A Hierarchical Demand Response Framework for Data Center Power Cost Optimization Under Real-World Electricity Pricing

Cheng Wang, Bhuvan Urgaonkar, Qian Wang†, George Kesidis‡
Depts. of CSE, MNE†, and EE‡
The Pennsylvania State University

Abstract—**We study the problem of optimizing data center electric utility bill under uncertainty in workloads and real-world pricing schemes. Our focus is on using control knobs that modulate the power consumption of IT equipment. To overcome the difficulty of casting/updating such control problems and the computational intractability they suffer from in general, we propose and evaluate a hierarchical optimization framework wherein an upper layer uses (i) *temporal* aggregation to restrict the number of decision instants during a billing cycle to computationally feasible values, and (ii) *spatial* (i.e., control knob) aggregation whereby it models the large and diverse set of power control knobs with two abstract knobs labeled *demand dropping* and *demand delaying*. These abstract knobs operate upon a fluid power demand. The key insight underlying our modeling is that the power modulation effects of most IT control knobs can be succinctly captured as dropping and/or delaying a portion of the power demand. These decisions are passed onto a lower layer that leverages existing research to translate them into decisions for real IT knobs. We develop a suite of algorithms for our upper layer that deal with different forms of input uncertainty. An experimental evaluation of the proposed approach offers promising results: e.g., it offers net cost savings of about 25% and 18% to a streaming media server and a MapReduce-based batch workload, respectively.**

I. Introduction

Data centers are large and growing consumers of power, in part due to their rapidly growing numbers and sizes to support various forms of cloud and utility computing. The global power demand of data centers grew from 24GW in 2011 to 38GW in 2012 - a growth of 63% within a year [24]. Correspondingly, the power consumption of data centers is an important and growing contributor to their overall *costs*, with large data centers spending millions of dollars per year on their electric utility bills. [1] Optimizing the utility bill (by which we mean trading it off judiciously against performance/revenue) is, therefore, widely recognized as an important problem by cloud providers and other operators of data centers.

The Problem: A large body of work exists in the broad area of power management which data centers can (and should) leverage for optimizing their utility bills. Our focus is on a subset of this work that carries out demand response (DR) by modulating the operation of the data center's IT equipment. Examples of control "knobs" used for such DR include Dynamic Voltage Frequency Scaling (DVFS) for CPUs, server/cluster shutdown, load balancing and scheduling of jobs/requests, partial execution to consume lower power and offer low-quality results, just to name a few. [2] Despite this work, optimizing the utility bill using DR based on IT knobs remains challenging due to:

- *Complexity of IT Knobs:* Most IT knobs offer discrete control options (e.g., DVFS states of a CPU, number of severs to shutdown, particular requests to turn away, etc.) which leads to combinatorial optimization formulations. Their operation is often intimately tied to idiosyncrasies of the software workload (e.g., load balancing or admission control at the unit of a "request," scheduling at the unit of a "job" or a "process," etc.) Finally, they exhibit significant diversity in the temporal (ranging from seconds to hours) and resource granularity (from single thread/CPU to much larger collections of resources) over which they operate. Due to these factors, even though these knobs are very well-understood individually, *employing them collectively* for optimizing a data center's utility bill remains difficult.
- *Complexity of Utility Pricing:* Further contributing to the scalability challenge is the fact that utility bills are computed over significantly long periods of time (typically a month) compared to the time granularity at which most IT knobs operate (seconds or minutes). Furthermore, real-world utility bills for large consumers such as data centers are not merely based on energy consumption over the billing cycle, but involve one (or a combination) of the following: (i) *peak-based pricing*, wherein a component of the electricity bill is dependent

[1]Data centers also incur high costs towards building/maintaining the infrastructure for reliably delivering power to and cooling their equipment. These are one time or occasional investments unlike the recurring utility bills. Whereas our direct focus is on the utility bill, our work is also likely to be of indirect value in reducing the construction time costs (e.g., by allowing a data center to reduce its peak power consumption and hence provision a smaller-sized delivery infrastructure).

[2]Other forms of DR based on non-IT equipment (e.g., smart cooling systems) may also form part of the data center's repertoire. Similarly, numerous techniques for cost optimization have also been explored on the "supply side" such as those based on employing energy storage, distributed generation (including renewables), participation in such programs as ancillary services, etc. We consider these complementary to our focus and outside the scope of the current work.

on the peak power drawn over the billing cycle [4] and (ii) *time-varying pricing*, wherein the per unit energy price fluctuates over time (examples of this operating at different time granularity includes fine time-scale spot prices [21], and higher prices during periods of coincident peaks experienced by the utility [3]).

These factors can combine to result in optimization problem formulations that are (a) computationally intractable [3], and (b) difficult to cast and update upon changes within the workloads, pricing, or IT infrastructure. A significant portion of related work is concerned with optimization problems defined over much shorter time-spans using a small and specific set of knobs, and ignores/simplifies these complexities of pricing, allowing such work to overcome these scalability problems. A small set of papers do look at the utility bill optimization problem but tend to be focus on specific knobs and utility pricing schemes. We discuss these works in Section II.

Research Contributions: Our goal is to devise an optimization framework that can overcome these scalability problems while still being general enough to accommodate both (a) the multitude of IT knobs on which extensive research has been carried out individually, and (b) the idiosyncrasies of real-world utility pricing schemes. We would also like our approach to be capable of handling uncertainty in workloads and electricity prices. Towards this, we make three contributions.

- *Section II*: We propose a hierarchical framework that employs temporal aggregation and spatial (control knob) abstraction to achieve scalability and ease of casting/updating upon changes. The central idea underlying our modeling is our use of the abstract knobs of power demand dropping and delaying to encompass the many IT knobs that have been extensively studied (mostly individually). A second novelty over related work is our incorporation of various features of real world electricity pricing schemes into a single unified formulation. A final benefit of our hierarchical framework is the ease of plugging future power control techniques into the framework due to its modular design and the separation of the abstract knobs and the underlying real knobs.
- *Sections III and IV*: We develop a suite of DR algorithms capable of accommodating different kinds of input uncertainty (both stochastic and completely adversarial). (i) Our first algorithm is based on a stochastic dynamic program (SDP) that is somewhat unconventional due to its incorporation of both average and maximum in its objective (to capture peak-based pricing). (ii) When the data center workload is not amenable to simple stochastic modeling, the resulting SDP becomes computationally difficult to solve. For such a case, we propose a "generalized stochastically bounded burstiness" (gsBB)-based [29] control approach which uses a general queuing model for the workload involving only a few statistical parameters. To our knowledge, this is the first attempt to use such a model for utility bill optimization in data centers (or even for other kinds of utility customers). (iii) Finally, we also develop a fully online algorithm (no prediction or historic data) for the case of a peak-based pricing scheme, and prove a competitive ratio of 2 or better.
- *Section V*: Finally, we study the efficacy of our framework through empirical case studies that use real-world traces for a variety of workloads and utility pricing. Our initial findings are promising: e.g., our approach offers net cost savings of about 25% and 18% to our streaming media server and MapReduce-based batch workloads, respectively. Additionally, our experiments help us understand several subtleties of when and why DR is (or is not) useful for cost-efficacy.

[3]Exceptions do exist and in some cases certain special structural properties allow for scalable optimization formulations - see Section II. We are, however, interested in the general case.

II. Background and Overview

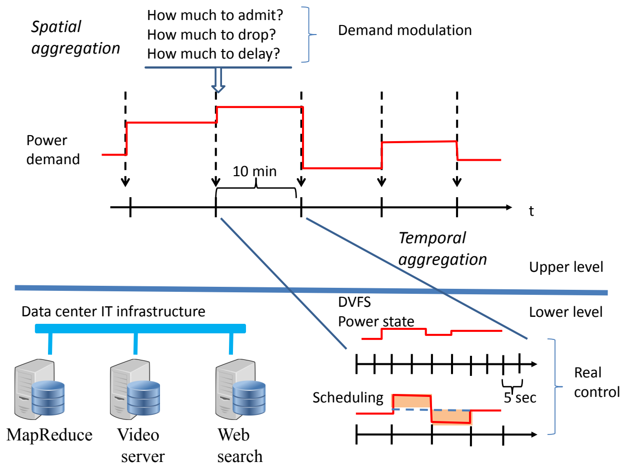

Fig. 1: An overview of our optimization framework.

Overview of Our Approach: We propose a hierarchical optimization framework wherein the "upper" layer (the focus of this paper) carries out decision-making based on two complementary forms of aggregation. First, it employs temporal aggregation whereby it restricts the number of decision instances over the billing cycle to computationally feasible values. Second, it employs spatial (i.e., control knob) aggregation whereby it replaces the large and diverse set of IT knobs available to the data center with two abstract DR knobs labeled demand dropping and demand delaying that operate upon a *fluid* power demand. Figure 1 presents the main ideas underlying our approach.

The key insight underlying our modeling is that *despite their immense diversity, all IT knobs for DR can be viewed as having the effect of either reducing (i.e., dropping) or* *postponing (i.e., delaying) a portion of the power consumption* (or a combination of these effects). To appreciate this, consider the following concrete examples:

- *Demand Dropping*: A Web search application employing the knob of partial execution for some of its queries for meeting response time targets [27] is an example of a workload that is delay-sensitive and can drop some of its power needs (relative to that corresponding to the best quality results) to meet delay targets. Another example is a video server that exploits multiple fidelity videos that MPEG allows to guarantee fluent streaming [7] by sending lowering the video quality during overloads (with lower power consumption).
- *Demand Delaying*: For several batch workloads (e.g., a MapReduce-based set of tasks [28]), dropping is intolerable but delaying demand as long as it finishes before a deadline (typically with sufficient flexibility for scheduling) is acceptable.
- *Both Dropping and Delaying*: Some applications can have combinations of the above two. E.g., a search engine typically has a user-facing Web front-end as well as a backend component that does crawling and indexing, with these two components' power demand modulated via dropping and delaying, respectively.

Since many data center workloads exhibit uncertainty [18] as do electricity prices [14] we devise DR techniques that can adapt their behavior to workload and price evolution in an *online* manner. We devise three such control algorithms for sequential decision-making in Section IV. Our algorithms differ in the assumptions they make about input predictability (with correspondingly different efforts involved in "learning" their respective input models) and offer different computational complexities (i.e., scalability). A data center might find one of these more appropriate than others based on how this trade-off between input prediction accuracy and computational complexity applies to it.

Finally, the "bottom" layer of our hierarchical framework employs techniques from existing work to enforce the power modulation decisions made by the upper layer. To illustrate our approach, in Section V, we use existing work on power/performance trade-off associated with admission control and MPEG video quality modulation (for our streaming media server workload).

Related Work: We discuss three classes of related work. *Salient Power Management Knobs*: A large body of literature focuses on exploiting IT control knobs to manage/reduce the "raw" power consumption of data centers. Examples of such work include the use of DVFS [9], DRPM [10], server shutdown [2], VM consolidation [6], partial execution [27], load balancing [15], and combinations of these knobs [17]. The efficacy of these control knobs has been explored individually by numerous research papers. However, these control knobs exhibit significant diversity in temporal granularity (e.g, DVFS at seconds, DRPM and server shutdown at minutes, etc) and control unit (e.g., load balancing at the unit of a "request," consolidation at the unit of a VM, etc), which adds to the complexity of employing them collectively. These are complementary to our work in that they provide salient ways for the lower level of our framework for translating our abstract control decisions.

Other Examples of Hierarchical Control: Power management in a hierarchical way is not new in the literature. Certain existing pieces of work have (either implicitly or explicitly) taken similar approaches (framework design), although not necessarily aiming at utility bill optimization. For example, Raghavendra et al. proposes a hierarchical power capping framework for data centers which coordinates different power control techniques [17]. Kontorinis et al. presents a hierarchical architecture for distributed UPSs to shave the power spikes of data centers [12]. Similar designs can also be found in [26]. Common across all these works is the use of "power budgets/caps" that the higher levels determine and pass on to the lower levels that implement them. Our abstraction is, in fact, a generalization of the power budgets used in these works because, in addition to specifying an upper bound on the power consumption during a slot (as in these works), we also specify how to achieve it via our combination of dropping and delaying.

Utility Bill Optimization in Data Centers: Most closely related to our work are some recent efforts that focus on the exact problem of utility bill optimization for the data center. More mature among these are papers that consider time-varying electricity pricing. For example, Urgaonkar et al. develop an Lyapunov optimization based technique to minimize operational cost using batteries [22]. Liu et al. propose distributed algorithms for geographical load balancing to exploit both the spatial and temporal aspects of time-varying pricing [14]. The time-varying pricing structure enables them to do DR within a single control window, or to minimize the long-term expected costs wherein the complexity of the optimization problem can be reduced greatly with minimum optimality loss. On the other hand, peak-based electricity pricing, which is commonly seen in the real-world, has received less attention. The difficulty of DR under peak-based pricing is that the peak power consumption remains unknown until the end of the month, which results in scalability issues if DR is done at a fine time-scale. Some existing works avoid this difficulty by considering only a short planning period. For example, Liu et al. consider a hybrid of peak-based and time-varying pricing together with coincident peak charge and reduce power costs using robust optimization [14]. Xu et al. exploit the cost saving potential of partial execution for the web search requests under peak-based pricing [27]. However, both these works are only evaluated for a week-long horizon, and might not be scale to a month-long billing cycle. Furthermore, they focus on either a single IT knob or external energy source, whereas our goal is to accommodate a more general set of knobs. .

III. Problem Formulation

In this section, we formulate the optimization problem that the upper layer of our hierarchical framework solves.

Input Parameters: We consider a slotted model wherein control decisions are made at the beginning of *control windows* of equal duration δ. We denote by T the number of such control windows within a single billing cycle, which constitutes our *optimization horizon*. Let the time-series $\{p_t : 1 \leq t \leq T\}$ denote the power demand of the data center over the optimization window of interest, with $0 \leq p_t \leq p_{\max}$ denoting its power demand during the t^{th} control window. Here $p_{\max}$ denotes the maximum power demand the data center may pose during a single control window. To represent the time-varying pricing employed in many electric tariffs (e.g., [19]), we define a time-varying energy price of α_t dollars per unit energy (in units of \$/kWH). To represent peak-based charging (e.g., [4], [8], [27]), we define a peak price of β dollars per unit peak power per billing cycle (in units of \$/kW). Based on evidence in existing work, we employ (i) a convex non-decreasing function to model the revenue loss due to demand dropping ($l_{\mathsf{drop}}(\mathsf{demand})$), and (ii) a linear non-decreasing function to model revenue loss (per slot of delay) due to demand delaying ($l_{\mathsf{delay}}(\mathsf{demand}) = k_{\mathsf{delay}} \cdot \mathsf{demand}$, $k_{\mathsf{delay}} > 0$), where demand is in units of power [7], [15].

Decision Variables: At the beginning of control window t (or simply t henceforth), the data center can have "residual" power demands due to delaying in the past (in addition to the newly incoming demand p_t). Let r_t denote the aggregate residual demand at the beginning of t. Denote as $y_{\max}$ the peak demand admitted in any window during $[1, ..., T]$. The control actions during t involve admitting, postponing, and dropping portions of r_t and p_t. We denote as d_t and a_t the demand that is dropped and admitted out of p_t respectively, ${a'}_t$ and ${d'}_t$ the demand that is dropped out of r_t. Finally, we denote as $\mathcal{A}$ the set $\{a_t\}_{t\in[1,T]}$, $\mathcal{A}'$ the set $\{{a'}_t\}_{t\in[1,T]}$, $\mathcal{D}$ the set $\{d_t\}_{t\in[1,t]}$, and as $\mathcal{D}'$ the set $\{{d'}_t\}_{t\in[1,t]}$ of decision variables.

Objective: We choose as our data center's objective the minimization of the sum of its utility bill and any revenue loss resulting from DR $\mathcal{O}(\{\mathcal{A}\}, \{\mathcal{D}\}, \{\mathcal{A}'\}, \{\mathcal{D}'\})$: $\beta y_{\max} + \sum_t \Big(\alpha_t(a_t + {a'}_t) + +l_{\mathsf{drop}}(d_t) + k_{\mathsf{delay}}{a'}_t\Big)$.

Notice that our objective does not have an explicit revenue term which might give the impression that it does not capture the data center's incentive for admitting demand. It is important to observe that the incentive for admitting demand comes from the costs associated with dropping or delaying demand. One would have $l_{\mathsf{drop}}(a) > \alpha_t(a) \cdot a$ to disallow scenarios where the data center prefers dropping all demand to admitting it. However, there can be situations (e.g., extremely high energy prices during coincident peaks [3]) when this is not so. Finally, we could also include a revenue model of accepted demand, simply resulting in $\alpha_t(.) < 0$ in our problem formulation.

Constraints: The aggregate demand in the data center at the beginning of t is given by the new demand p_t, and any demand unmet so far (deferred from previous time slots) r_t. Since this demand must be treated via a combination of the following: (i) serve demand a_t and ${a'}_t$, (ii) drop demand d_t and ${d'}_t$, and (iii) postpone/delay demand (r_{t+1}) (to be served during $[t + 1, ..., T]$), we have:

$$(p_t - a_t - d_t) + (r_t - {a'}_t - {d'}_t) = r_{t+1}, \ \forall t. \tag{1}$$

Any residual demand or new demand that is not admitted or dropped during t is postponed to the next time slot:

$$r_t - {a'}_t - {d'}_t \geq 0, \ \forall t. \tag{2}$$

$$p_t - a_t - d_t \geq 0, \ \forall t. \tag{3}$$

To keep our problem restricted to one billing cycle, we add an additional constraint that any delayed demand must be admitted by the end of our optimization horizon:

$$r_{T+1} = 0. \tag{4}$$

Alternate formulations that minimize costs over multiple billing cycles may relax the constraint above. The peak demand admitted $y_{\max}$ must satisfy the following:

$$y_{\max} \geq a_t + {a'}_t, \ \forall t. \tag{5}$$

Finally, we have:

$$y_{\max}, a_t, d_t, {a'}_t, {d'}_t, r_t \geq 0, \ \forall t. \tag{6}$$

Offline Problem: Based on the above, we formulate the following offline problem (called OFF):

$$\begin{aligned} &\text{Minimize} && \mathcal{O}(\{\mathcal{A}\}, \{\mathcal{D}\}, \{\mathcal{A}'\}, \{\mathcal{D}'\}), \\ &\text{subject to} && (1) - (6). \end{aligned}$$

OFF will serve as a baseline against which we compare the efficacy of our stochastic/online algorithms in Section V. Owing to the following lemma, we can simplify our problem formulation by setting ${d'}_t = 0, \forall t$:

Lemma 1. *If* $l_{\mathsf{delay}}(.) > 0$*, then there always exists an optimal solution of* OFF *that never postpones some demand only to drop it in the future.*

Our proof is based on a relatively straightforward contradiction-based argument. We omit it here for space and present it in a technical report [25].

Discussion: We point out some limitations and possible generalizations of our formulations below:

- A key assumption in our modeling is what one might call "conservation" of power demand (as captured by constraint (1)). E.g., we do not capture possibilities such as delaying of some power demand resulting in a change in overall power needs. Also, we assume that a power demand modulation decision does not affect the future external demand process. E.g., a request that is turned away (due to some power demand being "dropped") does not return. Finally, we work with a $l_{\mathsf{delay}}(\mathsf{demand})$ function that is linear in the demand delayed per slot. A non-linear $l_{\mathsf{delay}}(\mathsf{demand}, \mathsf{delay})$ would require that all the "residual" power demands that are postponed till a time slot and have not been admitted so far have to be recorded as state variables such that the delay costs can be evaluated based on their individual delay, which makes the framework unscalable. We make all of these simplifications for scalability (one of our primary goals), and leave it to future work to evaluate their impact on the efficacy of our approach.

- Our problem formulation is general enough to incorporate the so-called "coincident peak" as in [14]. This would be done by appropriately setting the α_t values for the windows when energy prices go up due to the occurrence of a coincident peak.
- Finally, one key benefit of our approach is the ease of casting (or re-casting upon changes in the environment) the overall optimization problem. In particular, the hierarchical nature of our approach implies that any changes only necessitate updates to the lower layer translations associated with these changes rather than the overall optimization framework. Section V implicitly illustrates this aspect of our approach.

IV. Decision-Making Under Uncertainty

A. Stochastic Dynamic Program

Since data center workloads can often be captured well via statistical modeling techniques [14], we develop a stochastic dynamic programming (SDP) that leverages such modeling.

We choose as the goal of our SDP the minimization of the *expectation* of the sum of the data center's electricity bill and demand modulation penalties over an optimization horizon, and denote it $\bar{\mathcal{O}}(\{\mathcal{A}\}, \{\mathcal{A}'\}, \{\mathcal{D}\})$ building upon the notation used in Section III. One key difficulty in our formulation arises due to the somewhat unconventional nature of our SDP wherein our objective is a *hybrid* of additive (total energy) and maximal (peak power) components.[4] Consequently, we define as our state variable a combination of the residual demands r_t and the peak power admitted so far. Using y_t to denote the peak demand admitted during $[1, ..., t-1]$, we represent the state at time t as the 2-tuple $s_t = (r_t, y_t)$. The peak demand at the end of the optimization horizon y_{T+1} then corresponds to the variable $y_{\max}$ employed in OFF.

Since the power demands arriving in different control windows are only known stochastically, we introduce the notation P_t to denote these random variables, and use p_t to denote a particular realization of P_t. Correspondingly, we denote by $P_{[t]} = (P_1, ..., P_t)$ the history of the demand up to t, and by $p_{[t]} = (p_1, ..., p_t)$ its particular realization. We assume that the conditional probability distribution of the power demand P_t, i.e., $\mathsf{Pr}\{P_t = p_t | P_{[t-1]} = p_{[t-1]}\}$, is known. Similar notations can be defined to handle the uncertainties in electricity price; we omit modeling electricity price as random variables for simplicity and present the full version with electricity price in [25].

Bellman's optimality rules for our SDP can now be written as follows. For the last control window, we solve the following optimization problem (we denote it as SDP(T)) which gives $V_T(s_T, p_{[T-1]})$:

$$\min_{a_T, a'_T, d_T} \mathbb{E}\Big\{\beta y_{T+1} + \alpha_T(a_T + a'_T) + l_{\mathsf{drop}}(d_T) + k_{\mathsf{delay}} a'_T \;\mid\; P_{[T-1]} = p_{[T-1]}\Big\},$$

Subject to:

$$\begin{aligned} &p_T - a_T - d_T = 0;\; r_T - a'_T = 0,\\ &y_{T+1} \geq y_T;\; y_{T+1} \geq a_T + a'_T,\\ &a_T, a'_T, y_{T+1}, d_T \geq 0. \end{aligned}$$

For control windows $t = T-1, ..., 1$, we solve SDP(t) to obtain $V_t(s_t, p_{[t-1]})$:

$$\min_{a_t, a'_t, d_t} \mathbb{E}\Big\{\alpha_t(a_t + a'_t) + l_{\mathsf{drop}}(d_t) + k_{\mathsf{delay}} a'_t + V_{t+1}(s_{t+1}, p_{[t]}) \;\mid\; P_{[t-1]} = p_{[t-1]}\Big\},$$

Subject to:

$$\begin{aligned} &(p_t - a_t - d_t) + (r_t - a'_t) = r_{t+1},\\ &r_t - a'_t \geq 0;\; p_t - a_t - d_t \geq 0,\\ &y_{t+1} \geq y_t;\; y_{t+1} \geq a_t + a'_t,\\ &a_t, a'_t, d_t, y_{t+1}, r_{t+1} \geq 0. \end{aligned}$$

This SDP offers a runtime of $O(R \cdot L_p^5 \cdot T)$ under the assumptions of stage-independence for demands and denoting as $O(R)$ the runtime of a sub-problem SDP(t) (L_p is the number of discretization levels for power demands).

Special Case of Only Dropping: For some workloads, delaying is not tolerable and dropping is the only option. E.g., consider a Web search engine which is delay-sensitive but can do partial execution of search queries to save energy as long as response time guarantees are maintained [27]). For such cases, the state simplifies even further to $s_t = y_t$ and the set of control variables shrinks to only d_t to describe dropping. This specialized SDP (that we label SDP_{Drop}) has a reduced runtime of $O(R \cdot L_p^3 \cdot T)$. More interestingly, the structure of the optimal policy for SDP_{Drop} given by the following lemma (proof in technical report [25]) provides a way of explicitly finding the optimal control policy. Define $\mu_t = \frac{a_t}{p_t}, \mu_t \in [0,1]$, $\mathcal{G}_t(\mu_t) = \mathbb{E}\{\alpha_t(\mu_t p_t) \times \mu_t p_t + l_{\mathsf{drop}}(p_t - \mu_t p_t) + V_{t+1}(\max\{\mu_t p_t, y_t\})\}$ and $\phi_t(y_t) = \arg\min_{\mu_t \in \mathcal{R}^+} \mathcal{G}_t(\mu_t)$.

Lemma 2. *If the demands p_t are independent across t, SDP_{Drop} has the following threshold-based optimal control*

[4]This represents an important difference from other existing SDP formulations in this area, such as [23], where their objective is optimizing a purely energy-based cost.

policy:

$$(a_t^*, d_t^*) = \begin{cases} (\phi_t p_t, p_t - \phi_t p_t), & \text{if} \phi_t \leq 1 \\ (p_t, 0), & \text{if} \phi_t > 1 \end{cases}$$

B. gSBB-based Control

When the data center workload is not stage-independent or is not even modeled well by a Markov process, the resulting SDP becomes computationally difficult to solve. For such cases, we propose a gSBB-based control approach which uses a general queuing model for the workload involving only a few statistical parameters.

Let μ be the mean of the data center's "raw" power demand. Here we consider more general demand modulation wherein units of demand will drop out and not return if they are deferred (or to be deferred) by more than τ^* time slots. In addition to μ and τ^*, suppose the "raw" demand is also modeled by a "generalized stochastically bounded burstiness" (gSBB) curve [29] $\{(\gamma, \phi(\gamma\tau^*)) | \gamma > \mu\}$; i.e., a queue whose arrivals are the "raw" demands and is served at rate γ will have backlog Q_γ such that

$$Pr(Q_\gamma \geq \gamma\tau^*) \leq \phi(\gamma\tau^*)$$

The **basic procedure** of this gSBB-based control is as follows (see Figure 2): The data center as a "server" in the gSBB queuing system runs at a maximum service rate γ, which is equivalent to regulating the peak power consumption to be within γ. Upon new arrival, if the queue is empty, the new power demand can be admitted directly. If the queue backlog is $\gamma\tau^*$, it would take the data center τ^* time slots to service all the backlog demand, and new demand will have to be dropped since it needs to wait for τ^* time slots has to be serviced. Otherwise, the new demand can enter the queue and be served within τ time slots in an FIFO order.

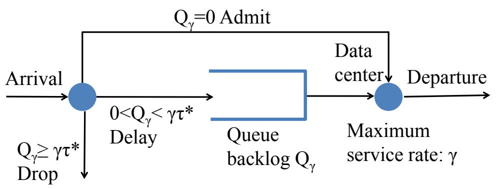

Fig. 2: gSBB-based control loop.

Note that the peak rate (and peak power consumption henceforth) of the queue-output is limited to γ. Under the peak-based pricing scheme, the objective of the top level demand modulation becomes one of selecting γ to minimize the expected cost over a billing cycle:

$$\min_{\gamma \geq 0} T\mu \int_0^\infty C(\tau) dF_\gamma(\tau) + \beta\gamma$$

where $T\mu$ is the expected total demand obtained from historical data. We define $C(\tau)$ as the "price/cost" (including energy cost, delay cost and dropping cost) imposed upon one unit of demand with delay τ. For any unit of demand, if it is not delayed, the "price/cost" is equivalent to the energy cost α; if it is delayed for τ time slots, the unit "price/cost" of $\alpha + k_{delay}\tau$ (or other forms of delay penalty) will be incurred for linear delay cost; if it is dropped, the "price/cost" will be k_{drop}. Here $C(\tau)$ can be viewed as an approximation for the actual unit "price/cost."

We define Δ_γ as a random variable for the delay of unit demand given that the service rate of the departure process is γ, and denote as $F_\gamma(\tau)$ the **CDF** of Δ, therefore

$$F_\gamma(\tau) = Pr(\Delta_\gamma \leq \tau) = Pr(\frac{Q_\gamma}{\gamma} \leq \tau) = 1 - \phi(\gamma\tau)$$

wherein the second equality holds since new demand will have delay $\frac{Q_\gamma}{\gamma}$ when the backlog is Q_γ. Then $dF_\gamma(\tau) = -d\phi(\gamma\tau)$. The objective cost function becomes:

$$\min_{\gamma \geq 0} -T\mu \int_0^\infty C(\tau) d\phi(\gamma\tau) + \beta\gamma$$

in which $C(\tau)$ and τ^* are pre-determined by the specific application running in the data center; $\phi(\gamma\tau)$ can be obtained from historical data by profiling $\phi(\gamma\tau^*)$. [5]

Implementation Considerations: To implement our gSBB-based control, the first step is to profile $\phi(\gamma\tau^*)$. Choose γ from $(0, p_{max}]$ and τ^* from $[0, \tau_{max}]$ and obtain a table of $\phi(\gamma, \tau^*)$ by repeatedly performing the **basic procedure** of gSBB-based control using given γ, τ^* and historical data. Figure 3 shows an example of profiling $\phi(\gamma\tau^*)$. The next step is to solve the above offline optimization problem. The solution (γ) of the problem together with τ^* determined by the application can be used to perform gSBB-based control on the new demand. Note that the gSBB-based control can also be done in a partially online fashion by repeating the above steps with most recent historical data periodically.

Discussion: One important advantage of gSBB-based control over SDP is scalability. Recall that to keep the top level of our demand modulation framework scalable, OFF and SDP have to resort to a linear approximation of the delay cost. However, the gSBB-based formulation is able to incorporate any form of delay cost by defining $C(\tau)$ appropriately while still being solved efficiently.

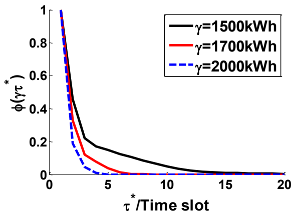

Fig. 3: Example of $\phi(\gamma, \tau^*)$ profiled for Google trace in Section V.

[5]Note that the $\beta\gamma$ term approximates a peak power (γ) penalty. The γ term here could be replaced instead by the approximation $(\mu + 2\sigma)(1 - \phi(\tau\gamma)) + 2\hat{\sigma}$, i.e., mean power consumed plus two standard deviations (by an abuse of PASTA and "fluctuations increase delay," the latter implying the standard deviation of the departures may be greater than than of the arrivals), where σ is the standard deviation of the incident power demand process (arrivals).

C. A Fully Online Algorithm

We now present a fully online algorithm labeled ON_{Drop} and competitive analysis for it for the special case where only power demand dropping is allowed. Notice that the resulting problem setting is identical to that considered for SDP_{Drop} with the difference that whereas ON_{Drop} assumes adversarial power demand inputs, SDP_{Drop} was designed for power demands that can be described stochastically. The following lemma is key to the design of ON_{Drop}:

Lemma 3. *If only demand dropping is allowed, and $l_{drop}(x) = k_{\mathsf{drop}}x$, the optimal solution of* OFF *has a demand dropping threshold θ of the following form: if we denote $\hat{p}_t$ as the t^{th} largest demand value in $\{p_t\}_{t=1}^T$ and $\hat{\alpha}_t$ the corresponding energy price, then $\theta = \hat{p}_n$, where $\beta - \sum_{t=1}^{n-1}(k_{\mathsf{drop}} - \hat{\alpha}_t) \geq 0$ and $\beta - \sum_{t=1}^{n}(k_{\mathsf{drop}} - \hat{\alpha}_t) \leq 0$. As a special case, if $\alpha_t = \alpha$ for all t (pure peak-based pricing), $n = \lceil \frac{\beta}{(k_{\mathsf{drop}} - \alpha)} \rceil$.*

The proof of Lemma 3 implies that the optimal demand dropping threshold of OFF when only dropping is allowed can be found in an efficient way. If we can keep track of the $\hat{p}_n$ in an online fashion, finally we can find the optimal demand dropping threshold after observing all the demand in the optimization horizon. So we exploit Lemma 3, to devise ON_{Drop} as follows:

```
ON_Drop: Online Algorithm With Only Dropping.
1: Initialization: Set demand dropping threshold $\theta_0 = 0$.
2: At time $t$, sort $p_1, ..., p_t$ into $\hat{p}_1, ..., \hat{p}_t$ such that $\hat{p}_1 \geq \hat{p}_2 \geq ... \geq \hat{p}_t$.
3: Update $\theta_t$ as follows: find index $n$ such that $\beta - \sum_{m=1}^{n-1}(k_{\mathsf{drop}} - \hat{\alpha}_m) \geq 0$ and $\beta - \sum_{m=1}^{n}(k_{\mathsf{drop}} - \hat{\alpha}_m) \leq 0$; set $\theta_t = \hat{p}_n$.
4: Decision-making: Admit $\min(p_t, \theta_t)$, drop $[p_t - \theta_t]^+$.
```

For the special case of $\alpha_t = \alpha$ for all t (peak-based pricing), since $n = \lceil \frac{\beta}{(k_{\mathsf{drop}} - \alpha)} \rceil$ according to Lemma 3, **step 3** simply becomes: if $t < n$, $\theta = 0$; Otherwise, $\theta = \hat{p}_n$.

Theorem 1. ON_{Drop} *offers a competitive ratio of* 2 *under peak-based pricing.*

Furthermore, ON_{Drop} offers a competitive ratio of $2 - \frac{1}{n}$ under peak-based pricing when $\alpha_t = \alpha$ for all t. We present our proofs in [25]. One useful way of understanding ON_{Drop} is to view it as a generalization of the classic ski-rental problem [11]: increasing the dropping threshold is analogous to purchasing skis while admitting/dropping demands according to the existing threshold is analogous to renting them. The generalization lies repeating the basic ski-rental-like procedure after every n slots.

Implementation Considerations:. At the **Initialization** step of ON_{Drop}, the demand dropping threshold is set to 0, which will not be changed until $t \geq n$. However, this is not practical in the real world (all the power demand before $t \geq n$ is dropped) and results in the inefficiency of ON_{Drop}. In fact, to reduce a monthly electricity bill, we can perform ON_{Drop} upon the most recent historical power demand data over a short period of time (e.g., one-day dataset) to obtain a better initial demand dropping threshold. We will evaluate this improved algorithm as $\text{ON}^*_{\text{Drop}}$ in Section V.

V. Empirical Evaluation

In this section, we evaluate our approach using real-world workload power traces and utility prices. As described in Section II, since several prior works have looked at DR for time-varying utility prices, we restrict the focus of our evaluation to peak-based pricing (while noting that our framework is general enough to also capture time-varying prices including coincident pricing as in [14]). We present our experimental evaluation with time-varying utility pricing in our technical report [25].

A. Experimental Setup

Demand Traces: We employ three real-world demand traces (spanning 30 days) adopted from recent studies:

- MediaServer: a service request trace [13] from a live video streaming system which provides MPEG-4 video to its clients. For simplicity, we assume that the system sends a 10-min MPEG-4 video to incoming clients at one of 3000 kb/s or 1500 kb/s (MPEG-4 standard [16]) during a control window. We transform the MPEG-4 bit rate into power consumption based on data reported in [20] and generate the power demand trace for MediaServer.
- Facebook: a power demand trace based on data reported from a Facebook data center [1] which runs MapReduce batch jobs.
- Google: a power demand trace based on data reported from a Google data center [5] which runs a mixture of MapReduce, Webmail, and Web search jobs.

Additionally, we create a Synthetic power demand series with an emphasis on including an unpredictable surge in power demand (e.g., as might occur due to a flash crowd), which is built by adding a high power surge to the demand for MediaServer on the 15th day. We show these power demand traces in Figure 4. Note that all of these real-world traces (except for Synthetic) exhibit a strong *time-of-day* behavior. To help understand the features of our workloads, we select two workload properties that we intuitively expect to be informative about achievable cost savings: (i) the *peak-to-average ratio* (PAR), which captures the peak-shaving potential of the workload and (ii) *peak width* (P_{70}), which is defined as the percentage of the time slots in which the power demand value is larger than 70% of the peak power demand. We show these parameters in Fig. 4.

Parameter Settings: We set the energy price $\alpha = 0.046\$/kWh$ and the peak power price $\beta = 17.75\$/kW$

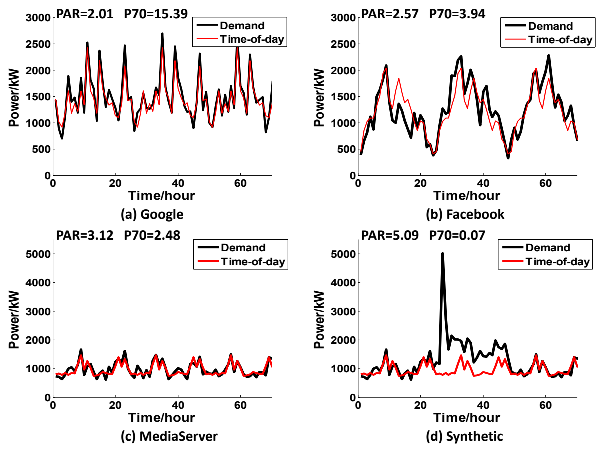

Fig. 4: Workloads from the 14th day to the 16th day. PAR: peak-to-average ratio; P_{70}: peak width.

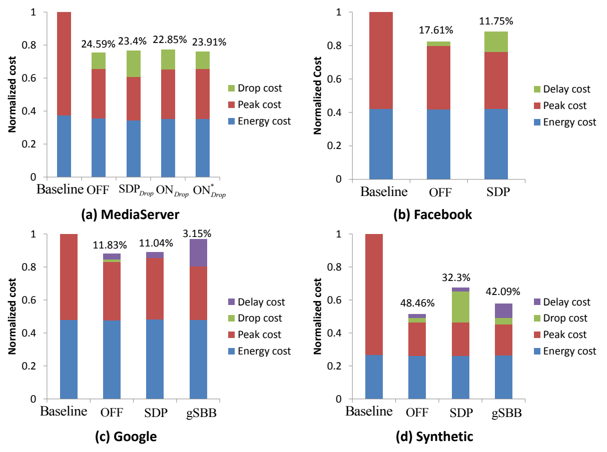

Fig. 5: Cost compositions of various algorithms.

based on the Duke electric utility pricing [4]. We work with a monthly electricity billing cycle, wherein the control window (one time slot) spans 10 minutes, so there are 4320 time slots in the optimization horizon. We set $k_{\text{delay}} = 0.02\$/kWh$ based on [15], and $k_{\text{drop}} = \alpha + k_{\text{delay}}\tau^* = 0.246$, wherein $\tau^* = 10$ is the maximum delay allowed in the gSBB-based control. Our choice of k_{drop} implies that it would be better to drop demand than to delay the demand for more than τ^* time slots. Because of the strong time-of-day behavior seen in the traces above, we use this time-of-day plus a zero-mean Gaussian noise in SDP and SDP_{Drop}. For $\text{ON}^*_{\text{Drop}}$, we train the algorithm using the demand of the first two days to obtain the initial dropping threshold, and then run for the remaining 28-days.

Our Evaluation Metric and Baseline: Our baseline is the case with no demand modulation on the workload, thereby no revenue/performance loss attached to the power costs. The metric we use to evaluate our framework is **cost saving** which is the percentage reduction in costs due to a particular algorithm compared to our baseline.

B. Cost Benefits

We show the cost compositions and savings for both real-world and synthetic traces in Figure 5. First, let us consider the MediaServer workload. During peak hours the streaming media system can either send low quality video or reject new clients both of which amount to power demand dropping. We find that both stochastic control and online control achieve near-optimal cost-savings (up to 25%) with revenue loss, with $\text{ON}^*_{\text{Drop}}$ providing a slightly higher cost-saving because the demand dropping threshold is trained using historical data.

Next, let us look at the Facebook workload (Figure 5(b)), wherein dropping MapReduce batch jobs is not acceptable. However, the data center can delay demand via batch job scheduling or DVFS (or other techniques which have similar effects). We find that SDP provides 11.75% lower costs compared to OFF. At the same time, inappropriate demand delaying during non-peak time results in the inefficiency (and higher delay-related costs) of SDP.

Moving to the Google workload (Figure 5(c)), both of our abstract knobs are employed since the MapReduce jobs can be delayed and Web search engine can reduce power consumption via demand dropping (e.g., partial execution of search requests). We find that SDP provides near-optimal cost savings whereas gSBB provides little cost savings due to much larger demand being delayed. We also notice that neither gSBB nor SDP drop demand whereas a small portion of the "raw" demand is dropped in OFF.

To see how our framework and algorithms work when there is unpredictable flash crowd, we evaluate the cost savings with Synthetic with both abstract knobs. In this case, SDP is not able to provide near-optimal cost saving whereas gSBB provides much better cost savings.

Finally, we observe that the total energy consumption in all cases are very similar before (i.e., with our baseline) and after DR. This means that the real peak over the billing cycle, which occurs rarely in the real-world, can be shaved by using our framework to greatly save the power costs without resulting in significant revenue loss (e.g., users might not come back once rejected ("raw" demand dropped)), which is true even when the only available knob is to drop (MediaServer). A second observation is that although OFF gives the optimal *costs savings*, it does not always have the least pure power costs, which is as expected since our goal is to optimize the total costs (including those due to revenue loss) instead of just the utility bill. This can also be explained by the (possibly) wrong decision makings (dropping more demand than the optimal solution during non-peak time) of other stochastic/online algorithms.

Key insights: (i) Under peak-based pricing, our approach is able to provide significant cost savings (up to

25% for MediaServer) for real-world workloads without losing much "raw" demand. (ii) Our stochastic control and online control approaches achieve near-optimal cost-savings; however, they are not so beneficial when there is unexpected flash crowd (Synthetic), whereas the gSBB-based control is still able to handle workload unpredictability and provide cost-savings. (iii) In terms of pure power costs, OFF does not always have the least costs since other control approaches might drop/delay more power demand (with higher performance/revenue loss) than the optimal solution and therefore incur lower pure power costs.

C. A Closer Look At Control Decisions

We take a closer look at the actual IT knob control decisions (i.e., client-level admission control and MPEG-4 modulation) for the experiment involving our MediaServer workload trace. Our translation is based on the following simple heuristic: degrade the Quality-of-Service (from 3000 kb/s to 1500 kb/s) to meet the demand dropping decision as much as possible; otherwise, reject some of the client requests to compensate for the target dropping power that is unmet. We show the real control decisions of our algorithms on the fifth day in Figure 6. We find that the control approaches of our framework make decisions similar to those made by OFF during the peak hour: all the clients are serviced via a lower bit rate; some clients are turned away. We also find that $\mathrm{ON}_{\mathrm{Drop}}$ drops more demand than $\mathrm{ON}^*_{\mathrm{Drop}}$ during the first 40 time slots from Figure 6(c)(d), which is in consistent with our expectation in Section IV that $\mathrm{ON}^*_{\mathrm{Drop}}$ provides better cost saving than $\mathrm{ON}_{\mathrm{Drop}}$ by choosing a higher initial demand dropping threshold.

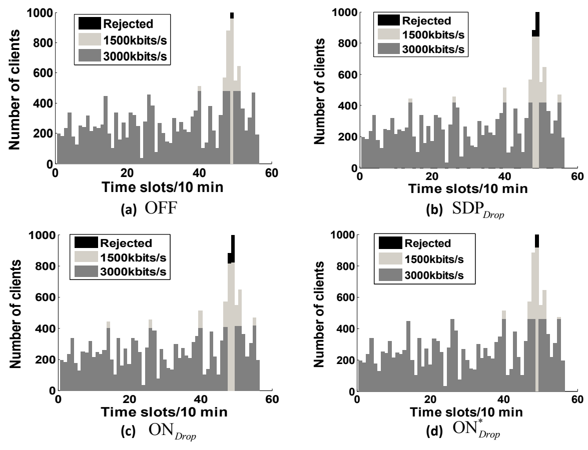

Fig. 6: An example of control decisions of real knobs corresponding to demand dropping.

D. Impact of Workload Properties

In this section, we try to evaluate the impact of workload properties on cost savings under peak-based pricing. Since we only focus on the *shape* of the power demand traces, we ignore the real-world implications of those traces and evaluate the cost-savings offered by different algorithms with different control on all the real-world traces as well as on the synthetic trace. We show the results in Table I.

Let us first consider only using dropping. According to Lemma 3, the optimal demand dropping threshold under peak-based pricing is determined by $\hat{p}_n, n = \lceil \frac{\beta}{(k_{\text{drop}}-\alpha)} \rceil$ in the non-increasing array of $\{\hat{p}_t\}_{t=1}^T$. Since n is fixed for all workloads given the pricing parameter settings, a larger P_{70} implies higher probability of the optimal dropping threshold (θ) being high, which means less power demand can be dropped to reduce the peak demand with lower cost savings. Among our workloads, Google has very "wide" peak/near-peak power values, which is the reason for the meager 7.52% cost savings. On the other hand, Synthetic exhibits "sharp" and "tall" peaks, for which a greater cost saving (45.01%) is possible.

Next, let us consider only using delaying. Deferring demand is possible only when a near-peak demand p_t is followed immediately by much lower power demands, and the costs saving due to the resulting peak reduction is larger than the delay cost incurred. The lower and longer these succeeding low demand periods are (typically for larger PAR and smaller P_{70}), the better should be the cost savings achieved. Our experiment results verify these intuitions. As shown in Figure 4, PAR increases drastically from Google, Facebook, MediaServer, Synthetic (in that order) while P_{70} decreases, and correspondingly, the cost savings improve (from 11.52% for Google to 42.47% for Synthetic).

Thirdly, let us consider the cost savings when both knobs are allowed: why are the cost savings very similar to those when only delaying is allowed? If a unit of delayed demand can be serviced with a small delay, the cost incurred is small compared to that for dropping. The low demand periods within short interval from near-peak demands in our workloads are plentiful, making dropping a rarely used knob. Only for Synthetic do we find that dropping demand helps improve cost saving by a non-trivial amount: from 42.47% with only delaying to 48.46% with both the knobs.

Finally, let us consider how our control algorithms perform. SDP and $\mathrm{SDP}_{\mathrm{Drop}}$ works well under all workloads, since the workloads exhibit strong time-of-day behavior, implying our SDP has a reliable prediction model to work with. However, we observe that the cost saving from SDP for Synthetic becomes negative (-3.68%) when only delaying is employed, which suggests stochastic model is not suitable for capturing unexpected flash crowd. The gSBB-based control, although not very effective for Google and Facebook, provides near-optimal cost savings for MediaServer and even for Synthetic despite the unpredictable peak in the workload. $\mathrm{ON}_{\mathrm{Drop}}$ performs well for almost all workload/control-knob combinations, except the 4.54% for Google with only dropping. Note that, in this case, there isn't much room for savings to begin with (the optimal

	Drop				Delay		Drop+Delay		
workload	OFF	SDP_{Drop}	ON_{Drop}	$\text{ON}^*_{\text{Drop}}$	OFF	SDP	OFF	SDP	gSBB
Google	7.52	7.5	4.54	6.28	11.52	11.04	11.83	11.04	3.15
Facebook	18.21	18.15	15.15	16.36	17.61	11.75	19.21	9.44	6.31
MediaServer	24.59	23.4	22.85	23.91	29.21	26.69	29.6	21.95	24.3
Synthetic	45.01	43.78	42.96	43.69	42.47	-3.68	48.46	32.3	42.09

TABLE I: Cost savings (%) offered by different algorithms under peak-based tariff.

savings are only 7.52%). We report similar observations for $\text{ON}^*_{\text{Drop}}$ with 6.28% cost savings for Google.

Key Insights: (i) For our workloads and parameters, delaying demand offers more benefits than dropping, (ii) workload properties strongly affect the cost savings under peak-based tariff, and (iii) our stochastic control techniques are able to leverage workload prediction to offer near-optimal cost savings when there is no unexpected flash crowd.

VI. Conclusions and Future Work

We proposed a general framework for optimizing data center electric utility bills using the many control knobs that exist for modulating the power consumption of IT equipment. To overcome the difficulty of casting/updating such control problems and the computational intractability they are likely to suffer from in general, our optimization framework was hierarchical and employed both temporal aggregation and control knob abstraction. We developed a suite of algorithms for our upper layer (geared towards dealing with different forms of uncertainty). Finally, we presented results from our empirical evaluation using real-world traces and workloads. Our initial findings were promising: e.g., our approach offered net cost savings of about 25% and 18% to a streaming media server and a MapReduce-based batch workload (both based on real-world traces and utility pricing), respectively.

References

[1] Y. Chen, A. Ganapathi, R. Griffith, and R. H. Katz, "The case for evaluating mapreduce performance using workload suites," in *MASCOTS*, 2011, pp. 390–399.

[2] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, "Managing server energy and operational costs in hosting centers," ser. SIGMETRICS '05, pp. 303–314.

[3] "Fort Collins Coincident Peak," 2013, http://www.fcgov.com/utilities/business/rates/electric/coincident-peak.

[4] "Duke utility bill tariff," 2012, http://www.considerthecarolinas.com/pdfs/scscheduleopt.pdf.

[5] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *ISCA*, 2007, pp. 13–23.

[6] T. C. Ferreto, M. A. S. Netto, R. N. Calheiros, and C. A. F. D. Rose, "Server consolidation with migration control for virtualized data centers," *Future Generation Comp. Syst.*, vol. 27, no. 8, pp. 1027–1034, 2011.

[7] N. M. Freris, C.-H. Hsu, X. Zhu, and J. P. Singh, "Resource allocation for multihomed scalable video streaming to multiple clients," in *Multimedia (ISM)*. IEEE, 2010, pp. 9–16.

[8] S. Govindan, A. Sivasubramaniam, and B. Urgaonkar, "Benefits and limitations of tapping into stored energy for datacenters," in *ISCA*, Jun 2011, pp. 341–352.

[9] S. Govindan, J. Choi, B. Urgaonkar, A. Sivasubramaniam, and A. Baldini, "Statistical profiling-based techniques for effective power provisioning in data centers," ser. EuroSys '09, pp. 317–330.

[10] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke, "Drpm: Dynamic speed control for power management in server class disks," ser. ISCA '03, 2003, pp. 169–181.

[11] A. R. Karlin, M. S. Manasse, L. A. McGeoch, and S. Owicki, "Competitive randomized algorithms for non-uniform problems," ser. SODA '90, 1990, pp. 301–309.

[12] V. Kontorinis, L. E. Zhang, B. Aksanli, J. Sampson, H. Homayoun, E. Pettis, D. M. Tullsen, and T. S. Rosing, "Managing distributed ups energy for effective power capping in data centers," *SIGARCH Comput. Archit. News*, vol. 40, no. 3, pp. 488–499, Jun. 2012.

[13] B. Li, G. Y. Keung, S. Xie, F. Liu, Y. Sun, and H. Yin, "An empirical study of flash crowd dynamics in a p2p-based live video streaming system," in *GLOBECOM*, 2008, pp. 1752–1756.

[14] Z. Liu, A. Wierman, Y. Chen, and B. Razon, "Data center demand response: Avoiding the coincident peak via workload shifting and local generation," in *ACM SIGMETRICS*, Jun 2013.

[15] Z. Liu, M. Lin, A. Wierman, S. H. Low, and L. L. Andrew, "Greening geographical load balancing," ser. SIGMETRICS '11, 2011, pp. 233–244.

[16] "MPEG-4 levels," http://en.wikipedia.org/wiki/MPEG-4.

[17] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, "No "power" struggles: coordinated multi-level power management for the data center," in *ASPLOS*, 2008, pp. 48–59.

[18] L. Rao, X. Liu, L. Xie, and Z. Pang, "Hedging against uncertainty: A tale of internet data center operations under smart grid environment," *Smart Grid, IEEE Transactions on*, vol. 2, no. 3, pp. 555–563, 2011.

[19] "Time-of-use large general service, SCEG," 2013, http://www.sceg.com/en/commercial-and-industrial/rates/electric-rates/default.html.

[20] Y. O. Sharrab and N. J. Sarhan, "Aggregate power consumption modeling of live video streaming systems," ser. MMSys '13, pp. 60–71.

[21] "Puget Energy: Time-of-use electricity billing," 2006, http://energypriorities.com/entries/2006/02/pse_tou_amr_case.php.

[22] R. Urgaonkar, B. Urgaonkar, M. J. Neely, and A. Sivasubramaniam, "Optimal power cost management using stored energy in data centers," in *ACM SIGMETRICS*, Jun 2011, pp. 221–232.

[23] P. Ven, N. Hegde, L. Massoulie, and T. Salonidis, "Optimal control of residential energy storage under price fluctuations," in *International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies*, 2011.

[24] A. Venkatraman, "Global census shows datacentre power demand grew in 2012, Computer Weekly, Oct. 08," 2012, http://www.computerweekly.com/news/2240164589/Datacentre-power-demand-grew-63-in-2012-Global-datacentre-census.

[25] C. Wang, B. Urgaonkar, Q. Wang, G. Kesidis, and A. Sivasubramaniam, "Data center cost optimization via workload modulation under real-world electricity pricing," Penn State University. Available: http://www.cse.psu.edu/research/publications/tech-reports/2013/CSE-13-010.pdf, Tech. Rep., 2013.

[26] D. Wang, C. Ren, A. Sivasubramaniam, B. Urgaonkar, and H. K. Fathy, "Energy storage in datacenters: What, where and how much?" in *ACM SIGMETRICS*, Jun 2012.

[27] H. Xu and B. Li, "Reducing electricity demand charge for data centers with partial execution," *ArXiv e-prints*, Jul. 2013.

[28] Y. Yao, L. Huang, A. Sharma, L. Golubchik, and M. Neely, "Data centers power reduction: A two time scale approach for delay tolerant workloads," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 1431–1439.

[29] Q. Yin, Y. Jiang, S. Jiang, and P. Y. Kong, "Analysis on generalized stochastically bounded bursty traffic for communication networks," in *in Proc. IEEE Local Computer Networks 2002*, 2002, pp. 141–149.