Profiling, Prediction, and Capping of Power Consumption in Consolidated Environments

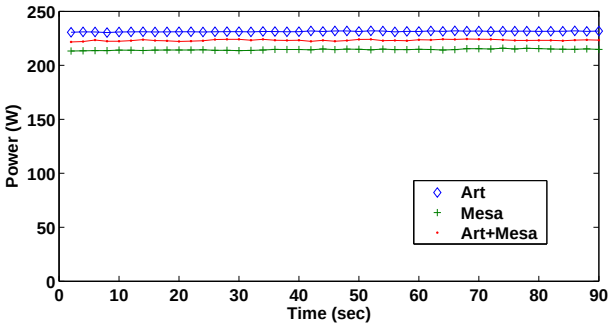
Jeonghwan Choi, Sriram Govindan, Bhuvan Urgaonkar and Anand Sivasubramaniam
Department of Computer Science and Engineering,
The Pennsylvania State University, University Park, PA 16802

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 co-located applications. Such characterization is crucial for effective prediction and enforcement of appropriate limits on power consumption—*power budgets or caps*—within the data center. Power caps need to be enforced at multiple spatial granularities within a data center: from server and rack to the room-level. Furthermore, power budgets must be also enforced at multiple temporal granularities: from durations of less than a second (dictated by fuses for reliability concerns) to longer periods of several minutes to hours (relevant to energy optimization considerations.) We capture these requirements in the form of two kinds of power budgets at each spatial level: (i) an *average budget* to capture an upper bound on long-term energy consumption within that level and (ii) 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 consumptions within 5% error-margin. Our sustained power prediction techniques allow us to predict close yet safe upper bounds on the sustained power consumption of consolidated applications.

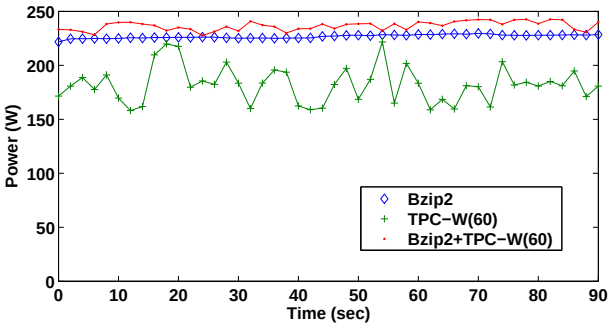
I. Introduction

A. Motivation

To accommodate modern resource-intensive high-performance applications, large-scale computing and storage platforms 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 high-performance computing platforms (or data centers) accounts for 1.2% of the overall electricity consumption in the U.S. More alarmingly, if current practices for the design and operation of these platforms continue, their power consumption is projected to keep growing at 18% every year. These observations have spurred great interest among providers of high-end computing platforms 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], [25]. Another major concern for such large-scale computing platforms 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 affecting the reliability concerns of these platforms. This has been addressed in literature by reducing the peak power consumption both at the server level [14] as well



(a) Two CPU-saturating applications: Art and Mesa



(b) A CPU-saturating (Bzip2) and a non CPU-saturating (TPC-W) application

Fig. 1. Measured power consumption behaves differently when CPU-saturating applications are co-located from when CPU-saturating and non-CPU-saturating applications are co-located. The graph provides a representative (median) sample for every 2 second interval. TPC-W is running 60 simultaneous sessions.

as at the cluster level [27]. Consolidation further increases the power density of the servers, aggravating the reliability concerns of the facility.

Literature has addressed the energy and reliability related concerns in a data center using the notion of power budgets [14], [27], [26]. Power budget is typically enforced at different hierarchies of a data center and it specifies a cap on the power consumption of applications consolidated under that hierarchy. Previous work looked at mechanisms to enforce power budgets both at the server level [21] and at the cluster level [40]. Typically power budget violations are handled either by throttling [40], [21] or by migrating [22], [36] the applications. These power budgets are ignorant of the power requirement of the applications that are consolidated and therefore could result either in poor performance or less efficient enforcement. Solutions for consolidation in enterprise-scale systems have benefited from detailed studies of the workloads and resource needs (like CPU) of applications [39]. Insights gained from these studies have been utilized to build models for predicting the performance and resource usage behavior of consolidated applications. [1] 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 [31].

Consolidation may occur at multiple spatial granularities, ranging from co-location 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 sub-second 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

[1]This research is commonly referred to as application modeling [38].

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 results in 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, shown in Figure 1. In each case, we compare the power consumptions of individual applications with that when they were co-located. Power consumption was sampled once every 2 msec, and we report the median sample over successive 2 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 [29]) are co-located (Figure 1(a)), they consume what appears to be an average of their individual power consumptions. When a CPU-saturating application (Bzip2, also from the SPEC CPU2000 suite) is co-located with a non-CPU-saturating application (TPC-W [37], an E-Commerce benchmark that spends significant time blocked on I/O), the power consumed appears to exceed their individual consumptions (Figure 1(b)). Roughly speaking, this can be explained as follows. In Figure 1(a), the two applications, both whose power consumption was attributable almost entirely to the CPU, share this resource. On the other hand, in Figure 1(b), 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 (b) over (a), from power consumption perspective (a) may be considered better than (b).

More generally, both average and sustained power consumption in a consolidated environment depend in non-trivial 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.

B. 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: (i) identifying crucial dependencies between the power consumption of individual applications and their resource usage patterns and (ii) 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 5% error-margin. Our sustained power prediction techniques provide close yet safe upper bounds on the sustained power consumption of consolidated applications. Finally, we demonstrate how these techniques could be employed to improve system utilization without compromising the safety of its operation.

C. Road-map

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

II. Background

A. Power Consumption in Data Centers

In a typical data center, a primary switch board 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: (a) at the lowest level, multiple applications are consolidated on a physical server, (b) 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. The CPU is by far the largest consumer of power within a server [6]. 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.

B. 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 towards maintenance, operational power, and cooling. [2] Regardless of the nuances of this decision-making, it

[2]Examples of such formulations may be found in several papers on resource management in data centers. Ideas for incorporating power costs into such dynamic optimization, however, are less common, and seem to be still evolving .

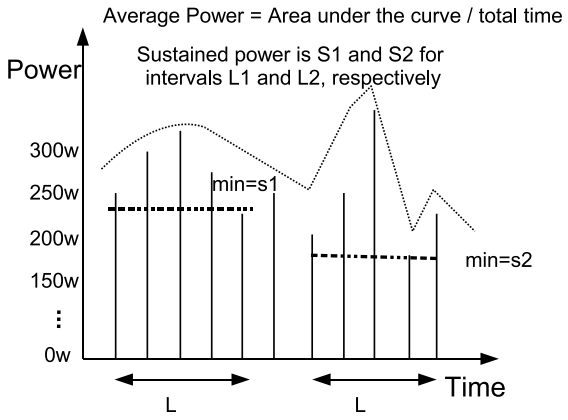


Fig. 2. Illustration of average and sustained power consumptions.

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 fuses [3] or circuit breakers associated with that component. In literature, the phrase *peak power* is sometimes used for this aspect of power usage [14]. As we mentioned earlier, sustained power budgets are defined over a period of few secs or msecs. A sustained power budget is represented by a tuple (S, L), which specifies a bound on the maximum power S that could be sustained over any interval of length L. This tuple could be obtained from the time-current characteristics curve of a fuse. Typically this curve is composed of multiple such tuples but for simplicity, in this paper, we just refer to a single tuple. Figure 2 shows a hypothetical power series to illustrate the difference between average and sustained power. Average power is obtained by taking the average of the power samples in the entire time interval (energy/time). Sustained power for any interval of length L corresponds to the *minimum* power consumed during that interval. This is because sustained power is the maximum power that was sustained throughout that interval which is nothing but the minimum power of that interval.

III. 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.

A. 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 [4], similar to those existing in current research [39]. The profiling technique involves

[3]Fuse is a metal wire that melts when heated by a prescribed current, opening the underlying circuit and thereby protecting the circuit from over-current situation. Typically a fuse is defined by its time-current characteristic curve which shows the time required to melt the fuse for any given level of overload current. Circuit breaker is similar to fuse in its function except that it could be reused after a over-current situation.

[4]Offline profiling is not unreasonable to assume, in fact, data-center applications do start in isolation. In addition, the usage patterns are repetitive over ”some” time granularity. Consequently, profiling can be done during the early stages of application deployment

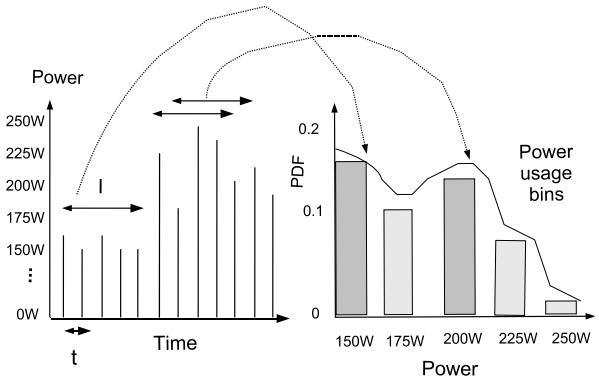


Fig. 3. Illustration of the derivation of power usage distribution from a power profile for $t_p = t$ and $I_p = I$.

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. [5] 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. It is important to emphasize that the workload used during profiling should be both realistic and representative of real-world workloads. 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 trace replay of actual workloads to running the application in a "live" setting, and from the use of synthetic workload generators to the use of well-known benchmarks. Any such technique suffices for our purpose as long as it realistically emulates real-world conditions.

Profiling power consumption: We connect a multi-meter to the server used for our offline profiling and use it to measure the power consumption of the server once every t_p time units. We refer to the resulting time-series of (instantaneous) power consumption samples as the *power profile* of the application. Note that due to our ability to only measure power usage at the granularity of the entire server (as opposed to measuring the power usage of constituent components such as CPU, disk, etc.), our measurements are only an approximation (in fact, an upper bound) of the power consumed by the application. By minimizing any other interfering activities during the offline profiling, we attempt to keep this gap small. 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 integers.) 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. Figure 3 illustrates the process of converting a power profile into a power usage distribution. Notice 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 (approx. 156W for our server).

Profiling resource usage: We use measurement techniques similar to those existing in research [39] to record resource scheduling events of interest. By recording CPU scheduling/de-scheduling 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

[5] In practice, a distributed application with multiple components may require multiple servers to meet its resource needs. We only consider applications whose resource needs can be met by a single server. Our technique easily extends to applications requiring multiple servers by simply running the application on the appropriate number of servers and conducting measurements on each of them.

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, throughput.)

B. 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 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 [39], [34]. Furthermore, workload patterns are often repetitive over some time granularity (such as daily cycles [19]), 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 [19]. 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*: Note that due to our ability to only measure power usage at the granularity of the entire server (as opposed to measuring the power usage of constituent components such as CPU, disk, etc.), our measurements are only an approximation (in fact, an upper bound) of the power consumed by the application. By minimizing any other interfering activities during the offline profiling, we attempt to keep this gap small.
- *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 [33], [4], [9], [11], [38]. 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.

C. 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: (a) how such profiling should be done, (b) the relationship between an application's power consumption and its usage of various resources, and (c) the extent of variability in the power consumption of these applications.

Our testbed consists of several Dell PowerEdge servers (details appear in Table I.) We use one of these servers for running the applications that we profile. We connect a Signametrics SM2040 multi-meter

Dell PowerEdge SC1450 Features [10]	
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

TABLE I
SPECIFICATIONS OF THE SERVER USED FOR PROFILING.

Signametrics SM2040 Features [28]	
Digits of Resolution	6-1/2
Measurement Rates	0.2/sec - 1000/sec
Measurement Range (AC current)	2.5A
Interface	PCI

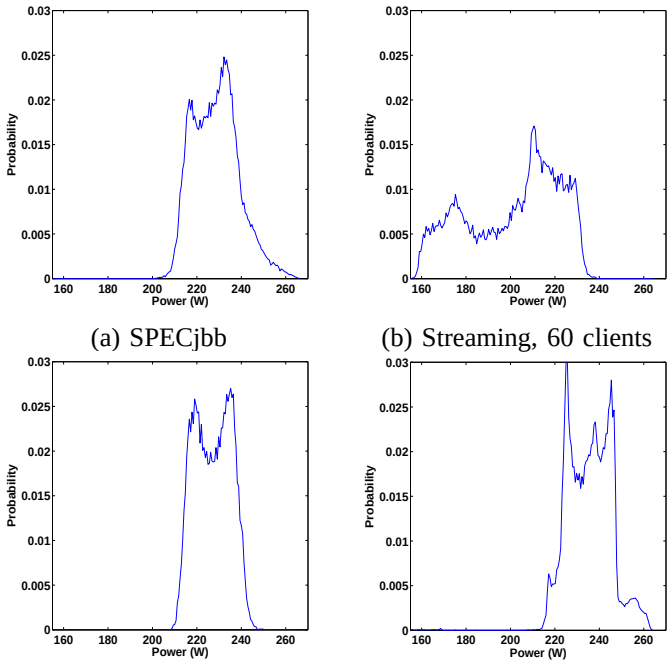
TABLE II
DETAILS OF THE MULTI-METER USED IN OUR PROFILING.

(details appear in Table II) in series to the power supply of this server. The multi-meter sits on the PCI bus of another server which is solely used for logging purposes. This multi-meter is capable of recording power consumption as frequently as once 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 be present when are consolidated.) The server running the application is connected to the multi-meter 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 non-essential 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 III. 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 SPECCPU suite are *CPU-saturating*, in that they are ready to use the CPU at all times. The remaining applications 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 *non-CPU-saturating* applications at different workload intensities. TPC-W is profiled with the 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.

Applications	
TPC-W [37]	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
SPECjbb2005 [30]	SPEC's 3-tiered client-server benchmark emulating server-side java applications
SPEC CPU2000 [29]	SPEC CPU2000 suite (Art, Bzip2, Mcf, Mesa)

TABLE III
SALIENT PROPERTIES OF OUR APPLICATIONS. TPC-W, STREAMING, AND SPECJBB ARE NON CPU-SATURATING, WHEREAS APPLICATIONS IN THE SPEC CPU2000 SUITE ARE CPU-SATURATING.



(a) SPECjbb (b) Streaming, 60 clients

(c) Bzip2 (d) Mcf

Fig. 4. Power distributions of SPECjbb, Streaming, Bzip2, and Mcf compared.

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 Figure 4.

Given that CPU consumes significantly more power than I/O devices in our environment, not surprisingly, power distributions for non CPU-saturating applications (Figures 4 (a) and (b)) are found to exhibit higher variance than CPU-saturating applications (Figures 4 (c) and (d).) 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 do not exhibit multiple phases with respect to power consumption that were mentioned in Section III-B.) 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 a 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.

Power state	CPU Utilization				Normalized Performace 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 IV
CPU USAGE OF TPC-W(60) OPERATING AT THREE DIFFERENT DVFS STATES.

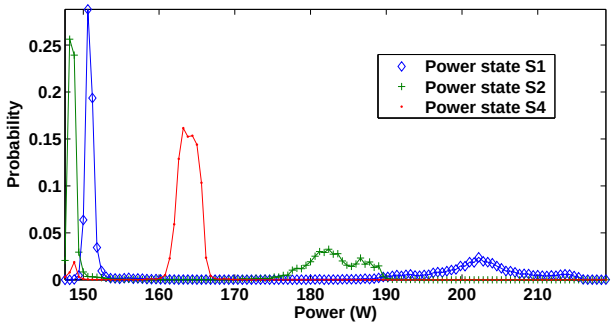


Fig. 5. Power distribution of TCP-W(60) collected at different CPU power states.

Next, we study the impact of changing CPU power states (DVFS and clock throttling) on the power consumption of 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: (3,4GHz,100%), S2: (3.2GHz,100%), S3: (3.0GHz,100%) and S4: (2.8GHz,50%). As a representative result, Figure 5 shows the power PDF of TPC-W(60) when it was run with the CPU at three different CPU power states; Table IV 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. becomes less bursty at lower power states.

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

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 V
IMPACT OF DVFS STATES ON POWER CONSUMPTION AND PERFORMANCE OF APPLICATIONS.

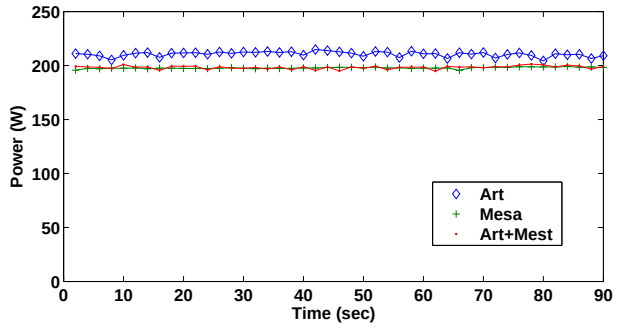
Finally, we observe the trade-off between power consumption and application performance as the CPU state is varied. Table V presents this comparison for a CPU-saturating application (Bzip2) and non CPU-saturating application (TPC-W(60)). The performance metric for Bzip2 was program completion time; the metric for TPC-W was average response time. As seen in Table V and Table IV, changes in average power usage are more pronounced for CPU-saturating application, reduction of 50W (between the two extreme power states in the table) for Bzip2, compared with a reduction of only 15W 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 non-trivial 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 it's very high CPU utilization (92% - 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.

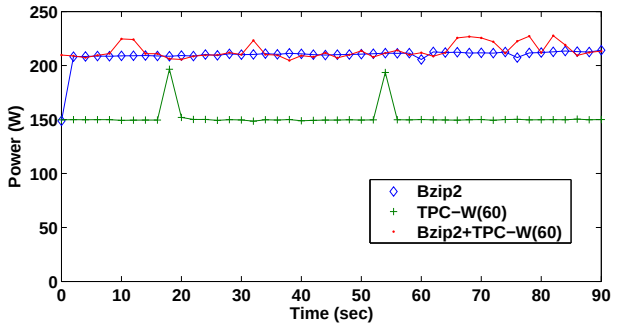
Observation 3: Power-performance trade-offs when operating at different CPU power states differ significantly for CPU-saturating and non-CPU-saturating applications. While it is easy to predict for CPU-saturating applications, it can depend in non-trivial ways for non-CPU-saturating applications.

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	

TABLE VI
AVERAGE POWER CONSUMPTION OF CPU-SATURATING APPLICATIONS AND NON-CPU-SATURATING APPLICATIONS BEHAVE DIFFERENTLY.



(a) Two CPU-saturating applications, Art and Mesa



(b) A CPU-saturating, Bzip2 and a non CPU-saturating, TPC-W(60) application

Fig. 6. Sustained power when CPU-saturating applications are co-located behaves differently from when CPU-saturating and non-CPU-saturating applications are co-located. We report the minimum power consumption over windows of length 2 sec each.

D. Power Profiles of Consolidated Applications: Experimental Results

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

We had seen in Figure 1 the server power consumption in two different consolidation scenarios, each co-locating a pair of applications. In Table(VI), we present the observed average power consumption and individual power consumption of two sets of consolidation. The power consumed when two CPU-saturating applications (Art and Mesa) are co-located 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 co-located 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 co-locating CPU-saturating applications versus when co-locating CPU-saturating and non-CPU-saturating applications.

Similar observations apply to the behavior of sustained power consumed by a set of co-located applications. Figure 6 plots the minimum power usage seen during non-overlapping intervals of length 2 sec each

for two consolidation settings: (a) when CPU-saturating applications are co-located and (b) when a CPU-saturating and a non CPU-saturating application are co-located. The minimum power usage during each interval is its sustained power consumption. In each case, we present the sustained 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.

IV. Average Power Prediction

In this section we firstdevelop techniques to predict the average power consumption of a server and then extend it to predict average power consumption across a set of servers.

A. Average Power Prediction for a server

In this section, we develop techniques to predict the average power consumption of a server which consolidates *'n'* applications. Input for these set of algorithms is the power and resource usage distributions of the individual applications. We provide experimental results to illustrate the efficacy of these techniques.

1) Baseline Prediction: We consider scenarios where applications are consolidated on the same CPU on our dual-processor server (see Table I.) Note that in these cases, the unused CPU continues to consume idle (leakage) power. We begin with the following simple *baseline* predictor for average power consumption of a server on which n applications $A_1, \cdots, A_n$ are consolidated. Our predictor employs a sum of the average power consumptions of the individual applications, weighted by their respective CPU utilization,

$$\bar{P}_{A_1,\cdots,A_n} = \begin{cases} \sum_{i=1}^{n} (\bar{P}_{A_i} \cdot R_{A_i}^{cpu}) & \text{//when CPU busy} \\ + & \\ \bar{P}^{idle} \cdot (1 - \sum_{i=1}^{n} R_{A_i}^{cpu}) & \text{//when CPU idle} \end{cases} \quad (1)$$

where $\bar{P}_{A_i}$ is the average of the power distribution of the application A_i (obtained from the offline profile); and $R_{A_i}^{cpu}$ ($0 \leq R_{A_i}^{cpu} \leq 1$) is the CPU allocation for it. [6] 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 VII the efficacy of baseline prediction in three consolidation settings, each co-locating a pair of applications.

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

TABLE VII
BASELINE PREDICTOR OF AVERAGE POWER CONSUMPTION.

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. When Art and TPC-W(60) are co-located, the baseline approach assumes that TPC-W(60) would utilize 40% of the CPU (as seen from its offline profile), while the CPU-saturating Art would consume the remaining 60%. Finally, when TPC-W(60) and TPC-W(10) were consolidated, the baseline approach uses a CPU utilization of 40% and 7% for these copies, respectively. Note that our prediction technique uses the utilization of the individual applications from their offline profile to predict their utilization in the consolidated setting.

[6] 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 called application modeling [38].

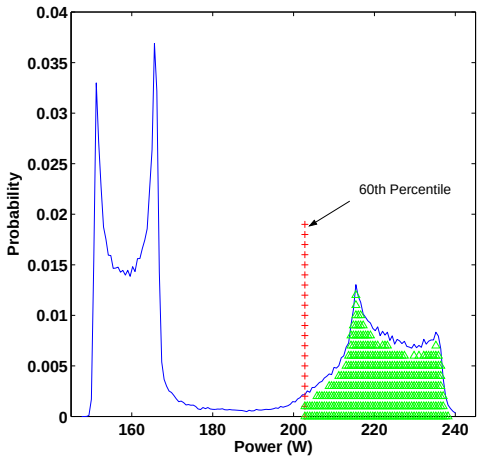


Fig. 7. Capturing the non-idle power portion for TPC-W(60).

Even though this simple utilization prediction may not work for all kinds of applications, it works with reasonable accuracy for our applications.

In the latter two consolidation settings, where we have non-CPU-saturating applications, we observe increasing error margins in the baseline prediction. There are two main reasons for these inaccuracies. First, the quantity $\bar{P}_{A_i}$ represents the average of the *entire* power distribution for application A_i including the durations when the CPU was idle. Non-CPU-saturating applications can have significant such idle periods. Upon consolidation, however, these idle durations are likely to be occupied by other co-located applications. Therefore, we need to employ the average power consumption by the application *only* over durations when it was using the CPU. The second cause for inaccuracies is that the baseline predictor does not accurately account for the power consumption due to all components other than the CPUs. Even though the power contributed by these I/O components are included in $\bar{P}_{A_i}$, they are assumed to be totally in proportion to the CPU (this may even be true for most of the applications, but at this point we do not know if this was a reasonable assumption to make). In particular, the I/O-intensive TPC-W application uses the disk and NIC on the server, which contribute to the power consumption besides the CPUs. In the rest of this section, we will enhance our baseline predictor to address these two sources of inaccuracies.

2) Improving Prediction of Average Power:

Improved Estimate of Active 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 (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. Figure 7 shows this for TPC-W(60) whose CPU utilization was 40%. We denote this quantity by $P_{A_i}^{busy}$ and replace P_{A_i} in Eq. 1 with it. In this example, P_{tpcw}^{busy} was found to be 225W, while P_{tpcw} was 185W – 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 (a) Art and TPC-W(60) and (b) TPC-W(60) and TPC-W(10) (Table VII.) With this enhancement, our estimates of average power improved from 209W to 226W for (a), and from 167W to 188W for (b) which reduces the error margin to 1.76% and 1.3% respectively.

Note that our enhancement to separate out the power consumption when the CPU was busy, in fact, (partly) also includes power consumed by I/O devices. This is because the I/O devices can be busy

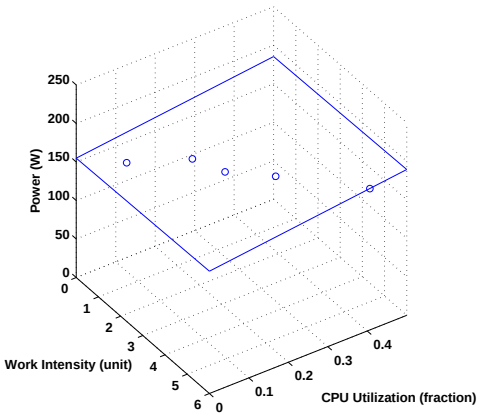


Fig. 8. Multiple regression for estimating average active CPU and I/O power. The maximum of the absolute value of the deviation of the data from the model is 0.45 and this is sufficiently small to be confident the model reasonably fits the data.

simultaneously with the CPU. Fortunately, in our setting the wattage of I/O devices was substantially smaller than the CPU power consumption (e.g., our disk has a maximum rating of only 7W), rendering our enhancement a reasonable predictor. In general, however, it is desirable to also separate out the contribution of I/O devices.

Incorporating I/O Power: As already described in Section II-A, in this research we restrict ourselves to the dissection of power consumed by an application into three components: idle CPU power, active CPU power, and I/O power.

We further refine our representation of average power consumed by an application by explicitly incorporating power consumed on its behalf by I/O devices. It is certainly possible to employ similar techniques to further partition I/O power into that due to individual I/O devices (e.g., breaking down I/O power into that consumed by disk and NIC), but we find this unnecessary in our environment. We introduce an additional component $P_A^{i/o}$ to represent the I/O portion of the average power consumed by an application.

To estimate this parameter $P_A^{i/o}$, we enhance our offline profiling in the following way. We subject the application under investigation to *multiple* workload intensities and formulate equations of the following form based on the average power usage and CPU, I/O utilization measurements from these.

$$\bar{P}_A = \bar{P}_A^{cpu/busy} \cdot U_A^{cpu} + \bar{P}^{idle} \cdot (1 - U_A^{cpu}) + \bar{P}_A^{i/o} \cdot U_A^{i/o} \tag{2}$$

where $\bar{P}_A^{cpu/busy}$, $\bar{P}^{idle}$, $\bar{P}_A^{i/o}$ represent averages of active CPU power, idle CPU power, and I/O power, respectively. Unlike U_A^{cpu}, $U_A^{i/o}$ is hard to define since: (i) it represents multiple I/O devices and (ii) utilization of I/O devices are not necessarily directly proportional to their throughput. We make the following simple approximation. We consider the combined data transfer rate (recorded during the profiling) for disk and NIC induced by the application as proportional to this utilization. The maximum data transfer rate we see in our profiling experiments is taken to correspond to a utilization of 1(other utilizations are normalized to this).

We then employ multiple regression to estimate the two unknowns $\bar{P}_A^{cpu/busy}$ and $\bar{P}_A^{i/o}$. This is shown in Figure 8. We subject TPC-W to different workload intensities ranging from 10 to 60 (Recall from Section III, these represents the workload intensity of the TPC-W benchmark - in the graph, we normalize these I/O intensities to 10). The measured power varied from 159.7W to 185.6W over this workload range. The active CPU power and I/O power are estimated to be 255W and 0.6W.

3) Component-aware Average Power Prediction: With the break-up of average power among the components described above, our predictor operates as follows for a server that consolidates applications

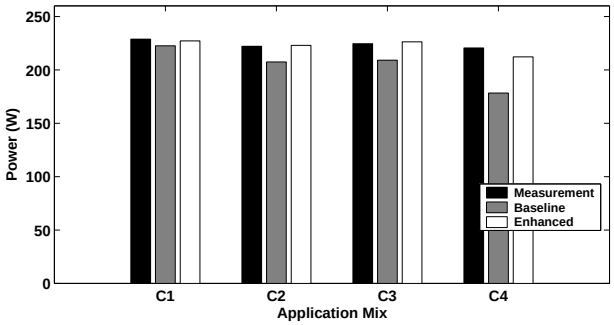


Fig. 9. Comparison of our predictors for a variety of consolidation scenarios. C1:TPC-W(10)+Art, C2:TPC-W(30)+Mesa, C3:TPC-W(60)+Bzip2 C4:TPC-W(60)+TPC-W(60).

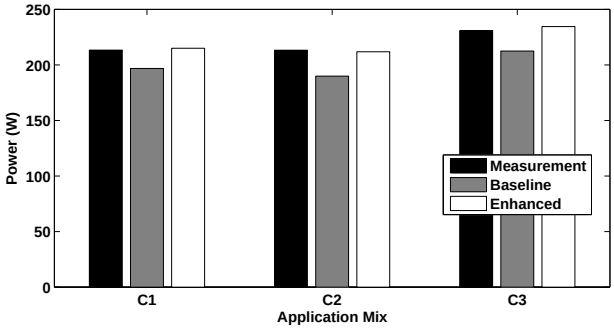


Fig. 10. Comparison of our predictors for a variety of consolidation scenarios. In C1 and C2, the applications are consolidated on the same physical processor and the second processor is idle. In C3, two applications each are consolidated on each processor. C1: TPC-W(60)+Bzip2+SM(100) C2: TPC-W(60)+Bzip2+SM(100)+SM(100) C3: Processor1:TPC-W(60)+SM(100), Processor2:SM(100)+Bzip2. SM(x) represents the streaming server with x clients.

$A_1, \cdots, A_n$,

$$\bar{P}_{A_1,\cdots,A_n} = \begin{cases} \sum_{i=1}^{n} (\bar{P}_{A_i}^{busy/cpu} \cdot R_{A_i}^{cpu}) & \text{//when CPU busy} \\ + & \\ \bar{P}^{idle} \cdot (1 - \sum_{i=1}^{n} R_{A_i}^{cpu}) & \text{//when CPU idle} \\ + & \\ \sum_{i=1}^{n} (\bar{P}_{A_i}^{i/o} \cdot R_{A_i}^{i/o}) & \text{//I/O power} \end{cases} \quad (3)$$

As two specific examples of the efficacy of these enhancements, our prediction of the average power consumed by Art and TPC-W(60) improved to 226W (within 0.8% of observed power); the prediction for TPC-W(60) and TPC-W(60) improved to 189 (within 0.5% of observed power.)

B. Average Power Prediction across multiple servers

Our prediction mechanism can easily be extended to a set of servers or a multi-processor system. For a system with *k* servers and each server with *l* processors hosting applications $A_1, \cdots, A_n$, the average power consumption is given by,

$$\bar{P}_{A_1,\cdots,A_n} = \begin{cases} \sum_{i=1}^{n} (\bar{P}_{A_i}^{busy/cpu} \cdot R_{A_i}^{cpu}) & \text{//CPU busy} \\ + & \\ \bar{P}^{idle} \cdot (k * l - \sum_{i=1}^{n} R_{A_i}^{cpu}) & \text{//CPU idle} \\ + & \\ \sum_{i=1}^{n} (\bar{P}_{A_i}^{i/o} \cdot R_{A_i}^{i/o}) & \text{//I/O power} \end{cases} \quad (4)$$

In Figure 9 and 10, we present the performance of our baseline and enhanced predictors for a variety of consolidation scenarios. Since we do not have an infrastructure to measure power across a set of servers, we evaluate our prediction technique by running applications on both the processors of our server. This is analogous to predicting average power consumption of a PDU which is connected to two servers. Our enhanced prediction is able to predict within 5% (on an average 1%) of the observed power while baseline approach has up to 20% (on an average 9%) error margin.

C. Average power - Discussion and Lessons Learned

Our study provides the following useful hints on predicting average power. It is important to separate out power expended when the application was not using the CPU from the off-line power distribution of an application. This is easily achieved if we collect CPU usage profile along with the power profile. It is also important to separate out the consumption of resources other than the CPU. While non-CPU devices contribute only a small amount to the overall power in our environment, including them in the analysis may be crucial in platforms where they are more significant contributors. Even in our environment, we were able to improve our prediction by incorporating estimates of I/O power in our calculations.

Based on our empirical evaluation, we conclude that it is possible to predict average power with reasonable accuracy using simple profiling and prediction techniques. We expect our technique to be easily extended for settings with other consumers of power such as graphic cards, storage over networks, networking equipment, etc.

V. Sustained Power Prediction

Next, we turn our attention to sustained power budgets in consolidated servers. Recall that sustained power budgets are enforced at different hierarchies of a data center to ensure that the applications consolidated under that level do not cause the capacity of the corresponding PDU to be exceeded. of the PDU deployed at that hierarchy In Section V-A we develop mechanisms to predict 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 [13], 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 over-provisioning present in existing power supplies, recent research [21] 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 V-B we extend our approach to predict sustained power violation at the PDU or rack level (across a set of servers.)

A. Sustained Power Prediction for a server

Recall that our goal is to predict the probability $Pr_{A_1,\cdots,A_n}(S, L)$, upon consolidating n applications $A_1, \cdots, A_n$ on a server, of S or more units of power being consumed for any L consecutive time units. We will refer to $Pr_{\_}$ 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 (as defined in Section II-A)—if this does not hold, the server would be inoperable. Recall from Section IV-A3 that 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.

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,

$$Pr_{A_1,\cdots,A_n}(S, L) = \prod_{i=1}^{n} \{Pr(w_{A_i}^{t_p} \geq S)\}^{m_i} \quad (5)$$

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 co-located 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 does not exhibit large temporal variation in its power consumption. Recall our observation from Section III that the CPU-saturating Mcf application 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 co-located 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 co-located 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 co-located applications resulting in a different CPU allocation than specified by $R_{A_i}^{cpu}$ for the applications).

Notice that (A) necessarily under-estimates and (B) necessarily over-estimates the probability of violation. (C), however, can cause errors in either direction. In the rest of this section, we will address these three shortcomings.

2) Improving Prediction of Sustained Power:

Addressing shortcoming (A): We had encountered a similar problem when dealing with reduction 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 co-located applications. Exactly as in Section IV-A2, 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. Figure 11 presents the power distribution of the TPC-W application before and after removing the idle-power portion.

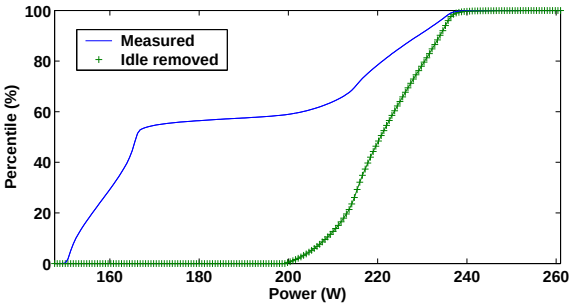


Fig. 11. Removing idle portion rom TPCW time series.

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. For each such window, we find the maximum sustained power throughout that duration by taking the *minimum* power sample within that window(as explained in Section II-B). 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 Eq. 5),

$$Pr_{A_1,\cdots,A_n}(S, L) = \prod_{i=1}^{n} \{Pr(w_{A_i}^{T_{m_i}} \geq S)\} \tag{6}$$

We provide an example to illustrate the seriousness of this problem, For TPC-W, the probability of violating a budget of (200W, 2ms) as obtained from TPC-W power profile collected at 2ms is 99.64%. The probability of violating 200W for a period of 1 second would be approximated as $(.9964)^{500}$ which is 16.47% But when we compute the actual sustained power by moving window of size 1 sec over TPC-W time series, we find this probability to be 21.55%.

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 their CPU allocations ($R_{A_i}^{cpu}$ for application A_i). We attempt to alleviate this problem by incorporating into our prediction the possibility of 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 profile collected over duration of length L (from offline profile). Let $Pr\{U^L_{(A_1,\cdots,A_n)} = (c_1, \cdots, c_n)\}$ be the probability that $(c_1, \cdots, c_n)$, are the fractional CPU consumption of applications $(A_1, \cdots, A_n)$, respectively, over all intervals of length L in our consolidated setting. We estimate this for all possible combination of $(c_1, \cdots, c_n)$ as follows.

The CPU utilization of the applications in the consolidated setting depends mainly on two things: (a) The CPU reservation of applications in the consolidated setting (b) The CPU requirement of the applications (both over periods of length L). When the reservation is higher than the requirement, then it means 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) divides 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 usages (CPU requirement) for every application over durations of length L. This distribution can be easily derived from the CPU usage profiles of the applications. Let $Pr(U^L_{A_i} \geq c)$ represent the probability that the

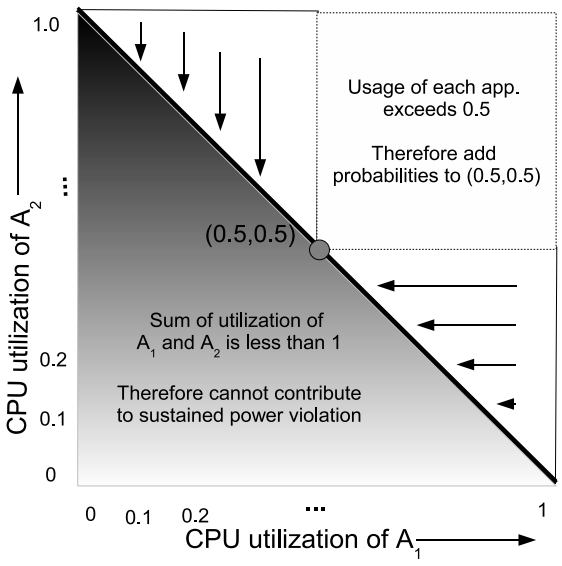


Fig. 12. Estimation of the fractional CPU usages of the applications A_1 and A_2 over durations of length L upon consolidation on a single server: size of the utilization bins $\delta = 0.01$ for this experiment.

CPU utilization of application A_i over duration of length L exceeds $c(0 \leq c \leq 1)$. We construct CPU requirement bins (RB) for every application over bins of length δ, $RB_{A_i}[\delta, 2\delta, 3\delta, \cdots, (1-\delta), 1]$, where $RB_{A_i}[j] = Pr(U^L_{A_i} \geq (j-\delta)) - Pr(U^L_{A_i} \geq j)$, that is each bin represents the probability of utilization being within its bin boundaries.

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 $Pr(U^L_{(A_1,A_2)} = (c_1, c_2))$ (which is the probability that CPU utilization of application A_1 is c_1 and that of A_2 is c_2) for $c_1 = \delta, 2\delta, \cdots 1$ and $c_2 = \delta, 2\delta, \cdots 1$. It could be obtained by,

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

where $(RB_{A_2}[\delta, \cdots, 1])'$ represents the transpose of the array $RB_{A_2}[\delta, \cdots, 1]$. Multiplying these two one dimensional matrices will generate a two dimensional matrix which provides the probability for all utilization pairs (c_1, c_2). [7] Note that in $Pr\{U^L_{(A_1,A_2)} = (c_1, c_2)\}$, ($c_1$+$c_2$) ranges from 0 to 2. But in a consolidated setting, utilization of the CPU cannot go beyond 100%. Let r_1 and r_2 be the reservations for the application A_1 and A_2 respectively. We estimate the utilization of the applications in the consolidated setting as follows,

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: $Pr(U^L_{(A_1,A_2)} = (r_1, r_2)) = Pr(U^L_{(A_1,A_2)} = (r_1, r_2)) + Pr(U^L_{(A_1,A_2)} = (c_1, c_2))$
4: **else if** $(c_1 > r_1) and (c_2 < r_2)$ **then**
5: $Pr(U^L_{(A_1,A_2)} = (1-c_2, c2)) = Pr(U^L_{(A_1,A_2)} = (1-c_2, c2)) + Pr(U^L_{(A_1,A_2)} = (c_1, c_2))$
6: **else if** $(c_1 < r_1) and (c_2 > r_2)$ **then**
7: $Pr(U^L_{(A_1,A_2)} = (c1, 1-c_1)) = Pr(U^L_{(A_1,A_2)} = (c1, 1-c_1)) + Pr(U^L_{(A_1,A_2)} = (c_1, c_2))$
8: **end if**
9: **end for**

This algorithm is graphically illustrated in the Figure 12. The figure assumes $r_1 = 0.5$ and $r_2 = 0.5$. Lines 2 and 3 handles the case when the (fractional) CPU usages of both the application is above their

[7] This extends easily to n applications; we omit these details here.

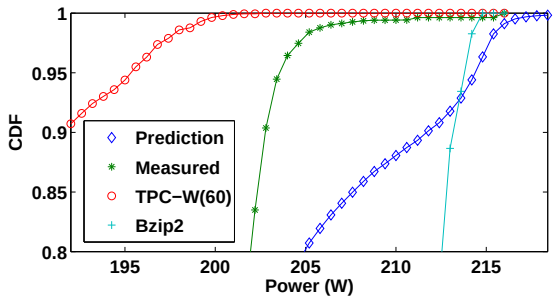


Fig. 13. Comparison of measured and predicted sustained power consumption (L=1 sec) for a server consolidating Bzip2 and TPC-W with reservations 50 and 50 respectively. For clarity, we just present the tail of these CDFs.

reservations (shown in the top right corner box of the figure), in this case the CPU usages of the applications in the consolidated setting would be (r_1, r_2). Lines 4 through 7 handles the case when the CPU requirement of one application is below its reservation and that of the other application is above its reservation, in which case, the needy application gets CPU from the surplus of the other (we add this probability as shown by the arrows in the figure). Also notice 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, $Pr_{A_1,A_2}(S, L) = Pr(w^{c_1}_{A_1} \geq S) * Pr(w^{c_2}_{A_2} \geq S)$ (similar to what was described in Eq. 6.) 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+δ; j=j-δ **do**
3: $Pr_{A_1,A_2}(S, L) = Pr_{A_1,A_2}(S, L) + Pr(w^i_{A_1} \geq S) \cdot Pr(w^j_{A_2} \geq S) \cdot Pr\{U^L_{(A_1,A_2)} = (i, j)\}$
4: **end for**

Line 2 of the above algorithm loops through all possible (c_1, c_2) pairs that adds 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. We run the above algorithm with L=1 sec and varying S from 0 to 300W for our experiments. Figure 13 compares our predictions with the observed sustained power for a server consolidating TPC-W and Bzip2. The graph also shows the CDF of sustained power when Bzip2 and TPC-W are running alone. Notice from Figure 13 that TPC-W+Bzip2 consolidation has a lower sustained power than TPC-W running alone. This provides an interesting insight that by interleaving low power-consuming applications with high power consuming applications, we can reduce the probability of violating a given sustained power budget. Figure 14 and Table VIII evaluate our prediction mechanism for a server consolidating 3 applications. We are able to bound the tail of the sustained power consumption within 2% error margin (Table VIII).

B. 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

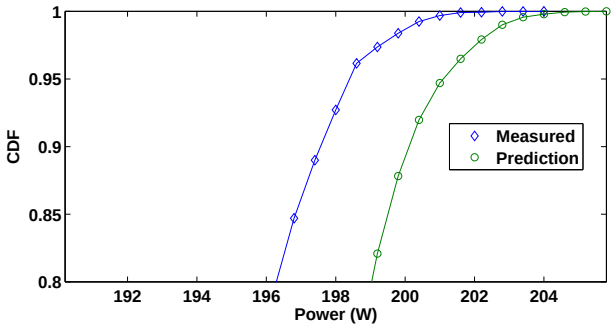


Fig. 14. Comparison of measured and predicted sustained power consumption (L=1 sec) for a server consolidating TPC-W(60), TPC-W(60) and Bzip2 with reservations 60, 20 and 20 respectively.

Probability of violation	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

TABLE VIII
EFFICACY OF SUSTAINED POWER PREDICTION ON A SERVER CONSOLIDATING TPC-W(60), TPC-W(60) AND BZIP2 WITH RESERVATIONS 60, 20 AND 20 RESPECTIVELY.

$Pr_{B_1,\cdots,B_m}(S,L)$, (that upon consolidating m servers $B_1,\cdots,B_m$ 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 II-B 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

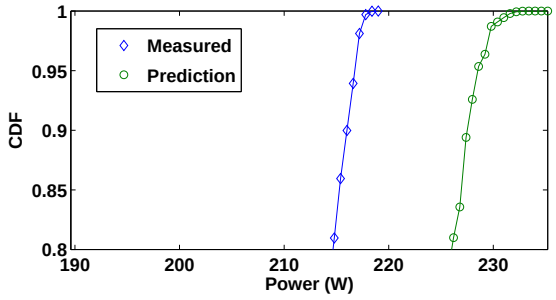


Fig. 15. Comparison of measured and sustained power consumption (L=1 sec) of a PDU connected to server1 (running TPC-W(60) and SM(100)) and server2 (running SM(100) and Bzip2). SM(x) represents Streaming Media Server streaming to x clients

Probability of violation	Measured Sustained power (W)	Predicted sustained power (W)	Error (%)
20%	214.0	226.2	5.70
10%	215.5	227.6	5.61
1%	217.5	230.4	5.93
0%	219.0	235.2	7.39

TABLE IX
EFFICACY OF SUSTAINED POWER PREDICTION ON A PDU CONSOLIDATING SERVER1 (RUNNING TPC-W(60) AND SM(100)) AND SERVER2 (RUNNING SM(100) AND BZIP2). SM(X) REPRESENTS STREAMING MEDIA SERVER STREAMING TO X CLIENTS

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 2 steps:

(Step1) Finding the distribution of average power consumption of the 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.[8]

Step1 can be easily achieved by slightly modifying the techniques developed in section V-A. Initially we estimate the CPU utilization of the consolidated applications, $Pr(U^L_{(A_1,\cdots,A_n)} = (c_1, \cdots, c_n))$ for all $(c_1, \cdots, c_n)$ and then instead of finding the minimum 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. Step2 is straightforward.

Figure 15 and Table IX evaluates our prediction mechanism for a server consisting of 2 processors each consolidating 2 applications. Since we do not have an infrastructure that could measure power consumption of a set of servers (we can measure power consumption only for a single server), we did our evaluation on a single server with 2 processors. This is analogous to an environment with two servers connected to a PDU. Even though our approach provides an upper bound for the sustained power consumption, it is not a tight upper bound. As part of our future work, we intend to develop mechanisms that provide a much tighter upper bound on the sustained power consumption.

VI. 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. As usual, we profile the power

[8]This is done using the z-transform. The z-transform of a random variable U is the polynomial $Z(U) = a_0 + za_1 + z^2a_2 + \cdots$ where the coefficient of the i^{th} term represents the probability that the random variable equals i (i.e., $U(i)$). If $U_1, U_2, ..., U_{k+1}$ 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 $U_1, U_2, \cdots, U_{k+1}$.

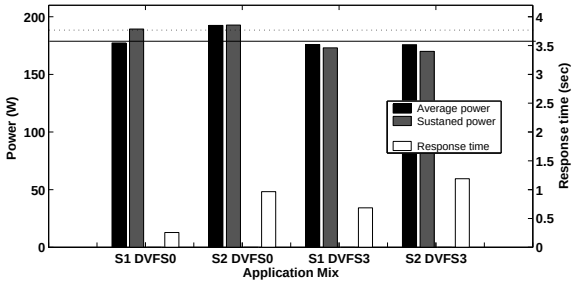


Fig. 16. Illustration of packing decisions made by our predicting technique involving 3 applications.

consumption and resource usages of these applications with the server operating at various power states (see Table X.) The salient features of the power consumption of these applications are as follows.

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: (a) an average power budget of 180W and (b) a sustained power budget of 185W 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.

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

TABLE X
PREDICTED VALUES FOR SUSTAINED AND AVERAGE POWER CONSUMPTION FOR TWO SUBSET OF APPLICATIONS AT TWO PROCESSOR POWER STATES

Our techniques predict that the only feasible configuration where all three of our applications could be co-located on the same server while meeting both the power budgets is when the CPU operates at DVFS3 (CPU operating at 2.8GHz.) Furthermore, we predict that packing any two of our applications with the CPU at the highest state DVFS0 (CPU operating at 3.4GHz) 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 X. 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. Figure 16 presents our observations for the two application sets S_1 and S_2. The figure shows that both the susbsets 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: (i) packings 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 (ii) packings for which it was indicated 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.

Finally, we claim that as a side-effect of striving to operate within specified power budgets, packing algorithms that use profiling and prediction, (in combination with application models that can translate performance goals into resource needs) are likely to find operating regimes that avoid excessive over-provisioning of power. Whereas some amount of over-provisioning is desirable to handle transient excess draw of power, lowering its degree without adversely affecting safety of operation can be very beneficial. This is similar and complementary to research that employs careful measurement and characterization of resource needs to achieve desirable trade-offs between resource utilization and likelihood of resource shortage.

VII. Related Work

While limitation on battery lifetime has been the main concern for mobile [16] and embedded systems, research on server systems [5] have 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 in 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 were proposed [6], [7], [25]. while the above research looked at reducing the number of servers, Femal *et al.* suggested that over-provisioning servers may increase the performance of throughput-oriented applications without compromising on the power budget of the infrastructure [15]. Interplay of power and performance both in the arena of uni-processors and multi-processor has been studied in great detail [1], [3], [32]. Chase *et al.* considered energy-aware resource provisioning in a data center in accordance to negotiated QoS agreements [7]. Stoess *et al.* [35] did accounting and capping of energy consumption for consolidated environments. Nathuji *et al.* looked at extending power management capabitites for the virtual machines [24]. 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 [14]. 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 [21]. Wang *et al.* extended the power capping mechanism to a cluster of servers [40]. Ranganathan *et al.* and Fan *et al.* did extensive profiling of real-world server clusters and they both observed that the *probability of synchronized peak* power consumption of all the servers happening at the same time is very low [27], [13]. Leveraging this fact, they showed that significant more applications/servers could be consolidated for the same power supply. Recent work from Ramya *et al.* looked at coordinating multiple power management activities (average and peak) happening throughout the hierarchy of a data center [26]. To the best of our knowledge, we are the first to investigate on possible prediction of simultaneous peak power consumption for a set of consolidated applications. We believe that our prediction techniques will greatly complement current research on deciding the right degree of consolidation keeping in mind the

reliability limits of the infrastructure. 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], [23], [18], [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 [20].

Modeling/Characterization of power: Modeling of power consumption has been done at various granularities from a data-center, server, individual components to an instruction [12], [17], [41] 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, SPECPower aims at characterizing the performance and power behavior of servers at different utilizations [31]. To the best of our knowledge, we are the first ones to do such an extensive characterization of power consumption in a consolidated environment.

VIII. Concluding Remarks 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.

Using a simple combination of hardware and software measurement infrastructure, we derived *power profiles*—statistical descriptions of the power consumption of applications. We used insights gained from detailed profiling of several applications—both individual and consolidated—to develop predictive models for average and sustained power consumption of our server. We implemented our techniques on a Xen-based platform and evaluated them in a wide variety of consolidation settings. Our results were promising. For a variety of consolidation scenarios, we were able to predict average power consumptions with an 5% error-margin. Our sustained power prediction techniques predict close yet safe upper bounds on the sustained power consumption of consolidated applications.

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 violated budgets. We plan to investigate how best to make our packing techniques work in tandem with such power-aware resource per-server schedulers.

References

[1] M. Annavaram, E. Grochowski, and J. Shen. Mitigating Amdahl's Law through EPI Throttling. In *Proceedings of the International Symposium on 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. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM.

[3] F. Bellosa, S. Kellner, M. Waitz, and A. Weissel. Event-driven energy accounting for dynamic thermal management. In *Proceedings of the Workshop on 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. In *Proceedings of IEEE International Conference on Autonomic Computing, Seattle (ICAC-05), WA* , June 2005.

[5] R. Bianchini and R. Rajamony. Power and Energy Management for Server Systems. *IEEE Computer*, 37(11), November 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. In *Proceedings of the IEEE Workshop on 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. In *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, 2001.

[8] J. Choi, Y. Kim, A. Sivasubramaniam, J. Srebric, Q. Wang, and J. Lee. Modeling and Managing Thermal Proles of Rack-mounted Servers with ThermoStat. In *IEEE 13th International Symposium on 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. In *Proceedings of the Sixth USENIX Symposium in Operating Systems Design and Implementation (OSDI 2004), San Francisco, CA*, December 2004.

[10] Dell Computers: PowerEdge servers SC1425 Spec Sheet . Dec 2005. www.dell.com/downloads/global/products/pedge/en/sc1425_specs.pdf.

[11] R. Doyle, J. Chase, O. Asad, W. Jin, and A. Vahdat. Model-Based Resource Provisioning in a Web Service Utility. In *Proceedings of the Fourth USITS*, Mar. 2003.

[12] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan. Full-system Power Analysis and Modeling for Server Environments. In *Proceedings of the Workshop on Modeling, Benchmarking, and Simulation (MoBS)*, June 2006.

[13] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *ISCA '07: Proceedings of the 34th annual international symposium on Computer architecture*, 2007.

[14] W. Felter, K. Rajamani, T. Keller, and C. Rusu. A Performance-Conserving Approach for Reducing Peak Power Consumption in Server Systems. In *Proceedings of the International Conference on Supercomputing (ICS)*, June 2005.

[15] M. E. Femal and V. W. Freeh. Safe overprovisioning: Using power limits to increase aggregate throughput. In *Workshop on Power-Aware Computer Systems (PACS'04)*, 2004.

[16] J. Flinn and M. Satyanarayanan. Managing battery lifetime with energy-aware adaptation. *ACM Trans. Comput. Syst.*, 22(2):137–179, 2004.

[17] 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. In *Proceedings of the International Symposium on High Performance Computer Architecture*, February 2002.

[18] T. Heath, A. P. Centeno, P. George, Y. Jaluria, and R. Bianchini. Mercury and Freon: Temperature Emulation and Management in Server Systems. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, October 2006.

[19] J. Hellerstein, F. Zhang, and P. Shahabuddin. A Statistical Approach to Predictive Detection. *Computer Networks*, Jan. 2000.

[20] IBM Active Energy Manager - Measure and Cap Energy. http://www-03.ibm.com/press/us/en/pressrelease/22551.wss.

[21] C. Lefurgy, X. Wang, and M. Ware. Server-level power control. In *ICAC '07: Proceedings of the Fourth International Conference on Autonomic Computing*, page 4, Washington, DC, USA, 2007. IEEE Computer Society.

[22] A. Merkel and F. Bellosa. Balancing Power Consumption in Multiprocessor Systems. In *Proceedings of the ACM SIGOPS EuroSys Conference*, April 2006.

[23] J. Moore, R. Sharma, R. Shih, J. Chase, C. Patel, and P. Ranganathan. Going Beyond CPUs: The Potential of Temperature-Aware Data Center Architectures. In *In the First Workshop on Temperature-Aware Computer Systems*, June 2004.

[24] R. Nathuji and K. Schwan. Virtualpower: Coordinated power management in virtualized enterprise systems. In *21st ACM Symposium on Operating Systems Principles (SOSP'07)*, 2007.

[25] E. Pinheiro, R. Bianchini, E. Carrera, and T. Heath. Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems. In *Proceedings of the Workshop on Compilers and Operating Systems for Low Power*, Sept 2001.

[26] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu. No Power Struggles: Coordinated multi-level power management for the data center. In *Thirteenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '08)*, Mar. 2008.

[27] P. Ranganathan, P. Leech, D. Irwin, and J. Chase. Ensemble-level Power Management for Dense Blade Servers. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, June 2006.

[28] Signametrics Multimeter SM2040 Series Spec Sheet. May 2000. http://www.signametrics.com/products/sm2040.pdf.

[29] SPEC CPU2000. http://www.spec.org/cpu2000/.

[30] SPECjbb. http://www.spec.org/jbb2005/.

[31] SPECpower. http://www.spec.org/specpower/.

[32] M. Steinder, I. Whalley, J. E. Hanson, and J. O. Kephart. Coordinated management of power usage and runtime performance. In *Network Operations and Management Symposium (NOMS'08)*, 2008.

[33] C. Stewart, T. Kelly, and A. Zhang. Exploiting Nonstationarity for Performance Prediction. In *Proceedings of EuroSys2007, Lisbon, Portugal*, March 2007.

[34] C. Stewart and K. Shen. Performance Modeling and System Management for Multi-component Online Services. In *Proceedings of the Second USENIX Symposium on Networked Systems Design and Implementation (NSDI'05), Boston MA*, pages 71–84, May 2005.

[35] J. Stoess, C. Klee, S. Domthera, and F. Bellosa. Transparent, power-aware migration in virtualized systems. In *Proceedings GI/ITG Fachgruppentreffen Betriebssysteme*, Oct. 2007.

[36] J. Stoess, C. Lang, and F. Bellosa. Energy management for hypervisor-based virtual machines. In *Proceedings of the 2007 USENIX Technical Conference (USENIX'07)*, June 2007.

[37] TPC-W. www.tpc.org/tpcw.

[38] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi. An Analytical Model for Multi-tier Internet Services and its Applications. In *SIGMETRICS 2005*, June 2005.

[39] B. Urgaonkar, P. Shenoy, and T. Roscoe. Resource Overbooking and Application Profiling in Shared Hosting Platforms. In *Proceedings of the Fifth USENIX Symposium on Operating Systems Design and Implementation (OSDI 2002), Boston, MA*, December 2002.

[40] X. Wang and M. Chen. Cluster-level feedback power control for performance optimization. In *Proceedings of the Fourteenth International Symposium on High-Performance Computer Architecture (HPCA'08)*, Feb. 2008.

[41] 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. Des. Autom. Electron. Syst.*, 12(3):26, 2007.