dots, c_n) as follows: The CPU utilization of the applications in the consolidated setting depends mainly on two things: 1) the CPU reservation of applications in the consolidated setting and 2) the CPU requirement of the applications (both over periods of length L). When the reservation is higher than the requirement, then it means that the application has spare CPU which could be used by other applications whose reservations are smaller than their requirements. Most reservation-based schedulers (as ours) divide this spare CPU equally among these needy applications. Our estimate for the CPU utilization of applications in the consolidated setting takes the above into account. We start by constructing the distributions of fractional CPU requirements for every application over durations of length L . These distributions can be easily derived from the CPU usage profiles of the applications. Let $Pr(U_{A_i}^L \geq c)$ represent the probability that the CPU utilization of application A_i over duration of length L exceeds c , where $(0 \leq c \leq 1)$. We construct CPU requirement bins (RBs) for every application over bins of length δ , $RB_{A_i}[\delta, 2\delta, 3\delta, \dots, (1 - \delta), 1]$, where $RB_{A_i}[j] = Pr(U_{A_i}^L \geq (j - \delta)) - Pr(U_{A_i}^L \geq j)$, that is each bin represents the probability of utilization being within its bin boundaries. (RB is straight forward to obtain from our offline profile.)

For simplicity, we conduct the rest of the discussion in the context of two applications A_1 and A_2 . We are interested in finding $Pr(U_{(A_1, A_2)}^L = (c_1, c_2))$ (which is the probability that CPU utilization of application A_1 is c_1 and that of A_2 is c_2) for $c_1 = \delta, 2\delta, \dots, 1$ and $c_2 = \delta, 2\delta, \dots, 1$. It could be obtained by

$$RB_{A_1}[\delta, \dots, 1] \cdot (RB_{A_2}[\delta, \dots, 1])',$$

where $(RB_{A_2}[\delta, \dots, 1])'$ represents the transpose of the array $RB_{A_2}[\delta, \dots, 1]$. Multiplying these two one-dimensional matrices will generate a two-dimensional matrix which provides the probability for all utilization pairs (c_1, c_2) .⁴ Note that in $Pr\{U_{(A_1, A_2)}^L = (c_1, c_2)\}$, $(c_1 + c_2)$ ranges from 0 to two. But in a consolidated setting, utilization of the CPU cannot go beyond 100 percent. Therefore, we estimate the utilization of the applications in the consolidated setting by using the following procedure:

4. This extends easily to n applications; we omit these details here.

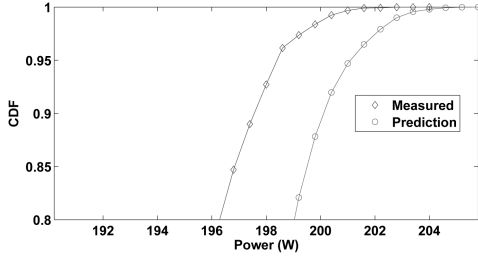


Fig. 9. Comparison of measured and predicted sustained power consumption ($L = 1$ sec) for a server consolidating TPC-W(60), TPC-W(60), and Bzip2 with reservations 60, 20, and 20 percent, respectively.

Let r_1 and r_2 be the reservations for the application A_1 and A_2 , respectively.

- 1: **for all** (c_1, c_2) such that $c_1 + c_2 > 1$ **do**
- 2: **if** $(c_1 > r_1)$ and $(c_2 > r_2)$ **then**
- 3: $Pr(U_{(A_1, A_2)}^L = (r_1, r_2)) = Pr(U_{(A_1, A_2)}^L = (r_1, r_2)) +$
 $Pr(U_{(A_1, A_2)}^L = (c_1, c_2))$
- 4: **else if** $(c_1 > r_1)$ and $(c_2 < r_2)$ **then**
- 5: $Pr(U_{(A_1, A_2)}^L = (1 - c_2, c_2)) = Pr(U_{(A_1, A_2)}^L =$
 $(1 - c_2, c_2)) + Pr(U_{(A_1, A_2)}^L = (c_1, c_2))$
- 6: **else if** $(c_1 < r_1)$ and $(c_2 > r_2)$ **then**
- 7: $Pr(U_{(A_1, A_2)}^L = (c_1, 1 - c_1)) = Pr(U_{(A_1, A_2)}^L =$
 $(c_1, 1 - c_1)) + Pr(U_{(A_1, A_2)}^L = (c_1, c_2))$
- 8: **end if**
- 9: **end for**

Lines 2 and 3 handle the case when the (fractional) CPU requirement of both the application is above their reservations; in this case, the CPU usages of the applications in the consolidated setting would be (r_1, r_2) . Lines 4-7 handle the case when the CPU requirement of one application is below its reservation and the other application is above its reservation in which case the needy application gets CPU from the surplus of the other. Also, note that for all durations with some idle portion ($c_1 + c_2 < 1$), the probability of sustained power violation would be zero (recall the trivial feasibility requirement for sustained power budget which requires it to be higher than idle power). This is captured by line 1 of the algorithm.

For any duration of length L units where the applications A_1 and A_2 occupy the CPU for c_1 and c_2 time units, the probability of sustained power (S watts) violation is given by $Pr_{A_1, A_2}(S, L) = Pr(w_{A_1}^{c_1} \geq S) * Pr(w_{A_2}^{c_2} \geq S)$ (similar to what was described in (3)). We use the above equation to find the probability of sustained power violation for all possible (c_1, c_2) pairs as follows:

- 1: $Pr_{A_1, A_2}(S, L) = 0$
- 2: **for** $i = 0$ to 1 ; $j = 1$ to 0 ; $i = i + \delta$; $j = j - \delta$ **do**
- 3: $Pr_{A_1, A_2}(S, L) = Pr_{A_1, A_2}(S, L) + Pr(w_{A_1}^i \geq S) \cdot$
 $Pr(w_{A_2}^j \geq S) \cdot Pr\{U_{(A_1, A_2)}^L = (i, j)\}$
- 4: **end for**

Line 2 of the above algorithm loops through all possible (c_1, c_2) pairs that add up to 1 (possible violation regions). The algorithm predicts the probability of sustained power budget violation for a given value of the power budget S .

TABLE 9
Efficacy of Sustained Power Prediction on a Server Consolidating TPC-W(60), TPC-W(60), and Bzip2 with Reservations 60, 20, and 20 Percent, Respectively

Power Percentile	Measured Sustained power (W)	Predicted sustained power (W)	Error (%)
20%	196.3	199.0	1.37
10%	197.5	200.1	1.31
1%	200.2	202.8	1.29
0%	204.0	207.0	1.47

We compare the tail of the measured power with our predicted power.

We run the above algorithm for $L = 1$ sec and varying S from 0 to 300 W for our experiments. Fig. 9 and Table 9 evaluate our prediction mechanism for a server consolidating three applications. We are able to bound the tail of the sustained power consumption within two percent error margin (Table 9).

5.2 Sustained Power Prediction across Multiple Servers

Having predicted the sustained power consumption of a single server, we next predict the probability of sustained power budget violation across a set of servers. Our goal is to predict the probability $Pr_{B_1, \dots, B_m}(S, L)$ (that upon consolidating m servers B_1, \dots, B_m on a PDU) of S or more units of power being consumed by the PDU for any L consecutive time units. Unlike the case when applications time-share the server, in this case, the applications are running simultaneously, and therefore, the power consumption would add up. Recall from Section 2.2 that we are interested in finding the minimum power consumption of the PDU over periods on length L time units. This minimum power consumption of the PDU (consisting of set of servers) is upper bounded by the sum of average power (over intervals of length L time units) of the individual servers. The proof is very simple. Consider two sets U and V consisting of k elements each. Let W be a set obtained by adding any permutation of the set U with any permutation of the set V (note that set W also has k elements). The minimum value in set W , W_{min} , is upper bounded by its average W_{avg} ($W_{min} \leq W_{avg}$). Note that average of the set W is nothing but the sum of the averages of the sets U and V ($W_{avg} = U_{avg} + V_{avg}$). Therefore, the sum of the averages of the sets U and V forms the upper bound of the minimum in set W ($W_{min} \leq U_{avg} + V_{avg}$).

We use the above idea to bound the maximum power sustained by the PDU. This can be achieved in two steps:

Step1. Find the distribution of average power consumption of individual servers (connected to the PDU) over intervals of length L time units.

Step2. Add all these average power distributions. Assuming individual consumptions to be independent—a reasonable assumption—the resulting distribution of the aggregate can be computed from elementary probability theory.⁵

5. This is done using the z-transform. The z-transform of a random variable U is the polynomial $Z(U) = a_0 + za_1 + z^2a_2 + \dots$, where the coefficient of the i th term represents the probability that the random variable equals i (i.e., $U(i)$). If U_1, U_2, \dots, U_{k+1} are $k+1$ -independent random variables, and $Y = \sum_{i=1}^{k+1} U_i$, then $Z(Y) = \prod_{i=1}^{k+1} Z(U_i)$. The distribution of Y can then be computed using a polynomial multiplication of the z-transforms of U_1, U_2, \dots, U_{k+1} .

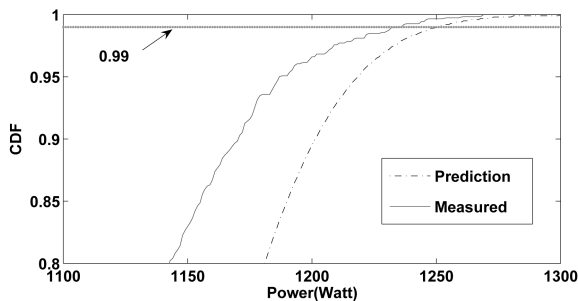


Fig. 10. Comparison of measured and sustained power consumption ($L = 5$ sec) of a PDU connected to seven Servers (each running TPC-W(60)).

Step1 can be easily achieved by slightly modifying the techniques developed in Section 5.1. Initially, we estimate the CPU utilization of the consolidated applications $Pr(U_{(A_1, \dots, A_n)}^L = (c_1, \dots, c_n))$ for all (c_1, \dots, c_n) , and then, instead of finding the effective power consumption of the server, we compute the average power consumption of the server using the distribution of average power consumption of the individual applications (obtained from offline profiling) over intervals of length L time units. Step 2 is straightforward.

A representative result is presented in Fig. 10 and Table 10. For space sharing experiments, since we need to measure power consumption of more than one server, we use a PDU from Raritan [34] which could measure power at the PDU level. Even though our prediction technique is capable of predicting the power consumption of a server consolidating multiple applications, for simplicity, in this paper, we assume that only one application runs on a server. As shown, for a PDU connected to seven servers, each consolidating TPC-W(60), our technique predicts within reasonable error margins (one-five percent). One important characteristic of our prediction technique worth mentioning here is that our prediction provides an upper bound for sustained power consumption of the aggregate.

5.3 Discussion on Sustained Power Budget Violation

Violation of sustained power budgets may result in degraded performance/availability, and therefore, it is imperative for a data center to develop techniques that ensure safe operation within sustained power limits. Fortunately, commercially available medium delay circuit breakers can tolerate up to 50 percent increase in power consumption (relative to the rated capacity of the circuit) for about 5-10 seconds, before tripping the circuit, which gives enough time for reactive throttling techniques to bring back the system under safe power limits. We evaluated the efficacy of such a reactive technique in our recent work [18], where we use the prediction technique developed in this paper to overbook the provisioned power capacity of a data center. Also, current servers are equipped with power meters that can measure power consumption at very fine granularity [26] (in milliseconds) and have in-built mechanisms [24], [22] that can limit power consumption at very fine time scales (less than a second). These device capabilities suggest the feasibility of designing a data center that employs reactive technique to deal with sustained power budget violation and still operate safely.

TABLE 10
Efficacy of Our Sustained Power Prediction on a PDU Consolidating Seven Servers Each Running TPC-W(60)

Power percentile	Measured Sustained power (W)	Predicted sustained power (W)	Error (%)
80	1143	1181	3.2
90	1171	1201	2.4
99	1236	1250	1.1
100	1269	1300	2.4

We compare the tail of the measured power with our predicted power.

6 POWER-AWARE PACKING

We now examine the utility of our prediction techniques in making consolidation decisions. A key component of a consolidation-based system is a *packing* algorithm that dynamically decides, based on changing workload conditions, which server machines the hosted applications should be made to run till its next invocation. In an energy-conscious platform, the packing algorithm should incorporate both performance (resource) and power (average and sustained) considerations into its decision making.

To illustrate how our prediction techniques can facilitate performance and power-aware packing, we present the results of one representative experiment where such packing is evaluated. We consider the problem of packing one or more applications from the following set on our server: two copies of TPC-W, each serving 20 sessions and one Streaming applications serving 60 clients. We obtain the power consumption and resource usage of these applications at various power states from our offline profile.

Since our work is not aimed at investigating the relationship between application-level performance goals and resources needed to meet them, we choose workloads that were empirically found to be sustainable even at the lowest CPU power state of the server. As mentioned before, we consider this research as complementary but orthogonal to our work.

We choose the following power budgets: 1) an average power budget of 180 W and 2) a sustained power budget of 185 W per second. It must be mentioned here that we do not claim that these budgets (particularly, the sustained budget) are realistic; in fact, these may appear to be rather low to the reader. These values have just been chosen to bring out important aspects of our packing strategy without having to conduct extremely large-scale consolidation experiments. However, we believe that our results represent general trends that are likely to apply in more realistic consolidation scenarios.

We pick these budgets so that they are feasible for any of our applications individually with the CPU operating at the highest power-consuming state. We use our prediction algorithms to determine the average and sustained power consumption upon packing different subsets of the applications with the CPU operating at various available power states. For this experiment, the clock throttling state is not changed.

Our techniques predict that the only feasible configuration where all three of our applications could be colocated on the same server while meeting both the power budgets is when the CPU operates at DVFS3 (CPU operating at 2.8 GHz). Furthermore, we predict that packing any two of our

TABLE 11
Predicted Values for Sustained and Average Power Consumption for Two Subset of Applications at Two Processor Power States

Applications consolidated	DVFS0		DVFS3	
	avg. (W)	sust. (W)	avg. (W)	sust. (W)
TCP-W(20)+TPC-W(20)	178.0	190.1	176.2	174.5
TPC-W(20)+TPC-W(20)+Streaming	193.5	193.2	177.0	172.0

applications with the CPU at the highest state DVFS0 (CPU operating at 3.4 GHz) would result in a violation of at least one of the power budgets. We present our predictions for two of these subsets S_1 (TPC-W(20) + TPC-W(20)) and S_2 (TPC-W(20) + TPC-W(20) + streaming) and the two CPU DVFS states in Table 11. Based on our prediction, we recommend a packing of all three applications with the server using the DVFS3 state.

We then conduct a series of experiments to evaluate the efficacy of our predictions. Fig. 11 presents our observations for the two application sets S_1 and S_2 . The figure shows that both the subsets can be operated at DVFS3 within the power limits. It also shows the performance degradation of the subsets (here, we just mention the average performance degradation of the two identical instances of TPC-W(20) in terms of their response times). Depending on the performance degradation of the subsets, the administrator may either choose S_1 or S_2 . We consider a few things worth mentioning about these results. First and most direct, we find them encouraging because: 1) packing for which our predictors indicated at least one type of budget violation were indeed found to result in a budget being exceeded (S_1 at both DVFS0 and DVFS3) and 2) packing for which it was indicated that there would not be violations (S_2 at both DVFS0 and DVFS3), in fact, operated safely.

Second, these results suggest that our profiling and prediction techniques are effective at comparing the power behavior of a variety of consolidation settings under different DVFS states. Techniques with such capabilities are likely to be of value in the power-aware platforms our research is concerned with.

7 RELATED WORK

While limitation on battery lifetime has been the main concern for mobile [17] and embedded systems, research on server systems [5] has mainly been focusing on reducing energy consumption and handling reliability constraints imposed due to electrical and cooling limits.

Reducing energy consumption. The tremendous increase in power consumption over the last few years is mainly attributed to the growth in the number of servers, with only a small percentage associated with increase in the power use per unit. In an attempt to reduce the number of active servers, mechanism to dynamically turns ON/OFF servers based on utilization was proposed [6], [7], [31]. While the above research looked at reducing the number of servers, Femal et al. suggested that overprovisioning servers may increase the performance of throughput-oriented applications without compromising on the power budget

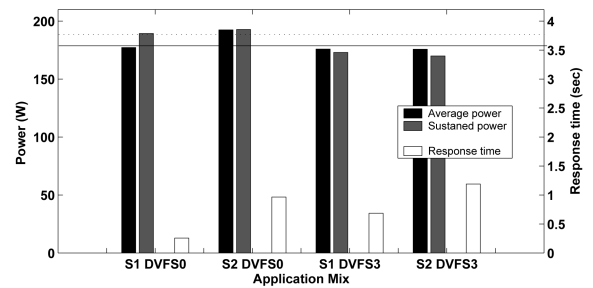


Fig. 11. Illustration of packing decisions made by our predicting technique involving three applications.

of the infrastructure [16]. Interplay of power and performance both in the arena of uniprocessors and multi-processor has been studied in great detail [1], [3], [39]. Chase et al. considered energy-aware resource provisioning in a data center in accordance to negotiated QoS agreements [7]. Stoess et al. [42] did accounting and capping of energy consumption for consolidated environments. Nathuji and Schwan looked at extending power management capabilities for the virtual machines [29]. We believe that the above techniques for energy management will greatly benefit from our average power prediction both in deciding on the energy budgets for the servers as well as on the placement of applications minimizing performance degradation.

Reliability concerns. Felter et al. proposed a technique that *reduces* the peak power demand on a server by dynamically distributing the power among the system components based on their requirement [15]. Lefurgy et al. recently presented a technique that uses system-level power measurement to *cap* the peak power consumption of the server while maintaining the system at the highest possible performance state [26]. Wang et al. extended the power capping mechanism to a cluster of servers [47]. Ensemble-level power management [33] observes that for real workloads, the possibility of peak power consumption occurring simultaneously is very low and uses this observation to reduce the cooling power capacity at a blade enclosure level by 20 percent. Fan et al. did extensive profiling of servers deployed in Google data centers and observed that the probability of synchronized peak power is very low only for very large collection of servers, but are relatively frequent for smaller subset of servers [14]. Recent work by Ramya et al. looked at coordinating multiple power management activities (average and peak) happening throughout the hierarchy of a data center [32]. Thermal reliability has extensively been looked at both server level and data center level including techniques to dynamically throttle or move applications upon reliability violations [8], [28], [20], [3]. Recent servers are currently being shipped with in-built capability to measure power at a very fine granularity. IBM's Active Energy Manager uses this capability to dynamically measure and control power consumption of a server [23]. Intel's Data Center Manager (DCM) is a programmable tool which provides power/thermal monitoring and management for Nehalem-based servers [25].

To the best of our knowledge, we are the first to develop a prediction technique to estimate the possibility of simultaneous peak power consumption for a set of consolidated applications. We believe that our prediction

techniques will greatly complement current research in deciding the right degree of consolidation keeping in mind the reliability limits of the infrastructure.

Modeling/characterization of power. Modeling of power consumption has been done at various granularity from a data center, server, and individual components to an instruction [13], [19], [48] either by using direct measurements or estimations from performance counters or a combination of both. We borrow ideas from the existing research correlating resource usage and power consumption to extend it to predict for consolidated setting. SPEC's ongoing effort, SPEC Power aims at characterizing the performance and power behavior of servers at different utilization [38]. To the best of our knowledge, we are the first ones to do such an extensive characterization of power consumption in a consolidated environment.

8 CONCLUSION AND FUTURE WORK

Our work was motivated by the need to ensure that emergent techniques for consolidating applications in enterprise-scale data centers exhibit robust and predictable power dissipation behavior. Consolidation of workloads has emerged as a key mechanism to dampen the rapidly growing energy expenditure within enterprise-scale data centers. However, before these consolidation-based techniques can be gainfully utilized, we must be able to predict and enforce appropriate limits on power consumption at various levels within the data center. In particular, two kinds of power budgets—average budgets defined over relatively coarse time scales and sustained budgets defined over short time scales—were found to be crucial to the safe and profitable operation of data centers.

Our research developed predictive models for the average and sustained power consumption of groups of applications colocated on a server as well as on a set of servers. We implemented our techniques on a Xen-based platform and evaluated them in a wide variety of consolidation settings. We demonstrated how these prediction techniques could allow us to achieve efficient yet safe system operation.

As part of our immediate future work, we plan to investigate on the design of resource schedulers within servers that can help enforce specified power budgets without arbitrarily degrading performance. In a well-designed system, these schedulers would complement the packing techniques by reacting to short-term/unanticipated fluctuations in the power usage behavior that could violate budgets. We plan to investigate how best to make our packing techniques work in tandem with such power-aware per-server resource schedulers.

REFERENCES

- [1] M. Annavaram, E. Grochowski, and J. Shen, "Mitigating Amdahl's Law through EPI Throttling," *Proc. Int'l Symp. Computer Architecture (ISCA)*, 2005.
- [2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the Art of Virtualization," *Proc. 19th ACM Symp. Operating Systems Principles*, pp. 164-177, 2003.
- [3] F. Bellosa, S. Kellner, M. Waitz, and A. Weissel, "Event-Driven Energy Accounting for Dynamic Thermal Management," *Proc. Workshop Compilers and Operating Systems for Low Power (COLP '03)*, 2003.
- [4] M. Benani and D. Menasce, "Resource Allocation for Autonomic Data Centers Using Analytic Performance Models," *Proc. IEEE Int'l Conf. Autonomic Computing (ICAC '05)*, June 2005.
- [5] R. Bianchini and R. Rajamony, "Power and Energy Management for Server Systems," *Computer*, vol. 37, no. 11, pp. 68-74, Nov. 2004.
- [6] P. Bohrer, D. Cohn, E. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, R. Rajamony, F. Rawson, and E.V. Hensbergen, "Energy Conservation for Servers," *Proc. IEEE Workshop Power Management for Real-Time and Embedded Systems*, May 2001.
- [7] J.S. Chase, D.C. Anderson, P.N. Thakar, A.M. Vahdat, and R.P. Doyle, "Managing Energy and Server Resources in Hosting Centers," *Proc. 18th ACM Symp. Operating Systems Principles (SOSP '01)*, 2001.
- [8] J. Choi, Y. Kim, A. Sivasubramaniam, J. Srebric, Q. Wang, and J. Lee, "Modeling and Managing Thermal Profiles of Rack-Mounted Servers with ThermoStat," *Proc. IEEE 13th Int'l Symp. High Performance Computer Architecture*, 2007.
- [9] I. Cohen, J. Chase, M. Goldszmidt, T. Kelly, and J. Symons, "Correlating Instrumentation Data to System States: A Building Block for Automated Diagnosis and Control," *Proc. Sixth USENIX Symp. in Operating Systems Design and Implementation (OSDI '04)*, Dec. 2004.
- [10] Commercial Circuit Breakers, http://circuit-breakers.carlingtech.com/all_circuits.asp, 2010.
- [11] Dell Computers: PowerEdge Servers SC1425 Spec Sheet, www.dell.com/downloads/global/products/pedge/en/sc1425_specs.pdf, Dec. 2005.
- [12] R. Doyle, J. Chase, O. Asad, W. Jin, and A. Vahdat, "Model-Based Resource Provisioning in a Web Service Utility," *Proc. Fourth USENIX Symp. Internet Technologies and Systems (USITS)*, Mar. 2003.
- [13] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan, "Full-System Power Analysis and Modeling for Server Environments," *Proc. Workshop Modeling, Benchmarking, and Simulation (MoBS)*, June 2006.
- [14] X. Fan, W.-D. Weber, and L.A. Barroso, "Power Provisioning for a Warehouse-Sized Computer," *Proc. 34th Ann. Int'l Symp. Computer Architecture (ISCA '07)*, 2007.
- [15] W. Felter, K. Rajamani, T. Keller, and C. Rusu, "A Performance-Conserving Approach for Reducing Peak Power Consumption in Server Systems," *Proc. Int'l Conf. Supercomputing (ICS)*, June 2005.
- [16] M.E. Femal and V.W. Freeh, "Safe Overprovisioning: Using Power Limits to Increase Aggregate Throughput," *Proc. Workshop Power-Aware Computer Systems (PACS '04)*, 2004.
- [17] J. Flinn and M. Satyanarayanan, "Managing Battery Lifetime with Energy-Aware Adaptation," *ACM Trans. Computer Systems*, vol. 22, no. 2 pp. 137-179, 2004.
- [18] S. Govindan, J. Choi, B. Urgaonkar, A. Sivasubramaniam, and A. Baldini, "Statistical Profiling-Based Techniques for Effective Power Provisioning in Data Centers," *Proc. ACM European Conf. Computer Systems (EuroSys '09)*, 2009.
- [19] S. Gurumurthi, A. Sivasubramaniam, M. Irwin, N. Vijaykrishnan, M. Kandemir, T. Li, and L. John, "Using Complete Machine Simulation for Software Power Estimation: The SoftWatt Approach," *Proc. Int'l Symp. High Performance Computer Architecture*, Feb. 2002.
- [20] T. Heath, A.P. Centeno, P. George, Y. Jaluria, and R. Bianchini, "Mercury and Freon: Temperature Emulation and Management in Server Systems," *Proc. Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Oct. 2006.
- [21] J. Hellerstein, F. Zhang, and P. Shahabuddin, "A Statistical Approach to Predictive Detection," *Computer Networks*, vol. 35, no. 1, pp. 77-95, Jan. 2001.
- [22] HP Power Manager, <http://h18013.www1.hp.com/products/servers/management/ilo/power-regulator.html>, 2010.
- [23] IBM Active Energy Manager—Measure and Cap Energy, <http://www-03.ibm.com/press/us/en/pressrelease/22551.wss>, 2010.
- [24] IBM Active Energy Manager, <http://www-03.ibm.com/press/us/en/pressrelease/22551.wss>, 2010.
- [25] Intel, Intel Data Center Manager, <http://software.intel.com/sites/datacentermanager/>, 2009.
- [26] C. Lefurgy, X. Wang, and M. Ware, "Server-Level Power Control," *Proc. Fourth Int'l Conf. Autonomic Computing (ICAC '07)*, p. 4, 2007.
- [27] A. Merkel and F. Bellosa, "Balancing Power Consumption in Multiprocessor Systems," *Proc. ACM SIGOPS European Conf. Computer Systems (EuroSys)*, Apr. 2006.

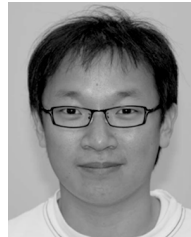
- [28] J. Moore, R. Sharma, R. Shih, J. Chase, C. Patel, and P. Ranganathan, "Going beyond CPUs: The Potential of Temperature-Aware Data Center Architectures," *Proc. First Workshop Temperature-Aware Computer Systems*, June 2004.
- [29] R. Nathuji and K. Schwan, "Virtualpower: Coordinated Power Management in Virtualized Enterprise Systems," *Proc. 21st ACM Symp. Operating Systems Principles (SOSP '07)*, 2007.
- [30] B. Pimentel Demand Grows, but Data Centers Don't Hog Power, San Francisco Chronicle, <http://www.sfgate.com/cgi-bin/article.cgi?f=/c/a/2007/02/15/BUGMAO4QLT1.DTL>, 2007.
- [31] E. Pinheiro, R. Bianchini, E. Carrera, and T. Heath, "Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems," *Proc. Workshop Compilers and Operating Systems for Low Power*, Sept. 2001.
- [32] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, "No Power Struggles: Coordinated Multi-Level Power Management for the Data Center," *Proc. 13th Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS '08)*, Mar. 2008.
- [33] P. Ranganathan, P. Leech, D. Irwin, and J. Chase, "Ensemble-Level Power Management for Dense Blade Servers," *Proc. Int'l Symp. Computer Architecture (ISCA)*, June 2006.
- [34] Raritan 20Amp PDU Model—DPCR 20-20, <http://www.raritan.com/products/power-management/Dominion-PX/DPCR20-20/>, May 2008.
- [35] Rubis, <http://rubis.objectweb.org>, 2010.
- [36] Signametrics Multimeter SM2040 Series Spec Sheet, <http://www.signametrics.com/products/sm2040.pdf>, May 2000.
- [37] SPEC CPU2000, <http://www.spec.org/cpu2000/>, 2010.
- [38] SPECpower, <http://www.spec.org/specpower/>, 2010.
- [39] M. Steinder, I. Whalley, J.E. Hanson, and J.O. Kephart, "Coordinated Management of Power Usage and Runtime Performance," *Proc. Network Operations and Management Symp. (NOMS '08)*, 2008.
- [40] C. Stewart, T. Kelly, and A. Zhang, "Exploiting Nonstationarity for Performance Prediction," *Proc. ACM European Conf. Computer Systems (EuroSys '07)*, Mar. 2007.
- [41] C. Stewart and K. Shen, "Performance Modeling and System Management for Multi-Component Online Services," *Proc. Second USENIX Symp. Networked Systems Design and Implementation (NSDI '05)*, pp. 71-84, May 2005.
- [42] J. Stoess, C. Klee, S. Domthera, and F. Bellosa, "Transparent, Power-Aware Migration in Virtualized Systems," *Proc. GI/ITG Fachgruppentreffen Betriebssysteme*, Oct. 2007.
- [43] J. Stoess, C. Lang, and F. Bellosa, "Energy Management for Hypervisor-Based Virtual Machines," *Proc. 2007 USENIX Technical Conf. (USENIX '07)*, June 2007.
- [44] TPC-W, www.tpc.org/tpcw, 2010.
- [45] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, "An Analytical Model for Multi-Tier Internet Services and Its Applications," *Proc. SIGMETRICS '05*, June 2005.
- [46] B. Urgaonkar, P. Shenoy, and T. Roscoe, "Resource Overbooking and Application Profiling in Shared Hosting Platforms," *Proc. Fifth USENIX Symp. Operating Systems Design and Implementation (OSDI '02)*, Dec. 2002.
- [47] X. Wang and M. Chen, "Cluster-Level Feedback Power Control for Performance Optimization," *Proc. 14th Int'l Symp. High-Performance Computer Architecture (HPCA '08)*, Feb. 2008.
- [48] W. Wu, L. Jin, J. Yang, P. Liu, and S.X.-D. Tan, "Efficient Power Modeling and Software Thermal Sensing for Runtime Temperature Monitoring," *ACM Trans. Design Automation of Electronic Systems*, vol. 12, no. 3, p. 26, 2007.



Jeonghwan Choi received the BS and PhD degrees in computer science in 1990 and 2007, respectively, and the MS degree in information and communication engineering in 1997 from Korea Advanced Institute of Science and Technology (KAIST). He has been a researcher at Sungkyunkwan University since 2009. He has several publications on leading journal and conferences. His research interests span the power management, system virtualization, and thermal management.



Sriram Govindan received the BTech degree in information technology from the College of Engineering, Guindy, Chennai, in 2005. He is currently working toward the PhD degree in computer science at The Pennsylvania State University. His research interests are in operating systems, server virtualization, and data center power management.



Jinkyu Jeong received the BS degree from the Computer Science Department, Yonsei University, and the MS degree in computer science from the Korea Institute of Science and Technology (KAIST), where he is currently working toward the PhD degree at the Computer Science Department. His current research interests include real-time system, operating systems, virtualization, and embedded systems.



Bhuvan Urgaonkar received the BTech (Honors) degree in computer science and engineering from the Indian Institute of Technology, Kharagpur, in 1999, and the PhD degree in computer science at the University of Massachusetts in 2005. He is an assistant professor in the Department of Computer Science and Engineering at the Pennsylvania State University. His research interests are in the modeling, implementation, and evaluation of distributed systems, operating systems, and storage systems, with a current focus on hybrid storage, cloud computing, and sustainable data centers.



Anand Sivasubramaniam received the BTech degree in computer science from the Indian Institute of Technology, Madras, in 1989, and the MS and PhD degrees in computer science from Georgia Institute of Technology in 1991 and 1995, respectively. Since fall 1995, he has been on the faculty of The Pennsylvania State University, where he is currently a professor. His research interests are in computer architecture, operating systems, performance evaluation, and applications for both high-performance computer systems and embedded systems. His research has been funded by the US National Science Foundation (NSF) through several grants, including the CAREER Award, and from industries including IBM, Microsoft, and Unisys Corp. He has several publications in leading journals and conferences, has been on the editorial board of the *IEEE Transactions on Parallel and Distributed Systems*, and the *IEEE Transactions on Computers*. He is a recipient of the 2002, 2004, and 2005 IBM Faculty Awards. He is a senior member of the IEEE and ACM.

► For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.