Dynamic Server Power Capping for Enabling Data Center Participation in Power Markets

Hao Chen*, Can Hankendi*, Michael C. Caramanis† and Ayse K. Coskun*
*Department of Electrical and Computer Engineering
†Division of Systems Engineering
*†Boston University, Boston, Massachusetts 02215
{*haoc, hankendi, mcaraman, acoskun*}*@bu.edu*

***Abstract*—Today's US power markets offer new opportunities for the energy consumers to reduce their energy costs by first promising an average consumption rate for the next hour and then by following a regulation signal broadcast by the independent system operators (ISOs), who need to match supply and demand in real time in presence of volatile and intermittent renewable energy generation. This paper leverages the power regulation capabilities of the servers so as to enable the data centers to participate in these emerging power markets. As the data center energy consumption continues to grow, proposed participation in the power markets has the promise to achieve significant monetary savings. The paper first solves a data center regulation service (RS) optimization problem to determine the optimal average power consumption and regulation quantity that minimize the energy cost. We then propose a dynamic server power capping technique to modulate the real-time power consumption in response to ISO requests while maintaining the desired quality-of-service (QoS). Experiments on a real-life server demonstrate that our technique can reduce the energy cost by 29% on average compared to using a fixed power cap.**

I. INTRODUCTION

Today, the needs to incorporate renewable energy sources (e.g., hydropower, wind power and solar energy) in power generation are growing rapidly. However, because of the lack of reliable and economical large-scale energy storage solutions, the adoption of these intermittent renewable energy sources creates challenges for ISOs who need to match supply and demand in electricity markets by securing some flexible regulation service (RS) reserves in forward markets and dispatching them in real time [17]. Failing to balance the supply and demand can lead to catastrophic events such as blackouts. Hence, power RS reserves have been playing an increasingly important role in modern power markets.

In tandem with the developments in the power markets, electricity used by the data centers has grown to account for 3% of the overall consumption in the US today [8]. Recent advancements in power capping and power management techniques for the servers in the data centers (e.g., [3], [10], [23]) have enabled the data centers to provide some flexibility in their energy consumption. Therefore, data centers offer a unique opportunity for providing power RS reserves. Exploiting this flexibility can help satisfy most of the growth in data center energy consumption from the renewable energy, and also provide additional reserves to other less flexible uses of electricity in our society.

The major contributions of our work are (1) we demonstrate the proof-of-concept that data centers offer the necessary capabilities to participate in the emerging power markets, and (2) we show that the energy costs can be dramatically reduced when data centers provide RS reserves. We first formulate and solve the optimization problem to minimize the energy costs while satisfying both user QoS constraints and a given ISO signal tracking error constraint. We then propose a practical technique to modulate the server power based on the ISO signal. We evaluate our optimization framework and the runtime implementation policy on a real-life multi-core server. Power capping, where compute nodes operate below a given peak power value, is commonly used today for meeting peak power constraints (e.g., [3], [21]). However, to the best of our knowledge, ours is the first work to leverage server power capping for enabling the data center to participate in the advanced power markets. Our experimental results show that we can achieve up to 29% monetary savings while satisfying the performance constraints.

The rest of the paper starts with a discussion of the background in power markets and data center power capping/management techniques. Section III outlines the general operational framework of a data center that participates in the power market. Section IV introduces our power and performance modulation methodology. Section V first solves the RS optimal bidding problem on a single server, and then provides our runtime power capping policy. Section VI presents the experimental results and Section VII concludes the paper.

II. BACKGROUND AND RELATED WORK

In this section, we first introduce the background on the power market and the RS provision problem, then we outline the related work on power capping and management.

A. Power Markets and Regulation Service (RS) Provision

Power market RS provisions have been widely studied in recent years. There are several power markets with different time-scales, where energy and reserve transactions are cleared simultaneously. These power markets are adopted to match electricity supply with demand in real time. Among them, short-term markets are the most relevant to our work. Short-term markets contain the day-ahead market [13], the hour-ahead adjustment market and the 5-minute economic dispatch market ([9], [14]). Some management mechanisms exist in the power markets for pricing, allocating energy, and securing the reserves needed for uncertainty contingency planning. Reserves secured in the forward markets include primary (also

978-1-4799-1071-7/13/$31.00 ©2013 IEEE

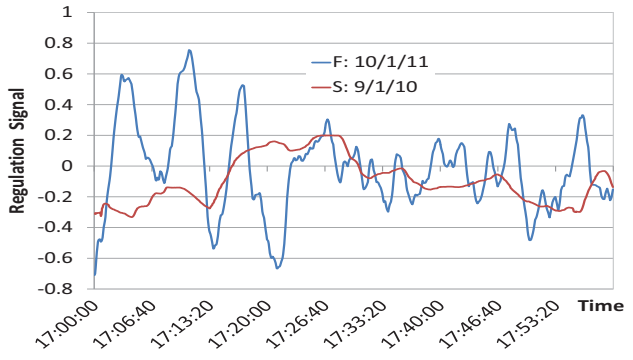

Fig. 1. Typical PJM 150sec ramp rate (F) and 300sec ramp rate (S) regulation signal trajectories.

known as *frequency control*), secondary (known as *regulation service*, RS), and tertiary reserves which are deployed by commands issued respectively in millisecond, 5 second and 5 minute intervals ([18], [19]). In our work we focus on the RS reserves in the hour-ahead power market as the price of reserves is high and data center can modulate their power at this timescale. Currently, RS reserves are mainly offered by centralized generators; however, market rules are changing to allow the demand side to provide reserves as well. For example, PJM, one of the largest US ISOs, has allowed electricity loads to participate in reserve transactions since 2006 [17]. Other ISOs are contemplating to follow this trend.

For each participant in the RS reserves power market, an average power consumption $\bar{P}$ and an RS reserve provision R should be declared to the ISO an hour in advance of each hour. With market clearing prices for energy consumed and RS reserves, Π^E and Π^R, the participant is charged for its average energy consumption and credited for the RS reserves such that the participant pays a net amount of $\Pi^E\bar{P} - \Pi^R R$. However, the credit for the RS reserves does not come for free. As the hour unfolds, the participant is asked to modulate its power consumption $P(t)$ dynamically so as to track the RS signal $z(t)$ broadcast by the ISO by ensuring that $P(t) \approx \bar{P} + z(t)R$. Part of the RS income $\Pi^R R$ is reduced based on the amount of the *tracking error*. Moreover, if the tracking error exceeds a statistical tolerance constraint, the participant may lose its contract for RS reserves provision [19].

The RS signal $z(t)$ is the main tool used by ISO to balance the supply and demand in the power market. This signal is generated based on the real-time power market situation. The specific signal dynamics are determined through an integral proportional filter of the Area Control Error, which measures the difference between the actual and scheduled net imports from adjacent balancing areas and frequency excursions outside of the tolerance interval $[59.980, 60.020Hz]$ [22]. $z(t)$ is a scalar in the interval $[-1, 1]$ with an average of zero over longer time intervals. $z(t)$ is updated every 4 seconds in increments that do not exceed $\pm R/(\tau/4)$, where τ is 150 seconds for the fast (F) RS and 300 seconds for the slower (S) RS. Typical hour-long trajectories of $z^F(t)$ and $z^S(t)$ from PJM historical data [22], which in fact conform to $|z^F(t+4) - z^F(t)| < 4/150$ and $|z^S(t+4) - z^S(t)| < 4/300$, are shown in Figure 1.

For intelligent RS provision in the hour-ahead power market, Caramanis *et al.* investigate optimal dynamic pricing policies for RS bidding [2]. Paschalidis *et al.* propose a market-based mechanism, which uses the dynamic pricing policy to enable a smart grid building operator to offer RS reserves and meet the ISO requirements [15]. Prior work on optimizing demand-side RS reserves, however, does not consider data centers as a potential participant.

B. Power Management and Capping

Server power management has advanced significantly in the recent years. Most of the modern processors support dynamic voltage-frequency scaling (DVFS) and have power gating capabilities to tune off idle units (e.g., [10], [23]). Multi-core processors have additional degrees of freedom in workload scheduling and allocation; thus, they introduce new opportunities for power management [24]. Power capping is another important technique that is widely used for meeting the peak or average power constraints today. Some modern processors provide power capping capabilities (e.g., [1], [4]). Gandhi *et al.* propose a power capping strategy that inserts idle cycles during execution to meet a given power budget [6]. For multi-threaded applications, policies that combine DVFS with thread allocation/migration improve the granularity and accuracy of capping (e.g., [3], [20]). For a mixture of single and multi-threaded applications, Ma *et al.* propose a power capping technique that power-gates the cores and applies per-core DVFS [11]. Reda *et al.* propose to improve performance under dynamically changing power caps by training power-performance models for a target server by using thread allocation with DVFS [21].

In virtualized servers, Nathuji *et al.* propose a power allocation technique that uses CPU utilization data for each virtual machine (VM) to allocate a given power budget across the VMs according to the service level agreement (SLA) requirements [12]. Resource management policies in virtualized environments affect the performances-power profiles of the servers. Hwang *et al.* calculate the optimum number of virtual cores (vCPUs) for single-threaded applications that have distinct characteristics (i.e., memory/CPU-bounded) and they propose a consolidation policy that uses DVFS and core power gating [7]. Vasic *et al.* propose to reduce the resource management overhead by making resource allocation decisions based on the history of the VMs [25].

Power management of servers is being addressed at many levels today; however, none of the existing techniques use the data center as a grid load stabilizer or enable participation in the power markets. Our work is the first to integrate RS provision and data center power management techniques together. This paper solves the data center RS optimization problem to determine the optimal average power consumption and the RS quantity to minimize the energy cost. We also implement a power capping technique on a real-life system using existing control knobs in virtualized servers to enable RS signal tracking.

III. GENERAL FRAMEWORK FOR DATA CENTER REGULATION SERVICE PROVISION

This section gives an overview of the data center power regulation framework. Figure 2 shows how the different sub-

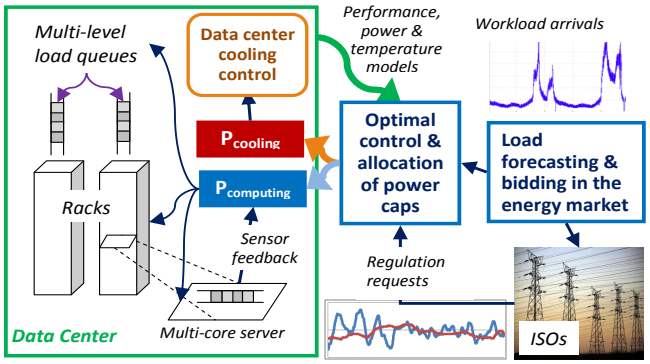

Fig. 2. Data center power RS provision framework.

components of the RS provision problem come together. The whole RS provision process includes the following steps:

(1) The data center first acquires the information of workload arrivals for the next hour. If such information is not available, the data center forecasts the workload based on the historical workload patterns.

(2) Using an estimation of future workload arrivals and the ISO requirements (e.g., tracking error), the data center computes the optimal average power consumption $\bar{P}$ and the RS reserves R, and bids in the power market for ($\bar{P}$, R).

(3) Once the ISO approves the bid, an RS signal $z(t)$ is dynamically broadcast to the data center from the ISO. Then the data center optimally distributes the total power cap $\bar{P}+z(t)R$ to the cooling units and to each server. The data center also performs workload allocation to servers.

(4) The cooling system maintains the thermal constraints and gives temperature feedback to the data center control unit. Because of the larger time constants involved in cooling temperature dynamics, ideally cooling power adjustment is performed less frequently compared to server power regulation.

(5) Each server has multiple cores and different levels of load queues. Workloads run on the servers and dynamic power capping is applied to track a given server-level cap. Performance and QoS feedback from each server is sent back to the data center control unit.

(6) Based on the feedback from the cooling units and servers, the data center re-allocates power caps and workloads so as to track the RS signal dynamically and to improve performance.

(7) The data center repeats the steps above for each hour.

In this paper, we focus on the power regulation of the computational units (i.e., servers) for fast regulation, as cooling power can be regulated as part of slower frequency markets due to the thermal time constants. Server-level power control is the main building block and a pre-requisite to achieve fine-grained power control at a few second intervals. Hence, our work solves the RS signal tracking problem on a single multi-core server, providing the proof-of-concept for RS signal tracking capabilities. Our technique can be integrated with data center level power budgeting techniques and cooling management techniques. In the rest of the paper, we focus on the single server sub-problem.

IV. METHODOLOGY FOR POWER-PERFORMANCE MODULATION

In this section, we describe our methodology for dynamically modulating the server power consumption, $P(t)$, to meet the ISO power signal, $\bar{P} + z(t)R$, for a given ($\bar{P}$, R). We explain how to derive ($\bar{P}$, R) in Section V.

We conduct our experiments on an AMD Magny Cours (Opteron 6172) based server. Magny Cours has 12 processing cores on a single chip. We virtualize the server using the VMware vSphere 5.1 ESXi hypervisor. We run selected applications from the PARSEC 2.1 benchmark suite [16], where each application runs on a VM in isolation (without any consolidation). We select the PARSEC suite as multi-threaded workloads are becoming increasingly common in the data centers as well as in supercomputing clusters. The server is connected to a Wattsup power meter that measures power consumption at every second.

In this paper, we use the *CPU resource limits* control knob in the hypervisor to tune the power for tracking the RS signal and control the power-performance settings at runtime. Using resource limits, we are able to quickly and dynamically change the server resources allocated to a virtual machine (VM) at a fine granularity, resulting in finer-grained power regulation compared to DVFS or power gating. Using models that relate the amount of resource limits, the power consumption and the performance of the server, we can set both the power and the performance at any needed level. Today over 50% of the data center servers are virtualized; therefore, changing resource limits is a practical and efficient way to control power and performance.

Next, we test the capacity of the server for participating in power markets by using the CPU resource limits control knobs. We also derive the power-performance model, which is used in solving the optimal bidding problem (Section V.A), and the power - resource limits model which is used in the runtime policy for tracking the ISO signal (Section V.B).

A. Regulation Service Provision Capacity Test

Before issuing a contract to an RS reserve provision candidate, ISO first gives the candidate a test signal to track and examines its performance mainly on three aspects: (1) the capability of consuming a stable power for a period of time (i.e., 5 minutes); (2) the time required for power consumption to ramp up to $\bar{P}+R$ and down to $\bar{P}-R$; (3) the capability of making dynamic power changes at a sufficiently fine granularity [19]. Our experimental results on the AMD server show the following:

- *Power Stability*: We keep the resource limit at a fixed setting for 10 minutes and observe the fluctuation of the power consumption. The standard deviation of the power consumption when a given PARSEC application is in its parallel phase is 1-3W, which is only 1-2% of $\bar{P}$. Note that if the power dynamics of the application running on the server changes, resource limits can be adjusted to keep $P(t)$ stable.
- *Ramp-up Capability*: Our server shows the ability to ramp up to 153W and down to 66W (66W is the server idle power, 153W is the maximum power consumption of blackscholes) at 1s intervals.

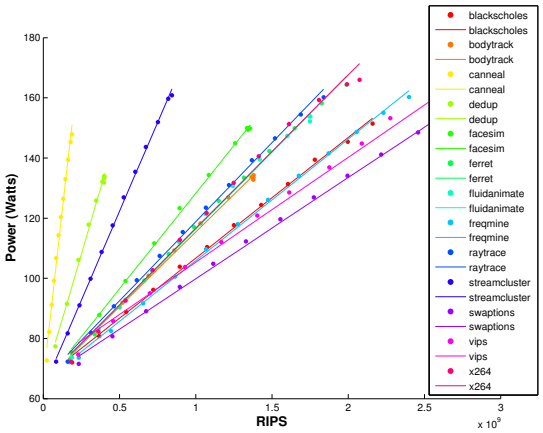

Fig. 3. Power-RIPS models for PARSEC workloads using linear fit. The dots show the measurements and the lines show the models.

- *Granularity of Modulation*: The resource limit control knob is able to modulate the power consumption in a granularity of a few milliwatts.

These results show that our server, a typical virtualized server in data centers, can meet all the ISO test requirements using the CPU resource limit control knob and it has sufficient capability for providing RS reserves.

B. Power - Performance Model

Retired Instructions Per Second (RIPS) is a commonly used metric for evaluating the performance of an application. A higher RIPS represents a faster processor service rate. We use RIPS as the performance metric in our power-performance model. Our model is derived by using regression methods on the power-performance data we measured on our server for each PARSEC benchmark. The results show that a linear regression $P_j = C_{1,j} * RIPS_j + C_{2,j}$ constructs the model with a least square error of less than 5%, where j represents the type of the workload (i.e., the specific application). We observe that $C_{1,j}$ differs depending on the application, while $C_{2,j}$ values are similar among the workloads. This is expected as in this linear model, $C_{2,j}$ is the power consumption when $RIPS_j = 0$, which means $C_{2,j}$ is actually the server idle power, and therefore is not influenced by the type of the workload. The data and model fits for each PARSEC benchmark are shown in Figure 3. We use this linear power-performance model while simulating the system for computing the optimal $(\bar{P}, R)$.

C. Power - Resource Limits Model

Figure 4 shows the power consumption of four PARSEC benchmarks as a function of the CPU resource limits. CPU resource limits restrict the maximum amount of time-share of the virtual CPUs (vCPUs) that are scheduled on the physical cores (pCPUs) and these limits are represented in units of MHz. The maximum amount of CPU resources that can be allocated to a VM is equal to the number of vCPUs multiplied by the frequency of the cores (f_{core}). For our system, the maximum amount that can be allocated to the VM is equal to $(12 \cdot f_{core})$ MHz. For most of the benchmarks, power and CPU resource limits are linearly correlated; therefore, the peak power consumption is observed at the highest CPU resource

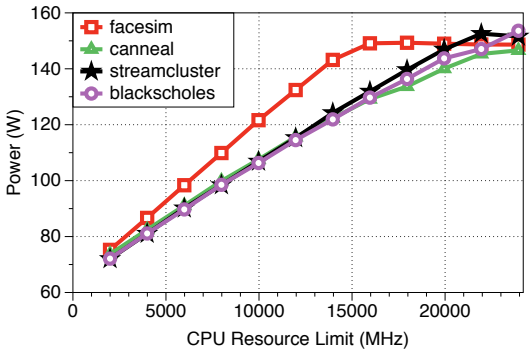

Fig. 4. Power-resource limit models for the PARSEC applications.

limit. However, for benchmarks such as `facesim`, power consumption is constant after reaching a certain amount of CPU resources, as the benchmark cannot continue to utilize the resources efficiently.

To capture the various relationships between power and CPU resource limits, we monitor the overall CPU usage (in MHz) (i.e., CPU_{used}) and the amount of CPU resources that are not utilized because of the existing CPU resource limits on the system (i.e., CPU_{ready}). VM statistics (i.e., CPU_{used} and CPU_{ready}) are polled every 2 seconds using the vSphere SDK for GuestOS library. CPU_{ready} metric allows us to capture the saturating performance effects (e.g., `facesim`), as it reflects the amount of CPU resources needed to reach the maximum performance. The CPU_{used} value captures the utilization levels of the pCPUs caused by the VM, excluding the system activity.

We also use feedback from the power meter to update the power/CPU resource limits models dynamically at runtime. In other words, power measurements (i.e., P_{tot}) are fed into the power-CPU resource limit model to estimate the CPU resource limit that meets the power constraints. We first remove the idle power consumption from the measured total power to compute the dynamic power, $P_{dyn} = P_{tot} - P_{idle}$. In our runtime policy, we estimate the CPU resource limit value, (R_{cap}), that meets the given power cap by using the Eq. 1, where P_{cap} is the target power constraint. From our experiments, the estimation error of this model is less than 5%.

$$R_{cap}(MHz) = (CPU_{used} + CPU_{ready}) * P_{cap}/P_{dyn} \quad (1)$$

All applications are run with 12 threads in our experiments. At lower power caps, running the applications with a high number of threads might introduce overhead due to potential resource contention. To reduce the overhead caused by running a high number of threads at lower power caps, we first derive the minimum number of active vCPUs (n_{vCPU}) such that $n_{vCPU} \cdot f_{core} > R_{cap}$. We then *pack* the active threads onto n number of vCPUs by setting thread affinities in the OS, as in [3]. Limiting the active number of vCPUs reduces the resource contention and achieves better performance. Note that thread packing alone often does not provide the sufficient granularity to match the given power caps; hence, we allocate R_{cap} amount of CPU resources for the VM.

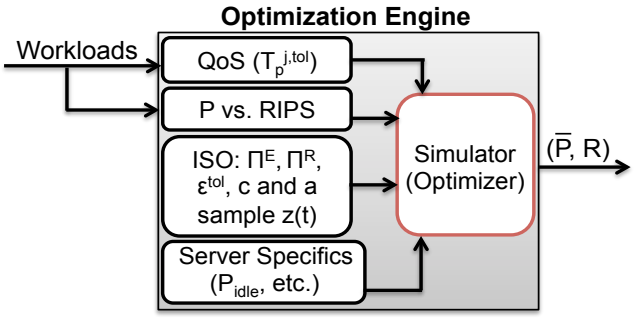

Fig. 5. The overview of the optimal bidding engine.

V. SERVER LEVEL POWER REGULATION SERVICE PROVISION

In this section, we introduce our technique for server-level RS provision. Given server and workload properties, we first describe how to find the optimal ($\bar{P}$, R). We then describe the runtime policy for tracking the power cap imposed by the ISO, $\bar{P} + z(t)R$, with a small tracking error.

In our work, we assume that a server runs one application at a time. Incoming workloads are maintained in a first-in-first-out (FIFO) queue. If the server is idle and the queue is not empty, the server starts running the next workload in the queue. This servicing policy is straightforward and widely used in today's data centers. Our approach is generalizable to servers with multiple priority queues, where jobs with higher priorities are serviced first.

A. Optimal Bidding Engine

The very first step of providing RS at the hour-ahead power market is to compute the average power and RS amount, ($\bar{P}$, R), for bidding in the power market every hour. The optimal bid should minimize the monetary costs while meeting the ISO requirements and user QoS constraints. We design a bidding engine to calculate the optimal bid.

We first formulate the optimization problem. We use a metric called *tracking error* to evaluate the tracking performance, which is defined as $\epsilon(t) = |P(t) - (\bar{P} + z(t)R)|/R$. For a data center, the total *system time* T for an application (i.e., waiting time plus processing time) is one of the most significant QoS indexes for the customers. However, as a data center runs thousands of applications per hour in general, it is highly inefficient or even impossible to look at the QoS constraint for each individual workload in the hour-ahead bidding optimization process. In fact, in a real-life data center, a large number of customer workloads are the same type, with the same priority and QoS requirements. Thus, we categorize workloads by type j (i.e., blackscholes, streamcluster, etc.) and priority level p (high, medium, low, etc.), and define their system time as T_p^j. Then we set statistical constraints on both $\epsilon(t)$ and T_p^j, $\forall p, j$. We denote the mean and standard deviation of $\epsilon(t)$ and T_p^j by $\bar{\epsilon}$, σ_ϵ and $\bar{T}_p^j$, $\sigma_{T_p^j}$. $\bar{\epsilon}$ and σ_ϵ define the tracking performance, which will affect RS revenues. Moreover, the data center will be disqualified for providing RS in the power market if a probabilistic tolerance of tracking error is exceeded. $\bar{T}_p^j$ and $\sigma_{T_p^j}$ define the user QoS contract guarantees, i.e., workloads from customers are guaranteed to perform within a certain QoS tolerance at a given confidence level. In this work, we focus more on computing workloads rather than real-time services, hence we set QoS tolerance as constraints instead of using real-time SLA delay penalties while formulating the problem.

We study the sample frequency distributions of the $\epsilon(t)$ and T_p^j trajectories for a sufficiently large number of simulations and notice that they mostly fit the Gamma distribution [5], Γ, with parameter shape k: $k_\epsilon = \bar{\epsilon}^2/\sigma_\epsilon^2$, $k_{T_p^j} = \bar{T_p^j}^2/\sigma_{T_p^j}^2$ and scale θ: $\theta_\epsilon = \sigma_\epsilon^2/\bar{\epsilon}$, $\theta_{T_p^j} = \sigma_{T_p^j}^2/\bar{T_p^j}$, for $\epsilon(t)$, an T_p^j. Hence when we solve the optimal bidding problem, we use Gamma distribution to construct the probabilistic constraints. We also assume the probabilistic tolerance of the tracking error given by the ISO is ϵ^{tol}, the QoS tolerance on each workload group (p, j) from data center users is $T_p^{j,tol}$, and the statistical confidence level required on these tolerances is P_{conf}.

Finally, based on the given workload and server information, we apply limits on $\bar{P}(t)$ based on the maximum achievable power value P_{max} and the server idle power P_{idle} which is basically the minimum achievable power value, assuming that the server is never shut down. The optimization problem is formulated as follows:

$$\begin{aligned} \underset{\bar{P},R}{\text{minimize}} \quad & \Pi^E \bar{P} - (\Pi^R R - \Pi^R c[\sigma_\epsilon^2 + \bar{\epsilon}^2]) \\ \text{subject to} \quad & \Gamma^{-1}(k_\epsilon, \theta_\epsilon, P_{conf}) > \epsilon^{tol}, \\ & \Gamma^{-1}(k_{T_p^j}, \theta_{T_p^j}, P_{conf}) > T_p^{j,tol}, \\ & \bar{P} + R < P_{max}, \\ & \bar{P} - R > P_{idle}, \\ & \bar{P} \geq 0, R \geq 0. \end{aligned} \tag{2}$$

where Π^E is the hour ahead clearing price of power and Π^R is the hour ahead clearing price of RS reserves, both in \$/kWh. In general $\Pi^R \approx \Pi^E$. c is the penalty coefficient on the second moment of the tracking error.

The optimal bidding engine is shown in Figure 5. The electricity price information (Π^E and Π^R), a sample RS signal $z(t)$, the tracking error tolerance given by the ISO (ϵ^{tol}) and the server specific information, e.g., P_{idle} are saved in the engine. The inputs of the engine are the information on the workloads and customer QoS contracts for the next hour. Using the workload information, a power-RIPS model (see Section IV) is derived first. Then along with the Power-RIPS model, all these inputs are sent to a simulator which simulates the whole RS provision process. The simulator uses exhaustive search to find the optimal ($\bar{P}$, R) values that satisfy the constraints. It is possible to first conduct a sensitivity analysis on $\bar{P}$ and R and use the results to construct a more structure search. Using exhaustive search, simulation takes only a few seconds; so it is not necessary to optimize the search for the problem size we focus on in this work.

In this paper, we assume that workload information (i.e., workload types and arrival rates) for the following hour is provided in advance. This is reasonable as for many real-life cases in the data centers, workload information is provided by customers to the data center some time before applications start executing. (e.g., in the case of batch job submissions in high-performance computing clusters). It is common to allocate a set

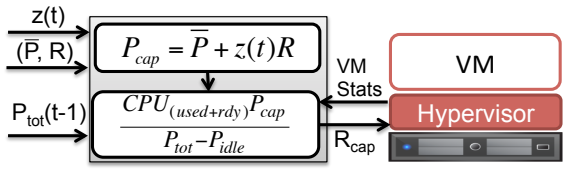

Fig. 6. The overview of the runtime power capping technique. Our technique receives input from the ISO and the VM (e.g., CPU_{ready}, etc.) to make CPU resource limit adjustments so as to keep the power consumption close to the current power cap.

of servers for specific types of applications (i.e., a blackscholes server, etc.). Mechanisms for workload forecasting can be designed and used in conduction to our optimization technique.

We solve the problem for a system with a single queue in this work. However, as shown in the equations above, it is straightforward to use the same formulation for optimizing the power regulation for a system with multiple priority queues.

B. Runtime Policy

The goal of the runtime policy is to track the RS signal on a virtualized server environment. Our power capping technique is implemented on a manager node (i.e., vCenter, which is part of the VMware virtualization framework), which have permissions to perform administrative tasks such as changing the resource limits on the ESXi hosts. Figure 6 shows the overview of our implementation. Our power capping technique receives three inputs: (1) $(\bar{P}, R)$ signals from the optimization engine, (2) the real-time power measurements (i.e., $P(t-1)$) from the power meter and (3) CPU resource usage statistics from the VM. The output of the power capping module is the CPU resource limit that is expected to keep the power consumption of the server close to the current power constraint.

After receiving the RS signal $z(t)$, we derive the corresponding power cap value, $P_{cap} = \bar{P} + z(t)R$. We then use Eq. (1) to calculate the CPU resource limit that matches the P_{cap}. After calculating the CPU resource limit, our policy communicates with the ESXi host to reconfigure the VM CPU resource limits by using the vSphere SDK library. We monitor the power consumption and the power cap value every second, and adjust the CPU resource limits if the average absolute tracking error over the last two 2 seconds is larger than 2W.

In our implementation, workload arrivals are managed through a queue manager module, `qMan`. `qMan` monitors the workload queue and executes applications based on the arrival time. Multi-threaded workloads consist of serial I/O phases and a parallel phase (i.e., region-of-interest (ROI)). In real-life applications, the parallel phases of the multi-threaded workloads dominate the compute cycles of the data centers; thus it is important to consider the ROI of the multi-threaded workloads to accurately evaluate the benefits of our technique. To evaluate only the ROIs of the PARSEC workloads, we synchronize the RS signal arrivals with the start and the end points of the ROIs. In default PARSEC package, ROIs of the benchmarks are marked with `ROI_start()` and `ROI_end()` functions of the `HOOKS` library that is included in the PARSEC package. We utilize the `HOOKS` library to detect the entrance and the exit points of the ROI. When a PARSEC benchmark is in its I/O stage, we pause the RS signal and wait for the benchmark to reach its ROI. We continue to receive the RS signal, when a benchmark reaches the ROI. Therefore, we evaluate our technique only for the ROI portions of the applications.

VI. EXPERIMENTAL RESULTS

In this section, we describe our experimental framework and quantify the benefits of our technique.

A. Workload Generation

As we focus on the *hourly* RS provision, we need to generate workload over one hour for the server to process. We generate the workload using Monte Carlo simulation and based on a general queuing model, where applications arrive at each queue in the system following an exponential distribution. Such models are commonly used to mimic the workload arrival behavior in real data centers. Queues with different priorities p can have different workload arrival rates, λ_p. We use a single queue in our experiments.

For each workload i, a random number $r_1^i \in [0,1]$ is generated and this number is used to determine the workload inter-arrival time, i.e, $\tau^i = -ln(1-r_1^i)/\lambda$ (can be generalized to multiple priority queues using τ_p^i and λ_p). In addition, we use another random number, r_2^i, to determine the type of the workload i. For example, in our experiments four benchmarks are selected from the PARSEC-2.1 suite. We number the workloads and use $r_2^i \in [1,4]$ to determine the type of the workload i. E.g., $r_2^i = 1$ means the application is blackscholes, etc. We also run experiments for homogeneous workload arrival cases (i.e., the system runs workloads consisting of only blackscholes or only streamcluster). Such homogeneous execution scenarios can be achieved by configuring the data center job allocation policy accordingly.

We set the workload arrival rates to achieve approximately 50% system utilization, which is a typical scenario in today's data centers. That is, the server is busy in processing workloads for 50% of time when no capping is applied.

B. Optimal Bidding Solution

We solve the optimization problem provided in Eq. 2 using the following values: $\Pi^R = \Pi^E = 0.1\$/kWh, c = 1, P_{conf} = 0.85, \epsilon^{tol} = 0.2$ [19], $P_{idle} = 66W$. P_{max} changes depending on the workload type. For PARSEC workloads running on our AMD server, P_{max} is in the range of 130W to 170W. As we have only one priority queue, the system time tolerance for the workload type j is $T^{j,tol}$. This tolerance depends on workload type, as even without any power capping, the runtimes are different among the workloads. In order to have a unified evaluation criteria among different types of workloads, we calculate the the system time of each application as a multiple of its shortest finishing time, $T^{j,min}$, which is the time the application spends on the system without any power capping or resource restrictions. We use T^{tol} to represent unified system time tolerance, which is $T^{tol} = T^{j,tol}/T^{j,min}$, for all j. In our experiments we set $T^{tol} = 8$, which means we allow the system time of a workload to be eight times of its shortest finishing time. For the case of multiple priorities, we can use a smaller T^{tol} for the higher priority workloads and a larger T^{tol} for the low priority workloads.

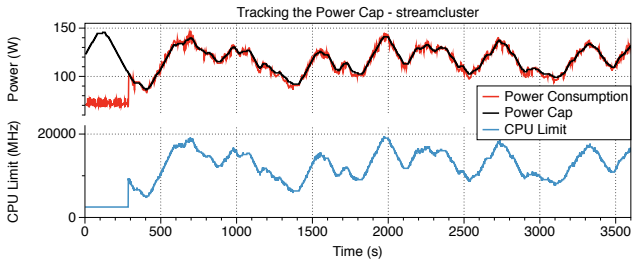

Fig. 7. 1-hour ISO signal power capping of streamcluster workload by adjusting the CPU resource limits. We both show the actual power consumption and the power cap values (top figure) and the dynamic adjustment of CPU resources (bottom figure)

The optimal $(\bar{P}, R)$ values for different workloads are shown in Table I. The first two rows are for the homogeneous workload cases, in which all workloads arriving are of the same type, i.e., streamcluster and blackscholes. The last three lines are the results of the heterogeneous workloads cases, in which we have four different applications. Without loss of generality, we create the heterogenous workloads queues using blackscholes, canneal, streamcluster and facesim. We run the experiment for the heterogenous case three times, where each instance has the same RS signal but a different workload queue, as the workloads are generated probabilistically using exponential distribution.

We observe that the optimal RS reserve R is always 30% of its corresponding average power consumption $\bar{P}$ and around 23% of the maximum power consumption P_{max} for each case. This result implies that the the optimal solution does not change significantly among different workloads. Based on these results, it is possible to design a faster optimal bidding method where the optimal solution search is centered around $R = 0.23 * P_{max}$. Such heuristics would be useful when expanding the optimization to large problem sizes.

After we compute the optimal bids for the average power consumption and the RS reserves ($\bar{P}$, R), we implement our policy to track the ISO power signal $\bar{P} + z(t)R$, on our real-life server. We evaluate the tracking accuracy, the user QoS, and the monetary savings.

C. Tracking Performance

We first investigate the homogeneous workload case. Figure 7 shows 1-hour long power profile for streamcluster workload. The result shows that except for the idle period at the very beginning, during which time we are unable to tune the power by resource limits control knobs, our runtime policy enables our server to dynamically track the ISO power cap very accurately. The average tracking error is 18% of R including idle period and 6% of R without considering the idle period. For the homogeneous blackscholes workload, which is a CPU intensive workload while streamcluster is a memory-intensive workload, the average tracking error is 15% of R including the idle period and 5% without considering the idle period.

We then investigate the heterogeneous workload case, where the workload consists of different applications. The results of the three experiments are shown in Figure 8. We achieve high tracking performance for all three experiments, with the average tracking errors as 15%, 10% and 16% of R including the idle period and 7%, 8% and 7% without considering the idle periods, respectively. Note that the constraint for the tracking error was 20% in the problem formulation.

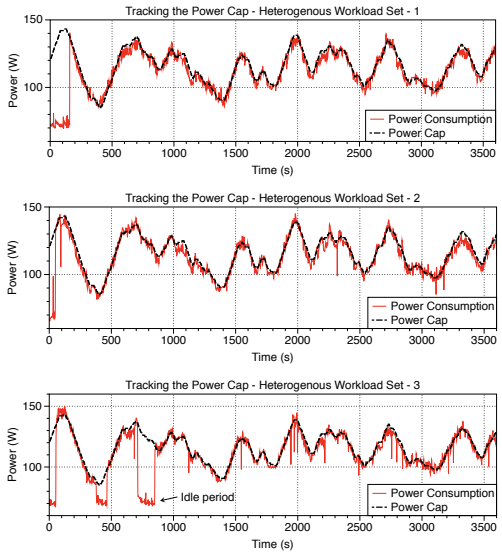

Fig. 8. 1-hour power profile of the server running the heterogeneous workloads when we apply our proposed power capping technique.

All the results confirm that our runtime policy is able to track the ISO power cap very accurately and satisfy the tracking error tolerance in both homogeneous and heterogeneous workload cases. This result is very promising as it shows that the success of the optimization is not constrained by the workload type.

D. Workload QoS Evaluation

In our work, the QoS values of the workloads are evaluated by the workload system time, which is represented in relation to the workload shortest finishing time, so that all workloads have a unified QoS evaluation scale. Figure 9 shows the average system time over each of the 1-hour experiments. We compare the optimal RS provision case (i.e., $(\bar{P}, R)$) against the case without providing any RS (i.e., $(\bar{P}, 0)$) for two homogeneous workloads and three heterogenous workload sets (i.e., Mix1, Mix2 and Mix3).

As Figure 9 shows, the optimal RS case provides similar performance when compared to the case without RS reserves. Providing RS even improves the performance for some cases (i.e., Mix2). Overall all the QoS constraints are satisfied within the tolerance T^{tol}. Thus, providing RS do not cause significant performance degradation and provides energy cost savings.

E. Monetary Savings

The monetary costs of the energy in the RS provision case is calculated by the objective function in Eq. 2. If no RS provision is claimed, the monetary costs are calculated based on the energy consumed, in our case which is $\Pi_E * E$, E is the energy consumed of the server during the hour.

In our experiments, we evaluate a single multi-core server and run experiments for an hour. However, data centers generally contain thousands of servers. We assume a data center,

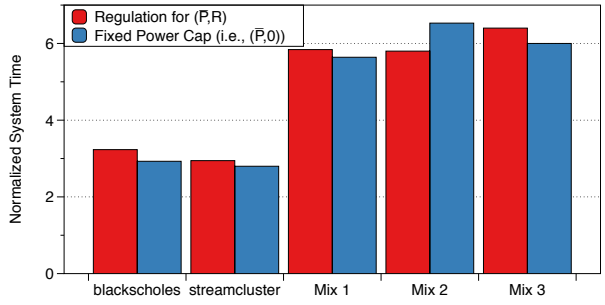

Fig. 9. The QoS (the system time) comparison between the optimal RS provision ($\bar{P}$, R) case and the fixed power cap case (i.e. $\bar{P}$, 0) for various homogeneous and heterogenous workloads. The system time shown here is normalized by the shortest system time without any power capping.

TABLE I. THE OPTIMAL BIDS AND THE MONETARY COSTS COMPARISON

	$\bar{P}$	R	\$ w/o Cap *	\$ fixed $\bar{P}$ **	\$ ($\bar{P}$, R) *
Streamcluster	116.7	35	119.2	113.8	84.0
Blackscholes	117.6	35	115.42	115.6	82.8
Mix 1	116.5	34	127.2	116.2	82.7
Mix 2	115.9	34	128.7	117.3	82.0
Mix 3	115.9	34	126.3	115.4	82.1

* w/o Cap stands for the case without any power cap.
** fixed $\bar{P}$ stands for the case with a fixed power cap $\bar{P}$ (i.e., $R = 0$).
* ($\bar{P}$, R) stands for the case of the optimal RS provision.

which consists of 10,000 identical servers, and calculate the monetary costs of it in various power capping scenarios. Table I shows the monetary costs (\$) per hour for the data center for three cases: (1) the case without any power caps, (2) the provision without regulation (i.e., with a fixed power cap $\bar{P}$), and (3) the case of proposed optimal RS provision (i.e., with $\bar{P}$, R). We evaluate all three cases for 2 homogeneous and 3 heterogenous workload sets. The energy consumption for non-RS provision cases (i.e., case (1) and (2)) is calculated based on the power measurements from the power meter.

As the results shown in Table I, the proposed optimal RS provision case reduces the energy costs by around 29% by providing RS for all type of workloads. In comparison to the case without any power cap, the proposed technique provides up to 36% monetary savings.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we have studied the capability and the benefits of data centers to provide power RS. We have constructed a server-level RS provision framework and solved the optimization problem for computing the RS reserves that maximize the energy cost savings subject to tracking error and customer QoS constraints. We have also implemented a runtime policy on a real-life server for RS provision by using the CPU resource limit control knob. Our experimental results show that our technique is capable of providing RS and meeting the QoS guarantees, while also reducing the energy costs by 29% on average. In our future work, we plan to investigate a wider range of application domains, such as real-time enterprise and transactional workloads. We are also going to investigate varying timescale markets for other power sources (cooling storage, network, etc.) in data centers to take part in. Other directions include integrating our technique with workload forecasting and data center power budgeting techniques.

REFERENCES

[1] T. Samson. AMD Brings Power Capping to New 45nm Opteron Line. *http://www.infoworld.com/d/green-it/amd-brings-power-capping-new-45nm-opteron-line-906*, 2009.

[2] M. C. Caramanis, I. C. Paschalidis, C. G. Casandras, E. Bilgin, and E. Ntakou. Provision of Regulation Service Reserves by Flexible Distributed Loads. In *Proceedings of IEEE Conference on Decision and Control*, pages 3694-3700, 2012.

[3] R. Cochran, C. Hankendi, A. Coskun, and S. Reda. Pack & Cap: Adaptive DVFS and Thread Packing Under Power Caps. In *ACM/IEEE International Symposium on Microarchitecture*, 2011.

[4] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna and C. Le. RAPL: Memory Power Estimation and Capping. *International symposium on Low power electronics and design (ISLPED)*, pages 189–194, 2010.

[5] R. V. Hogg and A. T. Craig. *Introduction to Mathematical Statistics*, 4th edition. New York: Macmillan. 1978.

[6] A. Gandhi, R. Das, J. Kephart, M. Harchol-Balter, and C. Lefurgy. Power Capping Via Forced Idleness. In *Proceedings of Workshop on Energy-Efficient Design*, 2009.

[7] I. Hwang, T. Kam and M. Pedram. A Study of the Effectiveness of CPU Consolidation in a Virtualized Multi-core Server System. *ISLPED*, 339–344, 2012.

[8] J. Koomey. Growth in Data Center Electricity Use 2005 to 2010. Oakland, CA: Analytics Press. August, 1, 2010.

[9] B. Kranz, R. Pike, and E. Hirst. Integrated Electricity Markets in New York. *The Electricity Journal*, 16(2):54 – 65, 2003.

[10] J. Li and J. F. Martinez. Dynamic Power-performance Adaptation of Parallel Computation on Chip Multiprocessors. *International Symposium on High-Performance Computer Architecture (HPCA)*, 77-87, 2006.

[11] K. Ma and X. R. Wang. PGCapping: Exploiting Power Gating for Power Capping and Core Lifetime Balancing in CMPs. *Proceedings of the 21st international conference on Parallel architectures and compilation techniques (PACT), ACM*, 13-22, 2012.

[12] R. Nathuji and K. Schwan. VPM Tokens: Virtual Machine-aware Power Budgeting in Datacenters. *International symposium on High Performance Distributed Computing (HPDC)*, page 119-128, 2008.

[13] *NYISO Day-Ahead Scheduling Manual 11, www.nyiso.com*, June 2001.

[14] A. L. Ott. Experience with PJM Market Operation, System Design, and Implementation. *IEEE Transactions on Power Systems*, 18(2):528–534, 2003.

[15] I. C. Paschalidis, M. C. Caramanis, and B. Li. A Market-based Mechanism for Providing Demand-side Regulation Service Reserves. *Decision and Control and European Control Conference, IEEE*, 2011.

[16] Christian Bienia. Benchmarking Modern Multiprocessors. *Ph.D. Thesis.* Princeton University, January 2011.

[17] PJM. *White Paper on Integrating Demand and Response into the PJM Ancillary Service Markets, www.pjm.com*, Febuary 2005.

[18] *PJM Manual 11: Energy and Ancillary Services Market Operations, www.pjm.com*, June 2013.

[19] *PJM Manual 12: Balancing Operations, www.pjm.com*, Dec. 2012.

[20] K. K. Rangan, G.-Y. Wei, and D. Brooks. Thread Motion: Fine-grained Power Management for Multi-core Systems. In *International Symposium on Computer Architecture*, pages 302–313, 2009.

[21] S. Reda, R. Cochran, and A. K. Coskun. Adaptive Power Capping for Servers With Multi-threaded Workloads. *IEEE Micro*, page 64-75, 2012.

[22] PJM. *Description of Regulation Signals, www.pjm.com*, Dec. 2011.

[23] K. Singh, M. Bhadauria, and S. A. McKee. Real Time Power Estimation and Thread Scheduling via Performance Counters. *SIGARCH Computer Architecture News*, pages 45-55, 2009.

[24] R. Teodorescu and J. Torrellas. Variation-aware Application Scheduling and Power Management for Chip Multiprocessors. *HPCA*, 363–374, 2008.

[25] N. Vasić, D. Novaković, S. Miučin, D. Kostić and R. Bianchini. DejaVu: Accelerating Resource Allocation In Virtualized Environments. In *Proceedings of the 17th international conference on Architectural Support for Programming Languages and Operating Systems. ACM*, 423-436, 2012.