An Emergency Demand Response Mechanism for Cloud Computing

Ruiting Zhou
Dept. of Computer Science
University of Calgary
rzho@ucalgary.ca

Zongpeng Li
Dept. of Computer Science
University of Calgary
zongpeng@ucalgary.ca

Chuan Wu
Dept. of Computer Science
The University of Hong Kong
cwu@cs.hku.hk

ABSTRACT

We study emergency demand response (EDR) mechanisms from data centers' perspective, where a cloud data center participates in a mandatory EDR program while receiving online computing job bids. We target a realistic EDR mechanism where: i) The cloud provider dynamically packs different types of resources on servers into requested VMs and computes job schedules to meet users' requirements; ii) The power consumption of servers in the cloud is limited by the grid through the EDR program; iii) The operating cost of the cloud is considered in the calculation of social welfare, measured by electricity cost. We propose an online auction for dynamic cloud resource provisioning under the EDR program, which runs in polynomial time, achieves truthfulness and close-to-optimal social welfare for the cloud ecosystem.

Keywords

Cloud Computing; Demand Response; Mechanism Design; Approximation Algorithms

1. INTRODUCTION

A key problem in a power network is the realtime balance of supply and demand. Demand response facilitates the efficiency, reliability, and sustainability of modern smart grids by reducing and temporally shifting peak loads [22]. Data centers are ideal candidates for participation in such demand response programs, as they demand a substantial fraction of the total power supply, yet often with an elastic nature [15]. In 2011, data centers consumed approximately 1.5% of electricity worldwide, and the ratio is predicted to increase to 8% by 2020 [8]. Furthermore, computing jobs in data centers are often elastic and hence can be scheduled flexibly across the temporal domain [15], amenable to demand curtailing and temporal shifting.

A representative scenario for data centers to participate in a demand response program is coordinated consumption reduction dictated by the grid in *emergency demand response* (EDR). When stability of the grid is otherwise jeopardized, EDR coordinates the power usage of large electricity users to prevent blackouts. Because of their huge yet flexible demand, data centers now serve as a main force in EDR. For example, on July 22, 2011, hundreds of data centers participated in EDR by shifting their workload to reduce the power consumption, preventing a nation-wide blackout in the USA and Canada [9]. A typical type of EDR program is mandatory EDR [11, 10]. Data centers sign a contract ahead with the smart grid, and commit to reduce load or only consume electricity up to a certain level when an EDR signal is dispatched. They receive monetary remuneration when their actual power consumption is below the commitment level, and face a heavy penalty otherwise.

We focus on EDR in cloud data centers which run jobs from many users. Such a cloud data center faces a highly non-trivial optimization problem in the event of EDR, in which it strives to satisfy the power consumption reduction dictated by the grid, make judicious online decisions on accepting/declining job bids submitted by cloud users, and compute the most efficient execution schedules for the accepted jobs to minimize operating cost. Operating cost of the cloud comprises mainly of electricity cost, which in turn is directly coupled with the processing power of the cloud data center, *i.e.*, how fast the cloud can serve the admitted jobs. A cloud user's job bid specifies (a) the number of each type of Virtual Machines (VMs) required, which can be directly mapped to the amount of each type of cloud resources (*e.g.*, CPU, RAM, disk) required, (b) the length of the job, in number of time slots required for job execution, (c) the preferred deadline for job completion, as well as a penalty function that describes the cost incurred by different degrees of deadline violation, and (d) the amount of monetary remuneration the cloud user is willing to pay.

The goal of this work is to design an online auction for execution at the cloud upon an EDR event, such that (i) the auction runs in an online fashion, making job admission and scheduling decisions immediately upon the arrival of a bid, (ii) the auction mechanism is time efficient and runs in polynomial time, (iii) the auction is truthful, in that it guarantees truthful bidding constitutes a dominant strategy for each cloud user, and (iv) social welfare, including the net utility of both the cloud and its users, is maximized. By definition, the social welfare depends on which cloud jobs are served as well as the cloud's operating cost for serving them, which comprises of primarily electricity cost.

In the design of the online auction, we first propose a new framework, *compact exponential* convex programs, to handle job scheduling constraints in the time domain. We

DOI: 10.1145/1235

then develop a posted pricing auction framework towards the truthful online auction design, which leverages the classic primal-dual technique for approximation algorithm design. We evaluate our online auctions through both theoretical analysis and empirical studies driven by real-world traces.

The rest of the paper is structured as follows. In Sec. 2, we review related work in demand response and cloud market mechanisms. Sec. 3 outlines the problem model and assumptions. Sec. 4 presents and analyzes an online cloud auction under EDR. Sec. 5 presents performance evaluation, and Sec. 6 concludes the paper.

2. RELATED WORK

The earliest cloud auctions do not consider the dynamic provisioning of VMs based on user demand, and assume that there are a signal type of VMs or the number and types of VMs are fixed [14]. Subsequently, auction design for dynamic VM provisioning, in which the cloud provider assembles VMs based on demand expressed in user bids, starts to appear later. Zhang *et al.* [17] formulate the dynamic resource provisioning problem in the cloud into a combinatorial optimization problem with a packing nature, and propose a truthful randomized auction to solve it. Zhang *et al.* [21] employ smoothed analysis and randomized reduction techniques to design a randomized cloud resource auction.

A series of recent studies have focused on the online auction design for dynamic VM allocation in cloud computing. Shi *et al.* present the first online combinatorial auction for cloud computing [12]. Zhang *et al.* [20] consider a more practical scenario where cloud users' bids arrive randomly over a long time span. Compared with existing studies, the cloud users in our EDR auction bid for job execution rather than VM occupation. This work is among the first in cloud-side auction design that considers the scheduling of jobs under EDR power usage reduction requirement, and proposes an online auction framework to properly handle the scheduling challenge.

The majority of the demand response literature studies demand response from the smart grid's perspective [22]. Along the direction of data center demand response, Zhang *et al.* [19] study EDR in multi-tenant colocation data centers. Zhou *et al.* [23] consider the electricity trade between smart grids and green data centers in demand response. This work is from the data center's perspective instead, and focuses on the admission and scheduling of cloud users' jobs to satisfy the power consumption constraint in EDR, while striving to maximize social welfare of the cloud ecosystem.

3. SYSTEM MODEL

For mandatory EDR [11, 10], the data center signs a contract with the smart grid *a priori* (*e.g.,* one year ahead with PJM [11]) and receives monetory rebates for committed load reduction, whereas failure to cut load as required during EDR incurs a heavy penalty. In the event of EDR, the grid sends a signal to the data center at the beginning of the auction, specifying the amount of energy reduction required in each time slot. The data center then calculates the amount of available power in each slot, $E_t, \forall t \in [T]$, to schedule its job execution.

The cloud data center hosts S servers, and offers K types of resources, as exemplified by CPU, RAM and disk storage, which can be dynamically assembled into different types of VMs. Let $[X]$ denote the integer set $\{1, 2, \ldots, X\}$. We assume the amount of type-k resource available in server $s \in [S]$ is c_{ks} units. The cloud service provider acts as the auctioneer to lease VMs to cloud users through an online auction.

There are I cloud users, each submitting one bid for executing its job, during a large time span $1, 2, \ldots, T$. User bids arrive randomly, each requesting a bundle of tailor-made VMs for job execution, mapping to a required amount of each type of resource. Let B_i denote user i's bid submitted at time t_i. It contains: i) r_i^k, the amount of type-k resource required. ii) w_i, the number of time slots (not necessarily consecutive) needed to complete the job by the tailor-made VMs. iii) d_i, the desired deadline for job completion. iv) $g_i(\tau_i)$, a penalty function defined over deadline violation, τ_i:

$$g_i(\tau_i) = \begin{cases} g_{c_i}(\tau_i), & \text{if } \tau_i \in [0, T - d_i] \\ +\infty, & \text{otherwise} \end{cases} \quad (1)$$

where $d_i + \tau_i$ is the job completion time. Let b_i denote user i's bidding price if its job is completed before the deadline d_i, and then $b_i - g_i(\tau_i)$ is the corresponding bidding price with completion time $d_i + \tau_i$. $g_{c_i}(\tau_i)$ is a nondecreasing function with $g_{c_i}(0) = 0$. User i's bidding language can be expressed as follows: $B_i = \{t_i, \{r_i^k\}_{k\in[K]}, w_i, d_i, b_i, g_i(\tau_i)\}$.

Upon the arrival of each bid, the cloud provider immediately computes the resource allocation and announces the auction results: i) $x_{is} \in \{0, 1\}$, where $x_{is} = 1$ if user i's job is accepted and allocated on server s, and 0 otherwise. ii) $y_{is}(t) \in \{0, 1\}$ encodes the scheduling of user i's job, where $y_{is}(t) = 1$ if user i's job is scheduled to run on server s at time t, and 0 otherwise. iii) p_i, user i's payment. Let v_i and $g_i'(\tau_i)$ be user i's true valuation if its job is completed before d_i and true penalty function, and then $v_i - g_i'(\tau_i)$ is the true valuation of user i's bid. User i's utility with bidding price $b_i - g_i(\tau_i)$ is $u_i(b_i - g_i(\tau_i)) = \sum_{s\in[S]} v_i x_{is} - g_i'(\tau_i) - p_i$. Each user is assumed to be selfish and rational, with a natural aim to maximize its own utility. They may choose to lie about the true valuation if doing so leads to a higher utility. In our online auction design, social "happiness" is the target of optimization; towards this goal, it is important to elect truthful bids.

Definition (Truthful Auction): A cloud auction is *truthful* if bidding true valuation is a dominant strategy for a cloud user, always maximizing the user utility: for all $b_i - g_i(\tau_i) \neq v_i - g_i'(\tau_i)$, $u_i(v_i - g_i'(\tau_i)) \geq u_i(b_i - g_i(\tau_i))$.

It is natural to consider the operating cost when we aim to maximize the social welfare. The operating cost mainly comprises of power consumption of the servers, increasing with the increment of the resource occupied on the server. The power consumption of a server can typically be modelled as: $\sum_{k\in[K]} \beta_k u_k$ [13], where u_k is the utilization of type-k resource, and β_k represents the power consumption when type-k resource is in full usage. The operating cost equals the electricity charge paid by the data center to the utility company.

Let $e(t)$ be the actual power consumption in slot t, the operating cost of the data center at t can be defined as:

$$f_t(e(t)) = \begin{cases} h_t e(t), & \text{if } e(t) \leq E_t \\ +\infty, & \text{otherwise} \end{cases} \quad (2)$$

where h_t is electricity price at time t, known by the data center before the auction starts.

Definition (Social Welfare): The social welfare is the aggregate of users' utilities $\sum_{i\in[I]}(\sum_{s\in[S]} v_i x_{is} - g_i'(\tau_i) - p_i)$ plus

cloud provider's utility $\sum_{i\in[I]} p_i - \sum_{t\in[T]} f_t(e(t))$. Since payments cancel themselves, the social welfare becomes $\sum_{i\in[I]} (\sum_{s\in[S]} v_i x_{is} - g'_i(\tau_i)) - \sum_{t\in[T]} f_t(e(t))$.

4. ONLINE AUCTION DESIGN

4.1 Social Welfare Maximization Problem

Under the assumption of truthful bidding, the social welfare maximization problem can be formulated into the following convex program:

$$\text{maximize} \quad \sum_{i\in[I]} (\sum_{s\in[S]} b_i x_{is} - g_i(\tau_i)) - \sum_{t\in[T]} f_t(e(t)) \tag{3}$$

subject to:

$$\sum_{s\in[S]} x_{is} \le 1, \forall i \in [I], \tag{3a}$$

$$y_{is}(t)t \le d_i + \tau_i, \forall t \in [T], \forall s \in [S], \forall i \in [I] : t_i \le t, \tag{3b}$$

$$w_i x_{is} \le \sum_{t\in[T]:t_i\le t} y_{is}(t), \forall i \in [I], \forall s \in [S], \tag{3c}$$

$$\sum_{i\in[I]:t_i\le t} r_i^k y_{is}(t) \le c_{ks}, \forall k \in [K[, \forall s \in [S], \forall t \in [T], \tag{3d}$$

$$\sum_{s\in[S]} \sum_{k\in[K]} \beta_{ks} \left(\frac{\sum_{i\in[I]:t_i\le t} r_i^k y_{is}(t)}{c_{ks}}\right) \le e(t), \forall t \in [T], \tag{3e}$$

$$\tau_i, e(t) \ge 0, x_{is}, y_{is}(t) \in \{0,1\}, \quad \forall i \in [I], \forall s \in [S], \forall t \in [T]. \tag{3f}$$

Note that the following constraint is redundant, and is not explicitly included in the above convex problem: $y_{is}(t) \le x_{is}, \forall i \in [I], \forall s \in [S], \forall t \in [T]$. Constraint (3a) indicates that each user's job is executed on at most one server. Constraint (3b) ensures that a job is scheduled to run only after its arrival time. Constraint (3c) guarantees that the number of time slots allocated to an accepted bid is sufficient for completing the job. The capacity limit of each type of resource is modelled in constraint (3d), and constraint (3e) records the total power consumption in each time slot into $e(t)$. Setting $f_t(e(t)) = +\infty$ when $e(t) > E_t$ ensures that the actual power consumption is capped at the EDR-specified amount.

If we let $g_i(\tau_i) = f_t(e(t)) = 0$, even in the offline setting, problem (3) without constraints (3b), (3b), (3c) and (3e) is still an NP-hard combinatorial optimization problem, equivalent to the classic knapsack problem. The challenge further escalates when we involve the jobs' soft deadlines and operating cost. We shall resort to the primal-dual algorithm design technique to address some of these challenges. However, the technique cannot be directly applied to (3) since it involves unconventional constraints for modelling job deadlines. We first propose a new framework to handle these unconventional constraints. More specifically, we reformulate the original problem (3) into a *compact exponential* convex problem with a **compact** packing structure, at the price of involving an **exponential** number of decision variables:

$$\text{maximize} \quad \sum_{i\in[I]} \sum_{l\in\zeta_i} b_{il} x_{il} - \sum_{t\in[T]} f_t(e(t)) \tag{4}$$

subject to:

$$\sum_{l\in\zeta_i} x_{il} \le 1, \forall i \in [I], \tag{4a}$$

$$\sum_{i\in[I]} \sum_{l:t\in T(l), s\in S(l)} r_i^k x_{il} \le c_{ks}, \forall k \in [K], \forall s \in [S], \forall t \in [T], \tag{4b}$$

$$\sum_{s\in[S]} \sum_{k\in[K]} \beta_{ks} \left(\frac{\sum_{i\in[I]} \sum_{l:t\in T(l), s\in S(l)} r_i^k x_{il}}{c_{ks}}\right) \le e(t), \forall t \in [T], \tag{4c}$$

$$e(t) \le 0, x_{il} \in \{0,1\}, \forall t \in [T], \forall i \in [I], \forall l \in \zeta_i. \tag{4d}$$

Here ζ_i is the set of feasible time schedules for user i's job. A feasible time schedule is the vector $l = (\{x_{is}\}_{\forall s\in[S]}, \{y_{is}(t)\}_{\forall s\in[S], t\in[T]}, \tau_i)$ that satisfies constraints (3a), (3b) and (3c). x_{il} is the binary decision variable where $x_{il} = 1$ if user i's job is accepted and executed according to schedule $l \in \zeta_i$, and 0 otherwise. b_{il} is the value based on schedule l, which equals $b_i - g_i(\tau_i)$ where τ_i is the duration of deadline violation according to l. $T(l)$ and $S(l)$ represent the set of time slots and the server when and where user i's job is executed in schedule l, respectively. Constraints (4b) and (4c) are equivalent to (3d) and (3e). Constraint (4a) guarantees that a job can only be accepted according to one schedule. A feasible solution to (3) corresponds to a feasible solution in (4) and vice versa, and hence the optimal objective values of both problems are equal.

We relax $x_{il} \in \{0,1\}$ to $x_{il} \ge 0$ and introduce dual variables u_i,$p_{ks}(t)$ and $m(t)$ to (4a), (4b) and (4c). The Fenchel dual [4, 5] of the relaxed problem (4) is:

$$\text{minimize} \quad \sum_{i\in[i]} u_i + \sum_{k\in[K]} \sum_{s\in[S]} \sum_{t\in[T]} c_{ks} p_{ks}(t) + \sum_{t\in[T]} f_t^*(m(t)) \tag{5}$$

s.t:

$$u_i \ge b_{il} - \sum_{k\in[K]} \sum_{t\in T(l)} \sum_{s\in S(l)} r_i^k (p_{ks}(t) + m(t)\frac{\beta_{ks}}{c_{ks}}), \quad \forall i \in [I], \forall l \in \zeta_i, \tag{5a}$$

$$p_{ks}(t), u_i, m(t) \ge 0, \forall i \in [I], \forall k \in [K], \forall s \in [S], \forall t \in [T]. \tag{5b}$$

Where $f_t^*(m(t))$ is the convex conjugate [16] of the cost function $f_t(\cdot)$, defined as:

$$f_t^*(m(t)) = \sup_{e(t)\ge 0} \{m(t)e(t) - f_t(e(t))\}.$$

PROPOSITION 1. *The explicit expression of $f_t^*(m(t))$ is:*

$$f_t^*(m(t)) = \begin{cases} 0, & m(t) \le h_t \\ (m(t) - h_t)E_t, & m(t) > h_t \end{cases} \tag{6}$$

All the missing proofs can be found in our technical report [2].

4.2 Online Auction Design

A key problem in the online auction design is to decide whether to accept user i's job and how to schedule its job to maximize its utility, while the power consumption in each slot is limited by the EDR program. If the cloud provider accepts user i's job on server s with schedule l, then $x_{is} = 1$, τ_i is assigned according to the completion time, $y_{is}(t)$ is updated and $e(t)$ is increased for slots in $T(l)$. To solve the original convex problem (3), we seek the help of the compact exponential convex problem (4) and its dual (5). We observe that for each primal variable x_{il}, there is a dual constraint (5a) associated to it. Complementary slackness in the primal-dual technique indicates that x_{il} is updated based on its dual constraint. x_{il} remains zero unless its associated dual constraint becomes tight. Because dual variable u_i is nonnegative, we let u_i be the maximum of 0 and the right hand side (RHS) of constraint (5a), that is,

$$u_i = \max(0, \max_{l\in\zeta_i} \{b_{il} - \sum_{k\in[K]} \sum_{t\in T(l)} \sum_{s\in S(l)} r_i^k (p_{ks}(t) + m(t)\frac{\beta_{ks}}{c_{ks}})\}). \tag{7}$$

Accordingly, the winner is determined based on u_i: the cloud provider accepts user i's job if $u_i > 0$, and serves it according to the schedule that maximizes the RHS of (5a). The cloud provider rejects user i's job if $u_i \leq 0$.

The rational can be explain as follows. If we interpret $p_{ks}(t)$ as the unit capital price of type-k resource on server s at time t and $m(t)$ as the unit electricity price at time t, then $\sum_{k\in[K]}\sum_{t\in T(l)}\sum_{s\in S(l)} r_i^k(p_{ks}(t)+m(t)\frac{\beta_{ks}}{c_{ks}})$ is the total cost of user i's job if it is accepted and scheduled by l. Furthermore, the RHS of (5a) is user i's utility with schedule l. If we interpret u_i as user i's utility, the assignment of u_i in (7) guarantees that user i's job is always served with the schedule that effectively maximizes its utility based on the current price, which leads to social welfare maximization and truthfulness.

Algorithm 1 A Primal-dual Online Auction A_{online}

Input: bidding language $\{B_i\},\{c_{ks}\},\{\beta_k\},\{E_t\},\{h_t\}$

1: Define cost function $f_t(e(t))$ according to (2);
2: Define function $p_{ks}(z_{ks}(t))$ according to (8);
3: Initialize $x_{is}=0, y_{is}(t)=0, z_{ks}(t)=0, \tau_i=0, x_{il}=0, u_i=0, p_{ks}(t)=0, m(t)=h_t, e(t)=0, \forall i\in[I], l\in\zeta_i, k\in[K], s\in[S], t\in[T]$;
4: **Upon the arrival of the ith user**
5: $(x_{is},\{y_{is}(t)\},p_i,\{p_{ks}(t)\},\{z_{ks}(t)\},\{e(t)\}) = A_{core}(B_i,\{c_{ks}\},\{E_t\},\{p_{ks}(t)\},\{z_{ks}(t)\},\{m(t)\},\{e(t)\})$;
6: **if** $\exists s\in[S], x_{is}=1$ **then**
7: Accept user i's bid and allocate resources to server s according to $y_{is}(t)$; Charge p_i from user i;
8: **else**
9: Reject user i.
10: **end if**

Although there are an exponential number of dual constraints involved in the computation of u_i, most of them can be filtered by a dual oracle based on dynamic programming. This is realized through the selection of schedules. We fix a polynomial number of schedules by the dual oracle (lines 1-15 in Alg. 2), and let u_i be the maximum of zero and the RHS of (5a) with these schedules. For each server $s\in[S]$, we construct a set of best schedules. We fix the job completion time to be t_c ($t_c\in[t_i+w_i-1,T$), then the best schedule is the one with the minimum price among all schedules of the same completion time. The output of the dual oracle is such S sets of best schedules. The construction of the best schedules can be accomplished through dynamic programming method. The base case is the schedule l_0 with slots in $[t_i,t_i+w_i-1]$. We push the completion time one slot forward each time. We calculate the price $c(t)$ for user i's job running at time t, *i.e.,* $c(t)=\sum_{k\in[K]} r_i^k(p_{ks}(t)+m(t)\frac{\beta_{ks}}{c_{ks}})$. If the completion time passes the deadline d_i, the price will be increased by the corresponding penalty, *i.e.,* $c(t)=\sum_{k\in[K]} r_i^k(p_{ks}(t)+m(t)\frac{\beta_{ks}}{c_{ks}})+g_i(t-d_i)$. In the process of replacing the completion time, we only need to compare the price of the old competition time and w_i-1 slots preceding the old competition time.

We next discuss the design of the two prices: unit capital price $p_{ks}(t)$ and unit electricity price $m(t)$. Recall that h_t is the unit electricity price at time t charged by the power grid; thus we let $m(t)=h_t$ based on the interpretation of dual variable $m(t)$. For the design of $p_{ks}(t)$, we introduce a new variable $z_{ks}(t)$, representing the amount of allocated type-k resource on server s at time t. Let U_k and L_k be the maximum and minimum values per unit of type-k resource per unit of time, respectively. U_k and L_k represent how users evaluate a unit of type-k resource, considering both the capital cost and electricity cost. Hence, we assume that $L_k > h_t, \forall t\in[T]$, without loss of generality. We propose a price function such that the total unit price $p_{ks}(t)+m(t)$ equals L_k at the beginning and reaches U_k ultimately. Because $m(t)=h(t)$, we let $p_{ks}(t)$ start at L_k-h_t and exponentially increase with the growth of the current usage $z_{ks}(t)$. $p_{ks}(t)$ equals U_k-h_t when $z_{ks}(t)$ exceeds its capacity c_{ks}. In this case, the cloud provider will not accept any more jobs. To sum up, $p_{ks}(t)$ and $m(t)$ are defined as follows:

$$p_{ks}(t)(z_{ks}(t)) = (L_k-h_t)\left(\frac{U_k-h_t}{L_k-h_t}\right)^{\frac{z_{ks}(t)}{c_{ks}}} \tag{8}$$

$$m(t)=h_t, \forall t\in[T] \tag{9}$$

where $U_k \leq \max_{i\in[I]}\frac{b_i}{w_i r_i^k}$ and $L_k \geq \min_{i\in[I]}\frac{b_i-g_i(T-d_i)}{\sum_{k\in[K]} w_i r_i^k}$ with truthful bidding.

Algorithm 2 A Scheduling Algorithm A_{core}.

Input: $B_i,\{c_{ks}\},\{E_t\},\{p_{ks}(t)\},\{z_{ks}(t)\},\{m(t)\},\{e(t)\}$
Output: $x_{is},\{y_{is}(t)\},p_i,\{p_{ks}(t)\},\{z_{ks}(t)\},\{e(t)\}$

1: **for all** $s\in[S]$ **do**
2: Add slot $t\in[t_i,T]$ to set $\mathcal{T}$ if $z_{ks}(t)+r_i^k\leq c_{ks}, \forall k\in[K]$ and $\sum_{k\in[K]}\beta_{ks}r_i^k/c_{ks}+e(t)\leq E_t$;
3: Let schedule l_0 include the first w_i slots $(t_1,t_2,\ldots,t_{w_i})$ in $\mathcal{T}$; Define $j=1$;
4: **while** $w_i+j\leq|\mathcal{T}|$ **do**
5: $l_j=l_{j-1}$;
6: Let t_c is the (w_i+j)th slot in $\mathcal{T}$;
7: $c(t) = \sum_{k\in[K]} r_i^k(p_{ks}(t)+m(t)\frac{\beta_{ks}}{c_{ks}}), \forall t\in\{t_1,t_2,\ldots,t_{w_i},t_c\}$;
8: If $t_c>d_i$, $c(t_c)=c(t_c)+g_i(t_c-d_i)$;
9: $t_m=\arg\max_{t\in\{t_1,\ldots,t_{w_i-1}\}} c(t)$;
10: If $c(t_{w_i})<c(t_m)$, for schedule l_j, replace the slot t_m with t_{w_i}; Save t_c into t_{w_i};
11: $\mathcal{P}_j=\sum_{t\in T(l_j)} c(t)$; $j=j+1$;
12: **end while**
13: $s^*=\arg\min_j\{\mathcal{P}_j\}; \mathcal{P}_s^*=\mathcal{P}_{s^*}; l_s^*=l_{s^*}$;
14: **end for**
15: $\hat{s}=\arg\min_s\{\mathcal{P}_s^*\}; \hat{\mathcal{P}}=\mathcal{P}_{\hat{s}}^*, \hat{l}=l_{\hat{s}}^*$;
16: **if** $b_i-\hat{\mathcal{P}}>0$ **then**
17: $x_{i\hat{s}}=1; y_{i\hat{s}}(t)=1, \forall t\in T(\hat{l}); x_{i\hat{l}}=1$;
18: $u_i=b_i-\hat{\mathcal{P}}; p_i=\sum_{k\in[K]}\sum_{t\in T(\hat{l})} r_i^k(p_{k\hat{s}}(t)+m(t)\frac{\beta_{k\hat{s}}}{c_{k\hat{s}}})$;
19: $z_{k\hat{s}}(t)=z_{k\hat{s}}(t)+r_i^k, \forall k\in[K], t\in T(\hat{l}); e(t)=e(t)+\sum_{k\in[K]} r_i^k\beta_{k\hat{s}}/c_{k\hat{s}}, \forall t\in T(\hat{l})$;
20: $p_{k\hat{s}}(t)=p_{k\hat{s}}(z_{k\hat{s}}(t)), \forall k\in[K], t\in T(\hat{l})$;
21: **end if**
22: **Return** $x_{is},\{y_{is}(t)\},p_i,\{p_{ks}(t)\},\{z_{ks}(t)\},\{e(t)\}$

The online auction A_{online} is shown in Alg. 1 with scheduling algorithm A_{core} in Alg. 2 running for each user. A_{online} first defines the cost function and price function in lines 1-2. Line 3 initializes all primal and dual variables. Upon the arrival of the ith user, the scheduling algorithm A_{core} selects the best schedule $\hat{l}$ that maximizes user i's utility through the dual oracle (lines 1-15). If user i can obtain positive utility, primal variables x_{is}, y_{is} and x_{il} are updated accordingly (line 17). Then line 18 calculates the utility and the payment. Line 19 increases the usage of K resources on the specified server and records the current power consumption level. Finally, unit resource price is updated in line 20.

4.3 Theoretical Analysis

THEOREM 1. *A_{online} terminates in polynomial time, and returns a feasible solution for problem (3), (4) and (5).*

THEOREM 2. *The online auction A_{online} is a truthful auction.*

We proceed to analyze the competitiveness of A_{online} in social welfare, measured by the competitive ratio. The *competitive ratio* is the upper-bound ratio of the social welfare achieved by the optimal solution of convex problem (3) to the social welfare achieved by A_{online}.

THEOREM 3. *The online auction A_{online} in Alg. 1 is α-competitive in social welfare with $\alpha = \max_{k\in[K]}\{\ln \frac{U_k - h_{max}}{L_k - h_{max}}\}$, where $h_{max} = \max_{t\in[T]} h_t$.*

5. PERFORMANCE EVALUATION

Simulation setup. We evaluate the performance of our online auction through large-scale simulation studies based on Google Cluster Data [1], which contains information about jobs running on Google compute cells, including start time, execution duration and normalized job demand (CPU and RAM). We translate each job into a bid, requesting two types of resources at demands extracted from the traces. We assume each time slot is 5 minutes long [1] and each job consumes $[1, 12]$ slots, arriving sequentially in 18 hours with $T = 220$. Each job's deadline is randomly generated between its arrival time and the system end time. We set the bidding price of each job by multiplying the overall resource demands by unit prices randomly picked within the range $[L_k, U_k]$. The default values are $L_k = 5$ and $U_k = 50$ for A_{online}. The capacity of type-k resource in server s, c_{ks}, is set to 1 as the resource demand is in normalized units.

For the power consumption of a server, parameter β_{ks} is set within $[20, 60]$ for CPU and within $[0.2, 2]$ or RAM [13, 7]. We assume that the data center is powered by BC Hydro with an electricity charge of ¢4.86 per kWh [18, 6]. The value of h_t is generated by adding randomness to it. The available power at each time slot E_t is set to within the range of $[20, 100]$ kW based on a report of data center server power usage and required demand response power reduction [3].

Performance of A_{online}. The optimal social welfare of the convex problem (3) is computed by `CVX` with the Gurobi Optimizer. Fig. 1 shows the ratio of the optimal social welfare over the social welfare achieved by A_{online} under different numbers of users and servers. We observe that A_{online} always performs well with a lower ratio (< 1.5), which is noticeably better than the theoretically proven competitive ratio. The ratio decreases as the number of servers increases, and fluctuates when the number of users grows. A_{online} allocates a user's job on the cheapest server to maximize its utility. Therefore, the algorithm has a larger solution space to explore when the number of servers is large, leading to a better ratio. The number of users does not influence the ratio, as confirmed by the analysis in Theorem 3. Recall that U_k and L_k are the maximum and minimum unit price of type-k resource respectively, defined in the price function (8). Fig. 2 illustrates that A_{online} still achieves a good ratio when we vary the value of U_k/L_k and use the estimated values of U_k as the input of A_{online}. We notice that the ratio becomes larger with the increment of U_k/L_k, while both underestimation and overestimation have minor impact on the performance, as compared to that achieved by the real U_k (labelled by 100%). Theorem 3 reveals that U_k/L_k determines the ratio, which is consistent with the downward trend in Fig. 2. Moreover, underestimation is slightly better than overestimation, due to the reason that overestimation leads to a higher price, filtering out jobs that are supposed to be accepted.

We next investigate the social welfare and the cloud service provider's revenue achieved by A_{online}. Fig. 3 compares the social welfare achieved by A_{online} to the optimal social welfare under different lengths of job execution time (w_i). A_{online} always achieves close-to-optimal performance regardless of the value of w_i. When $w_i \leq 18$, the social welfare increases when the user requests more slots for its job, which is reasonable because the social welfare is mostly contributed by the bidding price, and the user will raise its bidding price when its job needs a long execution time. The gap between the social welfare returned by A_{online} and the optimal social welfare becomes larger with the increment of w_i, as long execution time brings more computation difficulties for A_{online} to approach the optimal solution. Another interesting observation is that the social welfare drops sharply when $w_i > 20$. This is because the competition for resources in each time slot is fiercer with a larger w_i, and then the number of winners would decrease when the number of users is fixed, leading to a smaller overall social welfare.

The 3D figure in Fig. 4 shows that a large social welfare comes with a large number of users and a high value of U_k/L_k. The underlying reason is A_{online} can select more high value bids when there is a large set of users participating in the auction. Furthermore, the bidding price rises when the value of U_k/L_k increases, and hence a higher social welfare is achieved by high value bids. In Fig. 5, we plot the revenue of the cloud service provider under different numbers of users and servers. The change of the number of servers does not have major influence on the revenue. The cloud service provider is able to gain higher profit with a larger set of users, as more jobs with high bidding prices would be accepted to contribute to the revenue.

The performance of A_{online} in terms of winner satisfaction, as measured by the percentage of winning users, is demonstrated in Fig. 6. We observe that a larger percentage of jobs are successfully allocated when the number of participating users is small and U_k/L_k is large. The reason can be explained as follows: The number of winners is almost fixed and limited by the resource capacity and the available power. The users face stiff competition when a large number of users submits bids to the cloud. Furthermore, the winner is determined by the current price of the resource which rises from L_k to U_k. When L_k is close to U_k, the difference between the bidding prices is small. As a result, more bids with similar bidding prices are rejected as the price is increased during each round.

6. CONCLUSION

We studied data center EDR where (i) the power grid dictates an upper-bound in power consumption in each time slot during the EDR period, and (ii) cloud jobs with soft deadlines arrive in an online fashion. We adapt the classic primal-dual framework for efficient approximation algorithm design, and employ a posted-pricing framework for truthful online mechanism design, to derive a truthful online auc-

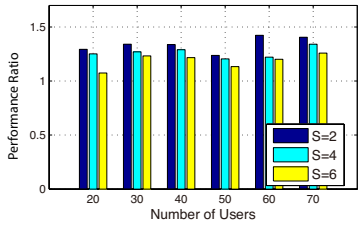

Figure 1: Performance of A_{online} under different numbers of users and servers.

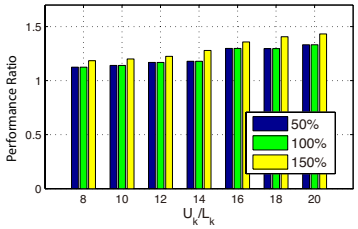

Figure 2: Performance of A_{online} under different U_k/L_k and estimated U_k.

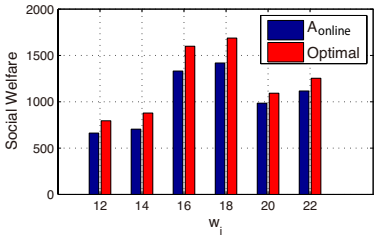

Figure 3: Social welfare of A_{online} under different values of w_i.

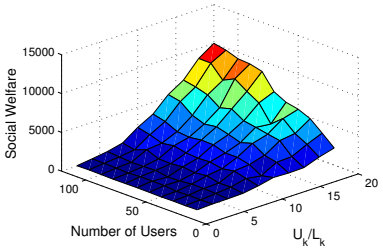

Figure 4: Social welfare of A_{online} under different numbers of users and U_k/L_k.

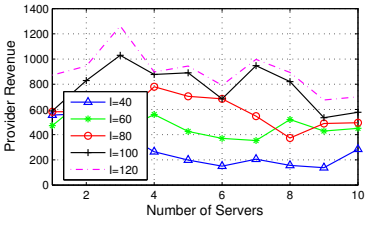

Figure 5: Cloud service provider's revenue of A_{online} with different numbers of servers and users.

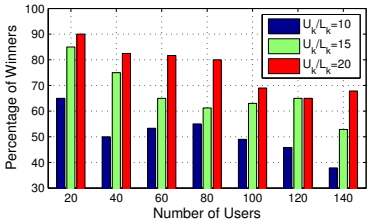

Figure 6: Percentage of winners of A_{online} with different numbers of users and U_k/L_k.

tion that runs efficiently and approaches optimal social welfare. We describe a compact exponential optimization technique, which works in concert with a dual oracle to handle the job completion time constraints imposed by their soft deadlines. Our method may shed light on other mechanism design problems where the optimization problem contains non-conventional constraints, such as delay-constrained optimization in cyber physical systems.

7. REFERENCES

[1] *Google Cluster Data*. https://goo.gl/kNfqAQ.

[2] *Technical report*. https://goo.gl/YYucol.

[3] *Toolkit: Calculate datacenter server power usage*. http://goo.gl/iB9hZv.

[4] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[5] N. R. Devanur. Fisher markets and convex programs. *JACM*, 2010.

[6] B. Hydro. *Power smart*. https://goo.gl/sKLhGG.

[7] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya. Virtual machine power metering and provisioning. In *Proc. of ACM SoCC*, 2010.

[8] Z. Liu, I. Liu, S. Low, and A. Wierman. Pricing data center demand response. In *Proc. ACM SIGMETRICS*, 2014.

[9] A. Misra. *Responding Before Electric Emergencies*. http://goo.gl/eNyquJ.

[10] PJM. *Retail Electricity Consumer Opportunities for Demand Response in PJM's Wholesale Markets*. https://goo.gl/q1lebf.

[11] PJM. *Emergency Demand Response Performance Report 2013/2014*. April 2014.

[12] W. Shi, L. Zhang, C. Wu, Z. Li, and F. Lau. An online auction framework for dynamic resource provisioning in cloud computing. In *Proc. of ACM SIGMETRICS*, 2014.

[13] W. Tian and Y. Zhao. *Optimized Cloud Resource Management and Scheduling: Theories and Practices*. Elsevier Science, 2014.

[14] Q. Wang, K. Ren, and X. Meng. When cloud meets ebay: Towards effective pricing for cloud computing. In *Proc. of IEEE INFOCOM*, 2012.

[15] A. Wierman, Z. Liu, I. Liu, and H. Mohsenian-Rad. Opportunities and challenges for data center demand response. In *Proc. of IEEE IGCC*, 2014.

[16] Wikipedia. *Convex conjugate*. http://en.wikipedia.org/wiki/Convex_conjugate.

[17] L. Zhang, Z. Li, and C. Wu. Dynamic resource provisioning in cloud computing: A randomized auction approach. In *Proc. of IEEE INFOCOM*, 2014.

[18] L. Zhang, Z. Li, C. Wu, and S. Ren. Online electricity cost saving algorithms for co-location data centers. In *Proc. of ACM SIGMETRICS*, 2015.

[19] L. Zhang, S. Ren, C. Wu, and Z. Li. A truthful incentive mechanism for emergency demand response in colocation data centers. In *Proc. of IEEE INFOCOM*, 2015.

[20] X. Zhang, Z. Huang, C. Wu, Z. Li, and F. Lau. Online auctions in IaaS clouds: welfare and profit maximization with server costs. In *Proc. of ACM SIGMETRICS*, 2015.

[21] X. Zhang, C. Wu, Z. Li, and F. Lau. A truthful (1-ε)-optimal mechanism for on-demand cloud resource provisioning. In *Proc. of IEEE INFOCOM*, 2015.

[22] R. Zhou, Z. Li, C. Wu, and M. Chen. Demand response in smart grids: A randomized auction approach. *IEEE Journal on Selected Areas in Communications*, 33(12):2540–2553, 2015.

[23] Z. Zhou, F. Liu, and Z. Li. Pricing bilateral electricity trade between smart grids and hybrid green datacenters. In *Proc. ACM SIGMETRICS*, 2015.