Online Demand Response of GPU Cloud Computing with DVFS

Yu He, Lin Ma, Chuanhe Huang
School of Computer Science, Wuhan University, huangch@whu.edu.cn

***Abstract*—GPU cloud computing is emerging as a new type of cloud service that drives computation-extensive jobs, such as big data analytics and distributed machine learning. The introduction of GPU brings parallel processing power at the cost of excessive energy consumption. Dynamic Voltage and Frequency Scaling (DVFS) is a promising method to control energy consumption of GPU VMs. This work focuses on using DVFS to reduce energy of cloud computing in datacenter demand response. We first consider an online demand response scenario where users arrive stochastically, aiming at maximizing social welfare and meeting energy reduction goals by employing DVFS. We address the challenge posed by DVFS through a new technique of compact infinite optimization. A more practical scenario where both energy and resource limitations present is further studied. We design a primal-dual approximation algorithm that can compute a feasible solution in polynomial time with guaranteed approximation ratio, and a payment scheme that works in concert to form a truthful cloud job auction.**

I. Introduction

Cloud computing provides cloud users with on-demand access to computing resources and services in a *pay-as-you-go* fashion. Given the agility, flexibility, and decreasing cost of cloud services, a growing pool of users and applications are migrating to the cloud, further enhancing its *economies of scale*. Traditional cloud computing are often based on virtual machines with CPU as the only computing resource. In the recent years, the Graphic Processing Unit (GPU) has proliferated for supporting big data analytics such as deep neural network training and multimedia processing, based on its massive parallel processing power. Mainstream cloud providers, such as Amazon and Microsoft, have started to combine the power of cloud computing and GPU, by offering GPU-accelerated cloud computing services.

GPU cloud computing represents a niche market for scientific computing, computational finance, data mining and machine learning. Using IT virtualization technologies, traditional cloud computing packs resources that include CPU, RAM, disk storage and network bandwidth into VMs for lease. GPU computing further packs GPUs into the VMs. Latest examples of GPU VMs offered by Amazon in its EC2 platform is shown in Tab. I. It is notable that unlike CPUs, GPUs are often not virtualized prior to inclusion in the VMs.

Consolidating computing power into central sites, cloud computing comes with enormous electric power consumption at datacenters. Greenpeace estimates that, if the global cloud computing industry was a country, its energy consumption would rank sixth in the world, between Germany and Russia. Furthermore, electricity demand for cloud computing grows non-proportionally fast, as compared to other industries, and is projected to increase by another 60% by 2020 [1]. Energy cost now represents $\sim$ 15% of a cloud provider's total cost of operation [2]. The introduction of GPU VMs exacerbates the situation, since GPUs are known to draw a high amount of power to sustain parallel processing at large-scale, with numerous, densely packed processing units. There is a common initiative in industry as well as academia to develop technologies that will help to create green datacenters and optimize energy consumption [3].

TABLE I: AMAZON EC2 P2 Instance Types

VM Instance	GPU	vCPU	Memory	Cores	GPU Memory
p2.xlarge	1	4	61 GiB	2496	12 GiB
p2.8xlarge	8	32	488 GiB	19968	96 GiB
p2.16xlarge	16	64	732 GiB	39936	192 GiB

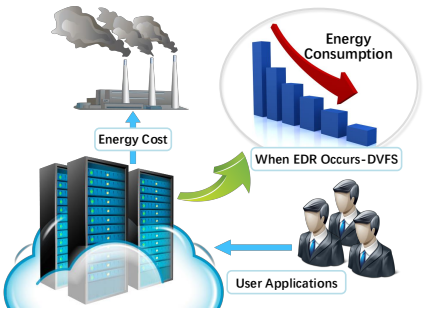

Fig. 1: An Energy-Aware Data Center.

Emergency Demand Response (EDR) guarantees the reliability and sustainability of grids by reducing energy consumption and temporally depressing peak loads [4]. Cloud datacenters operate over massive but often elastic workloads, and thus represent ideal participants of EDR. An important technique to manage their power consumption profile over the temporal domain is Dynamic Voltage and Frequency Scaling (DVFS), through voltage and frequency tuning during task processing. Since power consumption is super-linear to processing speed, DVFS enables interesting trade-off opportunities between energy saving and performance. Off-the-shelf GPU boards have two sets of adjustable voltage/frequency: the core voltage/frequency, and the memory voltage/frequency. In GPU cloud computing, each GPU VM is accelerated by one or more GPUs, and thus we could separately adjust voltage/frequency

for each GPU VM. Fig. 1 illustrates an energy-aware data center. When the datacenter receives a EDR signal, it admits user jobs whose value exceeds projected computing cost, and will use DVFS for efficient scheduling of accepted jobs.

This work considers GPU DVFS as an effective approach to EDR in GPU cloud computing. We leverage an auction mechanism to effectively motivate users to participate in energy conservation. Users submit their task valuation and requirement, as well as delay tolerance upon EDR events. The cloud provider employs DVFS to schedule admitted tasks, for striking a judicious balance between energy consumption (so that energy reduction goals in EDR are met) and execution time (so that user delay tolerance is respected), while deciding the rewards and the charges of users. We target a number of properties in the auction: (1) truthfulness, *i.e.*, cloud users who are selfish but rational will bid true information to maximize their own utilities, (2) time efficiency, *i.e.*, the auction algorithms run in polynomial time, (3) social welfare maximization, *i.e.*m, the total 'happiness' of both the cloud and its users is (approximately) maximized.

Along the design of EDR mechanisms for GPU Cloud Computing with DVFS, we consider two practical scenarios. The first is an online setting, in which we focus on energy constraint in the GPU-accelerated cloud upon EDR events, making task admission and scheduling decisions immediately upon the arrival of each bid. Capacity of computing resources is ignored in the first model. The second is a round-by-round auction setting, in which both energy and resources constraints are taken into account in the EDR mechanism. All bids are submitted at the beginning of each round, thus the auction have access to full knowledge for decision making.

Our contributions in tackling the above questions are summarized as follows: (i) We propose an auction framework that is expressive enough to allow cloud users to specify their workloads, deadline demands and delay tolerance, and we design an efficient mechanism for task scheduling and EDR energy reduction. (ii) We formulate the social welfare maximization problem into an integer linear program (ILP). For the online version of the auction, the natural ILP formulation of the social welfare maximization problem involves nonlinear constraints due to GPU DVFS, and can not be solved using standard algorithmic techniques the classic primal-dual framework. We propose a *compact infinite* optimization framework to efficiently solve this ILP. We convert the natural ILP into an equivalent one, which has a *compact formulation* with traditional packing type of constraints only, at the price of involving an *infinite number of variables* (that correspond to power profiles). We then formulate the LP dual of the compact infinite ILP, and iteratively update primal and dual solutions based on interpretation of the dual variables. Finally, we prove that the competitive ratio of our online algorithm is bounded by 2.68. (iii) In the offline round-by-round setting, we propose an approximation algorithm to handle both energy and resource constraints and obtain a feasible solution in polynomial time, with a guaranteed approximation ratio. Moreover, we design a corresponding payment scheme to guarantee truthfulness of the auction mechanism. (iv) We conduct extensive simulation studies to evaluate the efficiency and effectiveness of the proposed online and offline mechanisms.

In the rest of the paper, we discuss related literature in Sec. II, and the main system model in Sec. III. The online and offline EDR mechanisms are presented and analyzed in Sec. IV and Sec. V. Simulation results are presented in Sec. VI. Sec. VII concludes the paper.

II. Related Work

Early literature on cloud computing and datacenter demand response customarily assume CPU-based VMs. Zhang *et al.* [5] propose an online auction in IaaS clouds to dynamically allocate CPU-based VMs. Sonkar *et al.* [6] study a model of VM scheduling and resource allocation in CPU-only cloud computing. Fayyaz *et al.* [7] design an energy-saving mechanism to distribute VMs through appropriate task scheduling. The VMs considered in the above literature are all CPU-based. With the development of GPU VMs in cloud datacenters, Amazon pioneered in offering GPU-accelerated VMs, which may enable better performance than CPU-only VMs, for big data applications that benefit from parallel processing. This work studies the DVFS management of GPU-accelerated VMs, which are more energy-extensive.

A series of datacenter EDR mechanisms are recently designed for cloud computing. Zhou *et al.* [8] study EDR mechanisms in cloud datacenters with online computing job bids. Zhang *et al.* [9] study EDR in multi-tenant colocation datacenters. Zhou *et al.* [4] consider electricity trade between smart grids and green data centers in demand response. Observing that the energy consumption of GPU far exceeds that of CPU, we study EDR of GPU VMs, and leverage an auction mechanism to encourage users to save energy.

DVFS itself has been a subject of study, first for CPU and then more recently for GPU. Hsieh *et al.* [10] propose a memory-aware cooperative CPU-GPU DVFS governor that considers both memory access footprint as well as CPU/GPU frequency to improve energy efficiency of high-end mobile game workloads. Mei *et al.* [11] study the impact of GPU DVFS on computing performance and power consumption, and summarize efficient performance and power models. Neither of the above studies focuses on DVFS of GPU VMs or in the context of an EDR auction. Our work combines GPU DVFS with a cloud auction mechanism for the first time in the literature. While stimulating users to save energy, the cloud platform dynamically adjusts energy consumption through GPU DVFS.

Compact exponential optimization is a recent technique that aims to combine techniques from linear optimization with combinatorial optimization, for extending the classic primal-dual schema to ILPs with non-conventional constraints whose corresponding dual variables are hard to interpret and update. Zhou *et al.* [12] study online cloud resource allocation with job deadlines, and address deadline constraints through compact exponential optimization. Sun *et al.* [13] design an online mechanism to maximize social welfare and tenant utility in

colocation datacenters, using compact exponential ILPs to avoid directly dealing with deadline constraints. The compact infinite optimization framework in this work was inspired in part by the above literature. We highlight two important differences between compact exponential optimization and compact infinite optimization: (i) the former had been applied to deal with non-conventional *linear constraints*, while the latter is applied to deal with *non-linear constraints*; (2) the former transforms the original optimization problem into an ILP with exponentially many primal variables and dual constraints, while for the latter we have infinitely many primal variables (DVFS profiles in this work) and dual constraints. It is our hope that this new method itself may find further applications in hard non-linear optimization problems.

III. System Model

A cloud datacenter provisions multiple types of resources (CPU, GPU, RAM and disk storage) that are virtualized and organized into different types of GPU VMs. The cloud resource provider leases GPU VMs to cloud users. Cloud users arrive dynamically and bid for their job execution on designated GPU VMs. We design an online auction algorithm for the cloud provider, who acts as the auctioneer, to conduct admission control and job scheduling. During the EDR period, the cloud provider is further required to cap its grid power usage under a level dictated in each EDR time slot.

GPU DVFS is used to tune task energy consumption in our auction (Sec. III-B). The maximum power consumption in GPU is by a level of magnitude higher than that of CPU (*e.g.*, NVIDIA GeForce GTX 590 40nm: 365W; Intel Core i7-3770T 22nm: 45W [14]); we simplify energy consumption of CPU and other VM components into a relatively constant term, and concentrate on GPU energy management.

A. Auction mechanism with EDR

When EDR occurs, a signal is sent to the cloud provider, specifying the energy reduction requirement $E_{EDR}^{(t)}$ (to reduce from a pre-negotiated capacity $\mathcal{E}^{(t)}$) in each time slot, and the number of time slots $|T|$ it lasts. Let [X] denote the set of integers {1,2,...,X}. We focus on tasks that can be interrupted and resumed during execution. The cloud resource provider acts as the auctioneer. There are *I* participating users acting as bidders, and their bids arrive randomly in the EDR time span {1,2,...,T}. Users with different tasks usually have different sensitivities to task deferral.

When user i's task arrives, energy reduction of tasks can be achieved by adjusting GPU voltage/frequency $\{V_i,\ f_{ic},\ f_{im}\}$, based on the energy consumption function in Sec. III-B, while task execution time will be extended at the same time. Meanwhile, users will receive their corresponding reward.

Each user submits to the auctioneer a bid that is a 4-tuple: $bid_i = \{B_i, d_i, \delta_i, w_i\}$.

Here d_i is the preferred task completion deadline, and B_i is user i's willingness-to-pay for finishing its task punctually. We use a utility loss factor $\delta_i \in (0, 1]$ to quantify user i's sensitivity to deferred task completion, such that the auctioneer may schedule the execution during a larger time window $[t_i, t_i + \frac{d_i - t_i}{\delta_i}]$ instead of during $[t_i, d_i]$. For a user who does not want any delay, the loss factor $\delta_i = 1$. w_i is the workload of user i's task. In Sec. III-B, we present a performance model that closely tracks task execution time $\mathcal{T}_i$ given GPU frequency and user workload [15]. t_i is the arrival time of user i's task.

Upon a bid's arrival, the auctioneer decides immediately whether to accept it — a binary variable x_i is set to 1 if user i's bid is accepted, and 0 otherwise. Another binary variable $z_i^{(t)}$ represents the schedule of user i's accepted task: $z_i^{(t)} = 1$ if user i's task is executed at time slot t, and 0 otherwise.

Given an energy capacity limit of cloud datacenter at each slot, $\mathcal{E}^{(t)}$, the auctioneer judiciously schedules user task execution, and adjusts GPU voltage/frequency for energy shedding.

We next define a reward function that is offered to user i for its energy conservation:

$$\pi_i = \frac{\mathcal{T}_i^*}{\widetilde{\mathcal{T}}_i} \varrho log(1 + \Delta E_i),\ \Delta E_i = \widetilde{E}_i - E_i^* \tag{1}$$

Here the reward factor ϱ is non-negative and is decided by the resource provider. $\widetilde{E}_i$ is the energy consumption of user *i*'s task in default setting, and E_i^* is the optimal energy consumption in EDR, thus ΔE_i is the reduced energy. $\mathcal{T}_i^*/\widetilde{\mathcal{T}}_i$ is the ratio of the optimal execution time to the execution time in default setting. π_i is a non-decreasing function, when $\Delta E_i = 0$, $\pi_i = 0$. It also reflects that longer task deferral comes with higher remuneration. Once bidder i wins, the auctioneer further computes a reward π_i for its bid.

The true valuation of user i's bid is denoted by v_i, and $cost_i$ represents user i's electricity bill that the provider needs to pay to the power grid.p_i is the payment of user i for his accepted task. $cost_i$ increases with user energy consumption linearly, *i.e.*, $cost_i(E_i)$. If user i is accepted, its utility is $u_i = v_i - p_i$, and the cloud provider's utility is $\sum_{i \in I} p_i - cost_i$. The auction aims to maximize *social welfare*, which is the overall utility of both the cloud provider and users: $\sum_{i \in I}(v_i - p_i) + \sum_{i \in I} p_i - cost_i$, which can be simplified to $\sum_{i \in I} v_i - cost_i$ with payments cancelling themselves. Social welfare maximization depends on *truthful* bids from cloud users. An auction is *truthful* if, for each user, bidding its true valuation always maximizes its own utility, no matter how other users bid.

Lemma 1. (Myerson 1981:) *Let $Pr(B_i)$ be the probability of bidder i winning in an auction and B_{-i} the bidding prices except from user i. A mechanism is truthful if and only if the following holds for a fixed B_{-i}* [16]*:*

1) $Pr(B_i)$ is monotonically non-decreasing in B_i;

2) bidder i is charged $B_i Pr(B_i) - \int_0^{B_i} Pr(B) dB$.

Lemma 1 can be intuitively interpreted as: the payment charged to bidder *i* for a fixed B_i is independent of B_i [17]. This result is later utilized towards establishing the truthfulness of our auction mechanism. Under truthful bidding, the social welfare is $\sum_{i \in I}(B_i - cost_i)$.

B. DVFS of GPU VMs

Owing to the independence of GPUs in VMs, we can adjust the voltage/frequency for each GPU VMs individually.

The relationship between GPU performance and GPU frequency is complex in practice. Cloud datacenter has a default voltage/frequency setting in normal time $\{\widetilde{V},\ \widetilde{f_c},\ \widetilde{f_m}\}$, and the voltage/frequency of GPU VM has a minimum values: $\{V^-,\ f_c^-,\ f_m^-\}$. We use a first-order mathematical model with concise form to simplify the subsequent analysis of DVFS in auction mechanism.

We formulate the general performance model $\mathcal{T}_i$ of GPU-accelerated task execution as follows:

$$\mathcal{T}_i = \mathcal{T}(\frac{1}{f_{ic}}, \frac{1}{f_{im}}, w_i). \tag{2}$$

Here f_{im} and f_{ic} are GPU memory frequency and GPU core frequency, respectively. The performance, in terms of task completion time, depends on GPU frequency scaling and the workload of user i's task. We round the calculated execution time up to facilitate the schedule selection of user i's task.

Power consumption $\mathcal{P}_i$ of each time slot for user *i* is linearly dependant on the GPU's core frequency, and quadratically dependant on its voltage V_i:

$$\mathcal{P}_i = \mathcal{P}(f_{im}, f_{ic}, V_i^2). \tag{3}$$

There are some non-negative coefficients in Eq. 3 that depend on hardware and application characteristics, indicating the sensitivity to memory frequency scaling and core voltage/frequency scaling, respectively. Note that f_{ic} and V_i are correlated through the function $f_{ic} = g(V_i)$.

The runtime energy consumption of a GPU VM to process user i's task can be reformulated as:

$$E_i = \mathcal{T}_i * \mathcal{P}_i = \mathcal{T}(\frac{1}{f_{ic}}, \frac{1}{f_{im}}, \mathcal{W}_i) * \mathcal{P}(f_{im}, f_{ic}, V_i^2) \tag{4}$$

Eq. (4) illustrates that the reduction of GPU VMs' voltage/frequency leads to the extension of execution time and the reduction of power consumption. So we could carefully adjust the voltage/frequency to reduce the energy consumption E_i.

TABLE II: List of Notations

I	# of bidders	T	# of slots in EDR
$\|t\|$	length of each slot t	B_i	bidding price of bidder i
t_i	bidder i's arrival time	d_i	deadline for bidder i's task
δ_i	bidder i's sensitivities to deferment of task execution time		
w_i	workload of user i's task		
π_i	reward function for bidder i's energy reduction		
E_i	energy consumption of bidder i for each time slot		
$\widetilde{E}_i$	energy consumption of bidder i in default setting		
E_i^*	optimal energy consumption of bidder i after DVFS		
$E_{EDR}^{(t)}$	energy reduction amount requested by EDR in slot t		
x_i	accept bidder i's task (1) or not (0)		
$z_i^{(t)}$	accept bidder i's task at time t (1) or not (0)		
$\mathcal{E}^{(t)}$	energy capacity of cloud data center in slot t		
$p^{(t)}$	energy price function (10)		
$\mathcal{W}^{(t)}$	amount of consumed energy at slot t		
C_i	resource capacity of GPU cloud data center		
N_{ir}	demand of type-r resource by bidder i		

IV. Online Auction Mechanism with DVFS

In this section, we focus on energy constraints from EDR, and design and analyze an online algorithm without relying on future information. We first formulate social welfare maximization with DVFS into an ILP in Sec. IV-A. Then we design an online algorithm in Sec. IV-B, and theoretical analysis is in Sec. IV-C.

A. Social Welfare Maximization Problem

Under truthful bidding, the online EDR problem of GPU cloud computing with DVFS can be formulated into the following ILP. Our objective is to maximize social welfare.

$$\text{maximize} \sum_{i\in[I]} (B_i - cost_i)x_i \tag{7}$$

subject to:

$$\mathcal{T}(\frac{1}{f_{ic}}, \frac{1}{f_{im}}, w_i)x_i \le \sum_{t\in[T]:t\ge t_i} z_i^{(t)}, \forall i \in [I], \tag{5a}$$

$$z_i^{(t)} t \le (t_i + \frac{d_i - t_i}{\delta_i})x_i, \forall i \in [I], t \in [T], \tag{5b}$$

$$\sum_{i\in[I]:t_i\le t} \mathcal{P}(f_{im}, f_{ic}, V_i^2)|t| z_i^{(t)} + E_{EDR}^{(t)} \le \mathcal{E}^{(t)}, \forall t \in [T], \tag{5c}$$

$$\widetilde{V} \ge V_i \ge V^-, \widetilde{f_c} \ge f_{ic} \ge f_c^-, \widetilde{f_m} \ge f_{im} \ge f_m^-, \quad x_i, z_i^{(t)} \in \{0,1\}, \forall i \in [I], t \in [T]. \tag{5d}$$

Constraint (5a) guarantees that the number of allocated slots is sufficient for serving a successful bid. (5b) ensures a task is scheduled between its arrival and deadline. User energy reduction is enforced in constraint (5c).

Even in the offline scenario where user tasks and bids are all available, without constraints (5a) and (5b), (7) is a special case of the multidimensional multi-choice 0-1 knapsack problem, known to be NP-hard [18]. We introduce a new strategy, *Compact Infinite ILP*, to hide non-conventional and infinitely many constraints, at the price of involving an infinite number of variables. Then we adopt the classic primal-dual schema for algorithm design in Sec. IV-B to update only a polynomial number of variables.

$$\text{maximize} \sum_{i\in[I]} \sum_{s\in[\mathcal{S}_i]} \sum_{E_i\in[\Omega_i]} (B_i - cost_i)x_{isE_i} \tag{8}$$

subject to:

$$\sum_{i\in[I]} \sum_{s:t\in s} \sum_{E_i\in[\Omega_i]} E_i x_{isE_i} + E_{EDR}^{(t)} \le \mathcal{E}^{(t)}, \forall t \in [T], \tag{6a}$$

$$\sum_{s\in[\mathcal{S}_i]} \sum_{E_i\in[\Omega_i]} x_{isE_i} \le 1, \forall i \in [I], \tag{6b}$$

$$x_{isE_i} \in \{0,1\}, \forall i \in [I], s \in [\mathcal{S}_i], E_i \in [\Omega_i]. \tag{6c}$$

In the Compact Infinite ILP (8), s represents a feasible schedule of bidder i, which satisfies constraints (5a) and (5b). $\mathcal{S}_i$ contains an infinite number of all feasible schedules of user i's task, corresponding to different energy consumption represented by Ω_i

Let x_{isE_i} indicate whether schedule $s \in \mathcal{S}_i$ with energy $E_i \in \Omega_i$ is selected (1) or not (0). Constraints (6a) and (6b)

correspond to (5c) and (5d) in ILP (7). Next, we relax the integrality constraint on x_{is} to $x_{is} \geq 0$ and formulate the dual problem of the LP relaxation of ILP (8). We introduce dual variables u_i and $p^{(t)}$ to constraints (6a) and (6b), respectively.

$$\text{minimize} \sum_{t\in[T]} (\mathcal{E}^{(t)} - E_{EDR}^{(t)})p^{(t)} + \sum_{i\in[I]} u_i \tag{9}$$

subject to:

$$u_i \geq (B_i - cost_i) - \sum_{t:t\in s} E_i p^{(t)}, \forall i \in [I], \forall s \in [\mathcal{S}_i], \forall E_i \in [\Omega_i], \tag{7a}$$

$$u_i, p^{(t)} \geq 0, \forall i \in [I], \forall t \in [T]. \tag{7b}$$

Although the dual LP (9) has an infinite number of constraints, we later choose polynomially many through a dual oracle to update in our primal-dual algorithm. We next present an online auction algorithm that efficiently solves the primal and dual problems simultaneously.

B. Online Primal-dual Algorithm

We design an online auction mechanism considering task scheduling, reward valuation and energy consumption during EDR. A key problem is to decide whether to accept bidder a i's task and how to schedule accepted tasks. How to adjust the user's voltage and current to reduce power consumption is another important problem. If the cloud provider accepts i's task with schedule s, then let $x_{isE_i} = 1, \forall s \in [\mathcal{S}_i], \forall E_i \in [\Omega_i]$, and increase the amount of energy consumption by E_i in each time slot of schedule s. Otherwise, $x_{isE_i} = 0, \forall s \in [\mathcal{S}_i], \forall E_i \in [\Omega_i]$.

To schedule newly arrived tasks over future time slots, we minimize the increase of the dual objective and maintain dual feasibility (7a) by leveraging *complementary slackness*. Once the dual constraint is tight with bidder i's schedule s, the primal variable x_{is} is updated to 1. Therefore, we let u_i be the maximum between 0 and the right hand side (RHS) of constraint (7a):

$$u_i = max\{0, max_{s\in[\mathcal{S}_i]}(B_i - (cost_i + \sum_{s\in\mathcal{S}_i:t\in s} E_i p^{(t)}))\}, \forall i \in [I] \tag{8}$$

In the event that $(B_i - (cost_i + \sum_{s\in\mathcal{S}_i:t\in s} E_i p^{(t)})) > 0$, we have $u_i > 0$ and bidder i's task is scheduled by l which maximizes the RHS of (7a). Otherwise, let $u_i = 0$ and i's task is rejected. In Eq. 8, we can view $p^{(t)}$ as the marginal price per unit energy consumption in at t. Thus $\sum_{t\in T} E_i p^{(t)}$ denotes the cost of serving bidder i's task by schedule s, and the RHS of (7a) is the utility of bidder i. Eq. 8 chooses the schedule of bidders with maximum utility.

RHS minimization in EDR. We formulate the RHS minimization problem for each user i using quadratic programming, and compute the optimal GPU voltage/frequency setting.

$$\text{minimize} \sum_{s\in\mathcal{S}_i:t\in s} p^{(t)} * |t| * \mathcal{P}(f_{im}, f_{ic}, V_i^2) + cost_i \tag{11}$$

subject to:

$$\frac{d_i}{\delta_i} \geq t_i + \mathcal{T}(\frac{1}{f_{ic}}, \frac{1}{f_{im}}, w_i), \tag{9a}$$

$$\widetilde{V} \geq V_i \geq V^-,\ \widetilde{f}_c \geq f_{ic} \geq f_c^-,\ \widetilde{f}_m \geq f_{im} \geq f_m^-, \tag{9b}$$

As mentioned above, GPU core frequency is correlated with GPU core voltage. Thus we transform the minimization problem (11) into a two-variable optimization problem. $p^{(t)}, \forall t \in [T]$ is a constant. Given their independence, we can compute the optimal core and memory frequencies separately. We first solve (11) at each slot and the optimal GPU voltage/frequency setting $\{V^*,\ f_m^*,\ f_c^*\}$ is obtained. Then we have the new execution time $\mathcal{T}_i^*$ and the energy consumption E_i^* of bidder i to participate in our auction mechanism.

When dealing with the offline problem, we can resort to dynamic programming to solve the dual LP exactly. However, for the online problem we are investigating, our auction strives to reserve a certain amount of energy for potential high-value bids in the future. Designing an energy price function is crucial to guarantee the performance of the auction mechanism. We propose a price function $p^{(t)}$ based on realtime supply-demand. Let $\mathcal{W}^{(t)}$ denote the amount of consumed energy at t, and ω be the minimum consumption rate within slots T, *i.e.*, the total energy consumption of all winners is lower bounded by ω times the total energy capacity. We make the following two mild assumptions, which are standard in the literature of online optimization [13].

Assumption 1. *Total energy demand from all bidders is much higher than the energy capacity, and the real energy consumption of all winners is close to capacity,* i.e.*,* $\omega \to 1^-$.

Assumption 2. *The variability of bidder valuations is capped,* i.e.*,* $\frac{\omega\mathcal{L}}{\eta} \leq \frac{B_i}{E_i} \leq \mathcal{U}, \forall i \in [I]$*,* $\mathcal{L}$ *and* $\mathcal{U}$ *are lower and upper bounds of user valuation per unit of energy, respectively. Here* $\eta > 1$ *is a constant that scales down the price at the beginning of the auction.*

We design the energy price function as follows:

$$p(\mathcal{W}^{(t)}) = \frac{\omega\mathcal{L}}{\eta}(\frac{\eta\mathcal{U}}{\omega\mathcal{L}})^{\frac{\mathcal{W}^{(t)}}{\mathcal{E}^{(t)} - E_{EDR}^{(t)}}} \tag{10}$$

At the beginning of the auction, there is no energy consumption, *i.e.*, $\mathcal{W}^{(0)} = 0$, we have $p(\mathcal{W}^{(0)}) = \frac{\omega\mathcal{L}}{\eta}$. The initial marginal price $p(\mathcal{W}^{(0)})$ is carefully designed to ensure that it is low enough such that any task can be scheduled to run (in Assumption.2). When energy consumption reaches capacity, *i.e.*, $\mathcal{W}^{(0)} = \mathcal{E}^{(t)}$, we have $p(\mathcal{W}^{(t)}) = \mathcal{U}$. The value of $\mathcal{U}$ ensures that any schedule using this time slot will lead to a negative net utility.

Algorithm 1 with task scheduling Algorithm 2 is our online auction mechanism. We first initialize primal and dual variables, GPU voltage/frequency and consumed energy $\mathcal{W}^{(t)}$ in line 1. Upon arrival of a bidder i, Algorithm 2 calculates the price of feasible slots and sorts the slots in lines 1-4. Then we compute the longest execution time $\lceil\mathcal{T}^-\rceil$ in line 5, calculate the minimum cost $\Delta_i^{\mathcal{T}}$ and energy consumption for each $\lceil\mathcal{T}\rceil$ within $[\lceil\widetilde{\mathcal{T}}_i\rceil, \lceil\mathcal{T}^-\rceil]$ (lines 6-10). Line 11 selects the optimal execution time $\mathcal{T}^*$ with minimum cost. Line 12 updates GPU voltage/frequency settings and energy consumption. Line 13 selects the best schedule $\widetilde{s}$ under the reduced energy capacity $\mathcal{E}^{(t)} - E_{EDR}^{(t)}$. Line 14 computes the corresponding reward for bidder i due to its energy reduction. Then we update dual

Algorithm 1 Online Auction Algorithm

Input: $\{bid_i\}$, $\{\mathcal{E}^{(t)}\}$, $\{E_{EDR}^{(t)}\}$, $|T|$;
Output: Integral solutions of ILP (7);

1: Initialize: $x_i = 0$, $z_i^{(t)} = 0$, $x_{is} = 0$, $u_i = 0$, $V = \widetilde{V}$, $f_c = \widetilde{f}_c$, $f_m = \widetilde{f}_m$, $\forall i \in [I], t \in [T], s \in \mathcal{S}_i$, $p(\mathcal{W}^{(0)}) = \frac{\sigma\mathcal{L}}{\eta}$, $\mathcal{W}^{(0)} = 0$, user cost $\mathcal{C}_i = 0$;
2: **Upon the arrival of new bidder i's task**
3: $(\mathcal{U}_i, \widetilde{s}, \mathcal{C}_i)$ = Sub($\{bid_i\}$, $\{p^{(t)}\}$, $\{\mathcal{E}^{(t)}\}$, $\{\mathcal{W}^{(t)}\}$, $\{E_{EDR}^{(t)}\}$, $|T|$);
4: **if** $\mathcal{U}_i > 0$ **then**
5: *// Update primal and dual variables.*
6: **Update:** $x_{is} = 1$, $x_i = 1$, $\mathcal{W}^{(t)} = \mathcal{W}^{(t)} + \widetilde{E}_i$ or $\mathcal{W}^{(t)} = \mathcal{W}^{(t)} + E_i$ within EDR, $p^{(t)} = p^{\mathcal{W}^{(t)}}$;
7: **end if**
8: **if** $x_i = 1$ **then**
9: Accept bidder i, schedule its task in time slot t according to $\widetilde{s}$.
10: Charge $\mathcal{C}_i$ from bidder i;
11: **else**
12: Reject bidder i;
13: **end if**

Algorithm 2 Task Schedule and DVFS Algorithm *Sub*

Input: $\{bid_i\}$, $\{p^{(t)}\}$, $\{\mathcal{E}^{(t)}\}$, $\{\mathcal{W}^{(t)}\}$, $\{E_{EDR}^{(t)}\}$, $|T|$;
Output: $\mathcal{U}_i$, $\widetilde{s}$, $\mathcal{C}_i$;

1: **for all** $t \in [t_i, t_i + \frac{d_i - t_i}{\delta_i}]$ **do**
2: Calculate the price $p^{(t)}$ of each slot. Add slot t to the set Π_i if $\mathcal{W}^{(t)} + \widetilde{E}_i \leq \mathcal{E}^{(t)}$.
3: **end for**
4: Sort slots in Π_i by cost $p^{(t)}$ in non-decreasing order;
5: Compute the longest execution time $\lceil \mathcal{T}^- \rceil$ under the minimum setting according to Eq. 2.
6: **for all** $\lceil \mathcal{T} \rceil \in [\lceil \widetilde{\mathcal{T}}_i \rceil, \lceil \mathcal{T}^- \rceil]$ **do**
7: Calculate the minimum cost $\Delta_i^{\mathcal{T}}$ of user i according to quadratic programming (11).
8: Compute the optimal GPU voltage/frequency setting $\{V^{\mathcal{T}}, f_m^{\mathcal{T}}, f_c^{\mathcal{T}}\}$ of bidder i.
9: Calculate the energy consumption $E_i^{\mathcal{T}}$ of bidder i by Eq. 4.
10: **end for**
11: $\mathcal{T}^* = argmin_{\lceil \mathcal{T} \rceil \in [\lceil \mathcal{T}^- \rceil, \lceil \widetilde{\mathcal{T}}_i \rceil]} \{\Delta_i^{\mathcal{T}}\}$;
12: Set: $\{V_i^*, f_{im}^*, f_{ic}^*\} = \{V^{\mathcal{T}^*}, f_m^{\mathcal{T}^*}, f_c^{\mathcal{T}^*}\}$, $E_i^* = E_i^{\mathcal{T}^*}$;
13: Select $\mathcal{T}_i^*$ slots with minimum cost to $\widetilde{s}$; Set real completion slot m_i;
14: Compute bidder i's *Reward*: $\pi_i = \frac{\mathcal{T}_i^*}{\widetilde{\mathcal{T}}_i} \varrho log(1 + E_i^* - \widetilde{E}_i)$;
15: **Update:** Schedule $\widetilde{s}$, electricity bill $cost_i$,
bidder i's cost: $\mathcal{C}_i = \sum_{t \in [\widetilde{s}]} E_i^* p^{(t)} - \pi_i$,
utility: $\mathcal{U}_i = B_i - cost_i - \sum_{t \in [\widetilde{s}]} E_i^* p^{(t)} + \pi_i$;

variables and the cost of bidder in line 15. If bidder i can obtain positive utility, primal variables x_{is} and x_i are updated accordingly, and dual variables are also updated (lines 4-7). If bidder i is accepted, we will schedule its task according to $\widetilde{s}$ and collect payment from it (lines 8-13).

C. Theoretical Analysis

Theorem 1. *Algorithm 1 computes a feasible solution to ILP (7).*

Proof: x_i is initialized to 0 and updated to 1 in line 6 (Algorithm 1), and $\{V^*, f_m^*, f_c^*\}$ are updated according to quadratic programming (11) in line 16 (Algorithm 2). Therefore, the solution of our algorithm satisfies constraint (5d). For each bidder i, we select time slots satisfying energy capacity limits in line 2 (Algorithm 2), which guarantee the validity of constraint (5c). Line 6 and line 17 in Algorithm 2 guarantee that the schedule $\widetilde{s}$ for bidder i's task satisfies constraints (5a) and (5b). In conclusion, we obtain a feasible solution to ILP (7). □

Theorem 2. *ILP (7) is solved by Algorithm 1 in polynomial time.*

Proof: Line 1 can be executed in linear time for the initialization of functions, primal and dual variables (Algorithm 1). Upon the arrival of new bidder i's task in EDR period, we take $O(T)$ time to calculate the price of each slot in lines 1-4 (Algorithm 2). Line 5 computes the longest execution time $\lceil \mathcal{T}^- \rceil$ in $O(1)$ time. Lines 6-10 calculate the minimum cost $\Delta_i^{\mathcal{T}}$ and energy consumption for each $\lceil \mathcal{T} \rceil$ within $[\lceil \widetilde{\mathcal{T}}_i \rceil, \lceil \mathcal{T}^- \rceil]$ in $O(T)$ time, and lines 11-13 select the best schedule for each task in $O(1)$ time. Bidder i's reward, cost and utility are updated in linear time (lines 14-15). The body of if in lines 4-7 (Algorithm 1) takes $O(T^2)$ time to update the primal and dual variables, and compute the payment. The last step of Algorithm 1 announces the auction decision in constant time (lines 8-13). In summary, Algorithm 1 runs in polynomial time. □

Theorem 3. *The online auction mechanism in Algorithm 1 is truthful.*

Proof: (Bidding price) Leveraging Lemma 1, if the payment charged to user i is independent of its bidding price B_i, the auction mechanism is truthful. In Algorithm 1, the payment from a user i is $\mathcal{C}_i = \sum_{t \in [\widetilde{s}]} E_i p^{(t)} - \pi_i$. Here $p^{(t)}$ depends only on the amount of energy that has been allocated and user i's demand according to Eq. (10), and $\pi_i = (\mathcal{T}_i^* / \widetilde{\mathcal{T}}_i) \varrho log(1 + E_i^* - \widetilde{E}_i)$, which is independent of bidding price B_i.
(Workload) The payment depends on the user i's demand as mention above. If user lies about its workload w_i: 1) workload $<$ true workload, user task cannot be finished. 2) workload $>$ true workload, user payment is higher than true payment, thus user cannot lie about its workload to improve its utility.
(Deadline) If a user lies about its deadline d_i: 1) deadline $>$ true deadline, its task may not be completed within the user satisfaction period. 2) deadline $<$ true deadline, the risk of being rejected increases. If the task is accepted, user payment is higher than true payment according to Algorithm 2, and utility will decrease under fake deadline.
(Arrival time) The price function in Eq. 10 is a non-decreasing function of the amount of allocated energy. The price in each feasible time slot will be higher if user delays arrival time. Lying about arrival time is not helpful to its utility. In summary, our online auction mechanism is truthful. □

The *competitive ratio* is the upper-bound ratio of the social welfare achieved by the optimal solution of ILP 7 to the social welfare achieved by our auction mechanism. Next, we will analyze the competitive ratio of Algorithm 1 in Theorem 4.

Theorem 4. *The online auction mechanism in Algorithm 1 is* $\frac{k}{k-1}\alpha$*-competitive with* $\alpha = ln(\frac{\eta\mathcal{U}}{\omega\mathcal{L}})$*. When* η *satisfies* $\eta - 1 = ln(\frac{\eta\mathcal{U}}{\omega\mathcal{L}})$*, the minimum competitive ratio is* η*.*

The proof of Theorem 4 is in the technical report [19]. If we consider the case that competition for energy is intense, then ω is close to 1. When $\frac{\mathcal{U}}{\mathcal{L}}$ is 2, the competitive ratio is close to 2.68, as illustrated in Fig. 2.

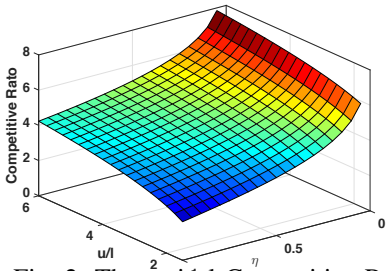

Fig. 2: Theoretical Competitive Ratio.

V. Auction Mechanism with GPU VM Constraints

In this section, we consider an one-round auction at the beginning of each EDR period like California's Demand Response Auction [20]. What's different is that cloud users, instead of investor-owned utilities, are bidders in our auction mechanism. We collect all tasks which are willing to extend their execution time to save energy in EDR. We first formulate the social welfare maximization into an ILP in Sec. V-A. Then we design an approximation algorithm in Sec. V-B, and the theoretical analysis is in Sec. V-C.

A. Social Welfare Maximization

In the one-round auction, we consider both energy and resource constraints. Each user i submits a demand N_{ir} of GPU VMs that comprise of GPU, CPU, RAM, and disk storage. We assume that tasks from all users are available at the beginning, and target effective offline algorithms that exploit such full information. The social welfare maximization problem can be formulated into the following ILP. The resource capacity of the cloud datacenter is denoted as C_r.

$$\text{maximize} \sum_{i\in[I]}(B_i - cost_i)x_i \tag{13}$$

subject to:

$$\sum_{i\in[I]} N_{ir}x_i \le C_r, \forall r \in [R], \tag{11a}$$

$$\sum_{i\in[I]} \mathcal{T}(\frac{1}{g(V_i)}, \frac{1}{f_{im}}, w_i) * \mathcal{P}(f_{im}, f_{ic}, V_i^2)x_i + E_{EDR} \le \mathcal{E}, \tag{11b}$$

$$x_i \in \{0,1\}, \forall i \in [I], \tag{11c}$$

$$\widetilde{V} \ge V_i \ge V^-, \widetilde{f_c} \ge f_{ic} \ge f_c^-, \widetilde{f_m} \ge f_{im} \ge f_m^-.$$

Constraint (11a) models the cloud's resource capacity. (11b) enforces users energy reduction. This original ILP involves four groups of variables and non-conventional constraints, precluding a direct application of the classic primal-dual schema of algorithm design. We resort to the *Compact Infinite ILP* technique to hide non-conventional constraints that arise from GPU DVFS. In the Compact Infinite ILP (14), Ω_i represents all valid energy consumption of tasks that satisfy constraint (11b). x_{iE_i} indicates whether to accept user i with energy consumption $E_i \in \Omega_i$.

$$\text{maximize} \sum_{i\in[I]} \sum_{E_i\in[\Omega_i]} (B_i - cost_i)x_{iE_i} \tag{14}$$

subject to:

$$\sum_{i\in[I]} N_{ir}x_{iE_i} \le C_r, \forall r \in [R], \tag{12a}$$

$$\sum_{i\in[I]} \sum_{E_i\in[\Omega_i]} E_i x_{iE_i} \le \mathcal{E} - E_{EDR}, \tag{12b}$$

$$\sum_{E_i\in[\Omega_i]} x_{iE_i} \le 1, \forall i \in [I], \tag{12c}$$

$$x_{iE_i} \in \{0,1\}, \forall i \in [I], E_i \in [\Omega_i]. \tag{12d}$$

With constraints (12a) and (12c) alone, ILP (14) degrades into the classic knapsack problem known to be NP-hard. Fortunately, Thanks to the compact-infinite reformulation of the ILP, we are now able to leverage the primal-dual schema towards efficient and effective approximation algorithm design. We first relax integer constraints $x_{iE_i} \in \{0,1\}$ to $x_{iE_i} \ge 0$, and introduce dual variables h_r, Q_i and y to constraints (12a), (12b) and (12c) respectively. The dual LP of the LP relaxation of ILP (14) can then be formulated:

$$\text{minimize}(\mathcal{E} - E_{EDR})y + \sum_{r\in[R]} C_r h_r + \sum_{i\in[I]} Q_i \tag{15}$$

subject to:

$$Q_i + E_i y + \sum_{r\in[R]} N_{ir}h_r \ge B_i - cost_i, \forall i \in [I], \forall E_i \in [\Omega_i], \tag{13a}$$

$$y, h_r \ge 0, \forall r \in [R]. \tag{13b}$$

Due to the infinitely many energy consumption profiles that arise from DVFS, ILP (14) is defined upon infinitely many variables. Correspondingly, the dual LP has infinitely many constraints. We next design a *dual oracle* to help update only a polynomial number of variables, for efficiently computing an approximate solution.

B. Algorithm

We design an offline mechanism considering both EDR energy reduction and resource capacity. In our mechanism, we select the bid with maximum unit resource and energy value from the remaining users. We regard the following fraction as the value of a unit-weight resource and energy.

$$\Theta_i = \frac{B_i - cost_i}{E_i y + \sum_{r\in[R]} N_{ir}h_r}, \forall i \in [I], \forall E_i \in [\Omega_i] \tag{14}$$

In our auction, the dual variable vector h_r is designed as follows.

$$h_r = \frac{2}{C_r}\exp((\zeta - 1) * \frac{\mathcal{Z}_r}{C_r - \mathcal{H}_r}),\ \mathcal{Z}_r > 0. \tag{15}$$

Here $\mathcal{Z}_r$ is allocated resource r in our auction, $\mathcal{H}_r$ is the maximum required amount of resource r by all users, *i.e.*, $\mathcal{H}_r = max_{i\in[I]}N_{ir}$. ζ is the minimum ratio between

the resource capacity and the maximum required amount of resource r, *i.e.*, $\zeta = min_{r\in[R]}\frac{C_r}{\mathcal{H}_r}$.

We next define the dual variable y which corresponds to constraint (12b). According to our decision-making mechanism, we calculate the maximum unit resource and energy value within the user's deadline. The energy consumption minimization problem for each user i is formulated as a quadratic programm.

$$\text{minimize} \quad \mathcal{T}(\frac{1}{g(V_i)}, \frac{1}{f_{im}}, \mathcal{W}_i) * \mathcal{P}(f_{im}, f_{ic}, V_i^2) \tag{16}$$

$$\text{subject to:} \quad \mathcal{T}(\frac{1}{f_{ic}}, \frac{1}{f_{im}}, w_i) \geq \frac{\widetilde{\mathcal{T}}_i}{\delta_i}, \tag{16a}$$

$$\widetilde{V} \geq V_i \geq V^-,\ \widetilde{f}_c \geq f_{ic} \geq f_c^-,\ \widetilde{f}_m \geq f_{im} \geq f_m^-, \tag{16b}$$

Here $\widetilde{\mathcal{T}}_i$ is user i's execution time under the default setting, and δ_i is i's sensitivity to task deferral. The optimal energy and execution time are computed through (16), which helps us define the dual variable y. $\mathcal{D}$ represents allocated energy in the algorithm. $\mathcal{J}$ is the maximum required amount of energy by all users, *i.e.*, $\mathcal{J} = max_{i\in[I]}E_i$. ξ is the minimum ratio between the energy capacity and the maximum required amount of energy of all users, *i.e.*, $\xi = \frac{\mathcal{E}-E_{EDR}}{\mathcal{J}}$.

$$y = \frac{2}{\mathcal{E}-E_{EDR}}\exp((\xi-1)*\frac{\mathcal{D}}{\mathcal{E}-E_{EDR}-\mathcal{J}}),\ \mathcal{D}>0. \tag{17}$$

Algorithm 3 Primal-dual algorithm

Input: $\{bid_i\}$, $\{\mathcal{E}\}$, E_{EDR}, C_r, $\{N_{ir}\}$;
Output: Integral solutions of ILP (13);
1: Initialize: $x_i = 0$, $x_{iE_i} = 0$, $V = \widetilde{V}$, $f_c = \widetilde{f}_c$, $f_m = \widetilde{f}_m$, user cost $\mathcal{C}_i = 0$, $h_r(0) = \frac{1}{C_r}$, $y(0) = \frac{1}{\mathcal{E}-E_{EDR}}$, $\mathcal{Z}_r = 0$, $\mathcal{D} = 0$;
2: **Upon the arrival of all bidders**
3: **while** Users set $[I] \neq NULL$ and $\mathcal{Z}_r + \mathcal{H}_r \leq C_r$ and $\mathcal{D}+\mathcal{J} \leq \mathcal{E} - E_{EDR}$ **do**
4: *// Users minimize energy consumption to improve the possibility of winning the bid.*
5: Compute the optimal GPU voltage/frequency setting $\{V_i^*, f_{im}^*, f_{ic}^*\}$ of bidder i according to quadratic programming (16).
6: Calculate the new energy consumption E_i^* of bidder i by Eq. 4, and the electricity bill $cost_i$.
7: $i^* = argmax_{i\in[I]}\{\frac{B_i - cost_i}{E_i y + \sum_{r\in[R]} N_{ir}h_r}\}$;
8: Remove i^* from the set $[I]$;
9: **Update**: $x_{i^*E_i^*} = 1$, $x_{i^*} = 1$, $\mathcal{Z}_r = \mathcal{Z}_r + N_{i^*r}, \forall r \in [R]$, $h_r = h_r(\mathcal{Z}_r), \forall r \in [R]$ according to (15), $\mathcal{D} = \mathcal{D} + E_{i^*}^*$, $y = y(\mathcal{D})$ according to (17), $Q_{i^*} = B_{i^*}$;
10: **end while**

Algorithm 3 is our task allocation mechanism with resource and energy constraints. We first initialize primal and dual variables in Line 1. The *while* loop (lines 3-10) iteratively selects the best bidder. Optimal energy consumption is computed in lines 5-6. Then we choose user i^* with maximum unit resource and energy value (line 7), remove the selected user from set $[I]$ and update the corresponding primal and dual variables in lines 8-9.

Next, we design a payment mechanism in Algorithm 4 leveraging Myerson's characterization. This payment algorithm works in concert with Algorithm 3 to form a *truthful* auction. In Algorithm 4, we compute the payment $\widetilde{pay}_i$ for each winning bidder i, through the *while* loop with $i\backslash[I]$. Meanwhile, the maximum unit resource and energy value is $\Theta_{i^*} = \frac{B_{i^*} - cost_{i^*}}{E_{i^*}y + \sum_{r\in[R]} N_{i^*r}h_r}$, and we obtain the minimum price that user i should bid to be selected in the algorithm. The payment is the minimum of $\{pay_i, \widetilde{pay}_i\}$.

Algorithm 4 Payment algorithm

Input: $\{bid_i\}$, $\{\mathcal{E}\}$, E_{EDR}, C_r, $\{N_{ir}\}$;
Output: Payment pay_i;
1: **for all** $i \in [I]$ and $x_i = 1$ **do**
2: $pay_i = B_i$, remove i from the set $[I]$
3: **while** Users set $[I] \neq NULL$ and $\mathcal{Z}_r + N_{ir} \leq C_r$ and $\mathcal{D} + E_i^* \leq \mathcal{E} - E_{EDR}$ **do**
4: $i^* = argmax_{i\in[I]}\{\frac{B_i - cost_i}{E_i y + \sum_{r\in[R]} N_{ir}h_r}\}$;
5: Remove i^* from the set $[I]$;
6: $\widetilde{pay}_i = \frac{B_{i^*} - cost_{i^*}}{E_{i^*}y + \sum_{r\in[R]} N_{i^*r}h_r} * \{E_i y + \sum_{r\in[R]} N_{ir}h_r\}$;
7: **Update**: $pay_i = min\{pay_i, \widetilde{pay}_i\}$, $\mathcal{Z}_r = \mathcal{Z}_r + N_{i^*r}, \forall r \in [R]$, $h_r = h_r(\mathcal{Z}_r), \forall r \in [R]$ according to (15), $\mathcal{D} = \mathcal{D} + E_{i^*}^*$, $y = y(\mathcal{D})$ according to (17), $Q_{i^*} = B_{i^*}$;
8: **end while**
9: **end for**

As shown in Algorithm 5, the auction framework consists of two phases. In the winner selection phase, Algorithm 3 computes a feasible solution to determine the winning bids. The payment scheme in Algorithm 4 is used to guarantee the truthfulness.

Algorithm 5 Auction mechanism

Input: $\{bid_i\}$, $\{\mathcal{E}\}$, E_{EDR}, w, C_r, $\{N_{ir}\}$;
1: Compute feasible allocation solution by Algorithm 3;
2: Compute payments for winning bids by Algorithm 4;
3: **for all** user $i \in [I]$ **do**
4: **if** $x_i = 1$ **then**
5: Process its task and charge a price pay_i;
6: **end if**
7: **end for**

C. Analysis

Theorem 5. *Algorithm 3 computes a feasible solution to ILP (13).*

Proof: Primal variable $x_i, \forall i \in [I]$ is initialized to 0 (line 1) and updated to 1 (line 9 or 15), and $\{V^*, f_m^*, f_c^*\}$ are updated according to (16) in line 5. The critical conditions in line 3 guarantee that the allocated resources and energy are limited by their respective capacities. These conditions can also be guaranteed in line 14. Therefore, constraints (11a) and (11b) are satisfied. In summary, Algorithm 3 computes a feasible solution to ILP (13). □

Theorem 6. *The computational complexity of Algorithm 3 and Algorithm 4 is polynomial.*

The proof of Theorem 6 is in the technical report [19].

Theorem 7. *Algorithm 3 is a φ-approximate algorithm for ILP (13), in which $\varphi \leq \frac{\psi}{ln(\frac{2}{1+R})+(\psi-1)} + 1$, and $\psi = min\{\zeta, \xi\}$.*

The proof of Theorem 7 is in the technical report [19].

Theorem 8. *Auction mechanism 5 with payment algorithm 4 is truthful.*

The proof of Theorem 8 is in the technical report [19].

VI. Simulations

We evaluate the performance of our auction mechanisms through trace-driven simulations. We exploit the trace in Google Cluster Data [21]: ClusterData2011-2 provides information of tasks workload, and TraceVersion1 contains information including start time, deadline, and resource demands (CPU and RAM) for each cloud user. We translate each task into a bid, requesting resources at demands extracted from the traces. We assume that the length of each time slot t is an hour. The default setting of GPU voltage $\widetilde{V}$, core frequency $\widetilde{f}_c$ and memory frequency $\widetilde{f}_m$ are: $\{1.5V, 1000MHZ, 2000MHZ\}$. According to the performance function and power function, we compute the execution time and energy consumption of each task i with known workload. Each task's deadline is randomly generated between its arrival time and the system end time.

EDR signals arrive every two hours during a [200,600]-hour span; each EDR event lasts 10 minutes. We scale the total energy reduction trace such that the EDR signals represent an overall energy shedding of at most 25% of the total energy consumed by the cloud data center.

A. Performance of online auction mechanism

We first simulate the impact on social welfare by $\frac{\mathcal{U}}{\mathcal{L}}$. The number of users is set to [100,600]. We generate the bidding price of each job by multiplying the overall energy demands and by unit prices randomly picked from range $[\mathcal{L}, \mathcal{U}]$. The default values are $\mathcal{L} = 5$ and $\mathcal{U} = 15$.

We next study social welfare achieved by Algorithm 1 in Fig. 3 and Fig. 4. Social welfare under different number of users and $\frac{\mathcal{U}}{\mathcal{L}}$ is illustrated in Fig. 3. When the number of users grows, the number of bids with larger bidding price also increases. Therefore, social welfare has a steady upward trend. However, due to the limitation of total amount of resources, social welfare gradually plateaus. Furthermore, the bidding price rises with the increase of $\frac{\mathcal{U}}{\mathcal{L}}$, thus high value bids lead to a higher social welfare. Fig. 4 plots social welfare under different time slots. Our algorithm is able to allocate more tasks as the length of the system increases, manifesting a positive correlation between the number of slots and social welfare.

Fig. 5 shows that the competitive ratio of online Algorithm 1 fluctuates moderately with the variation of number of users. As proven in Theorem 4, the competitive ratio depends on $\frac{\eta\mathcal{U}}{\mathcal{L}}$ and grows as $\frac{\mathcal{U}}{\mathcal{L}}$ increases. The observed competitive ratio is better than the theoretical bound, and remains below 1.5; this can be partly explained by the fact that the theoretical bound is a pessimistic worst case scenario uncommon in practice. The ratio fluctuates with user population and sightly decreases with as $\frac{\mathcal{U}}{\mathcal{L}}$ decreases.

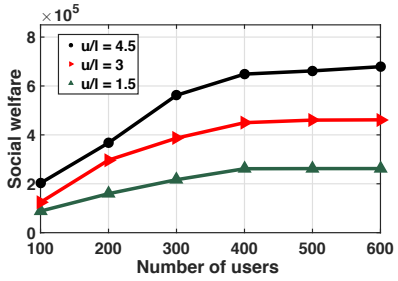

Fig. 3: Social welfare with different value of $\frac{\mathcal{U}}{\mathcal{L}}$.

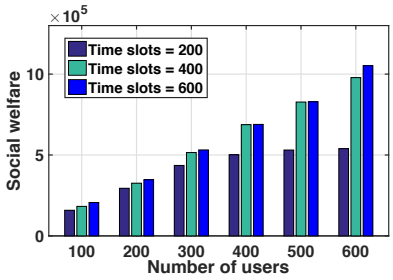

Fig. 4: Social welfare with different number of time slots.

B. Performance of Algorithm 3

Fig. 6 plots the social welfare under different number of users and resources. Our offline auction Algorithm 3 achieves a higher social welfare when there is a larger number of users participating the auction. We also observe that the number of resource types doesn't influence the auction result as the winner determination process is not affected by the change of the value of R. As shown in Fig. 7, the primal-dual approximation algorithm can achieve a higher user satisfaction than the heuristic algorithm. The gap between the two algorithms widens as the number of users grows, which means that Algorithm 3 can be more efficient for a larger user population. Here our primal-dual approximation algorithm outperforms the heuristic algorithm.

Finally, we examine the performance of Algorithm 3 under different number of users and types of resources. Fig. 8 shows that the approximation ratio increases (the social welfare becomes lower) as the number of resource types increases, because there are more restrictions involved in constraint (7a), imposing greater challenges for the approximation algorithm to approach the optimal solution.

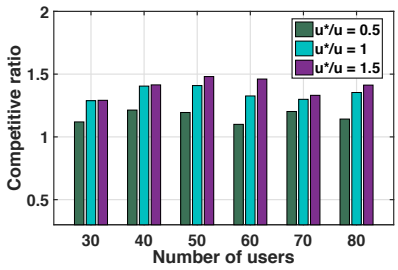

Fig. 5: Competitive ratio with different ratio of $\frac{\mathcal{U}}{\mathcal{L}}$.

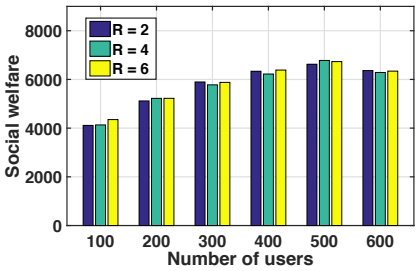

Fig. 6: Social welfare under different types of cloud resources.

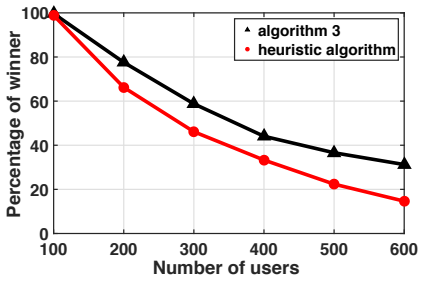

Fig. 7: Percentage of winner, approximation algorithm VS. heuristic algorithm.

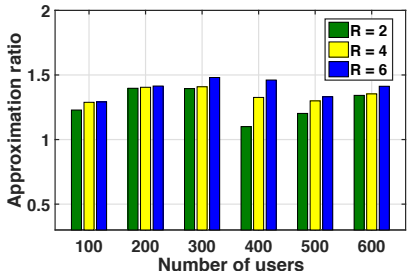

Fig. 8: Offline approximation ratio with different R.

VII. Conclusion

A crucial problem in GPU cloud computing is excessive energy consumption. DVFS represents an effective method to reduce energy consumption of GPU VMs. This work studies demand response of GPU cloud computing. We first consider an online demand response scenario where users arrive stochastically, towards maximizing social welfare and reducing energy consumption by DVFS. We address the challenge posed by DVFS through a new technique of compact infinite ILP coupled with dual oracles. We further extend the investigation to scenarios where energy and resource are both limited. We hope that the compact infinite optimization framework may shed light on a broad range of optimization problems beyond GPU DVFS.

Acknowledgement. This work was supported in part by grants from NSFC (61772385, 61373040, 61572370, 61571335, 61628209) and Hubei Science Foundation (2016CFA030, 2017AAA125), and by China Scholarship Council (CSC).

References

[1] "Clicking Clean: How Companies are Creating the Green Internet April 2014," *Sb Business Weekly*, 2014.

[2] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The Cost of a Cloud:Research Problems in Data Center Networks," *Acm Sigcomm Computer Communication Review*, vol. 39, no. 1, pp. 68–73, 2008.

[3] A. Verma, P. Ahuja, and A. Neogi, "pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems," in *Acm/ifip/usenix International Conference on MIDDLEWARE*, 2008, pp. 243–264.

[4] 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*, vol. 33, no. 12, pp. 2540–2553, 2015.

[5] X. Zhang, Z. Huang, C. Wu, Z. Li, and F. C. M. Lau, "Online Auctions in IaaS Clouds:Welfare and Profit Maximization with Server Costs," *Acm Sigmetrics Performance Evaluation Review*, vol. PP, no. 99, pp. 1–14, 2016.

[6] S. K. Sonkar and M. U. Kharat, "A Review on Resource Allocation and VM Scheduling Techniques and a Model for Efficient Resource Management in Cloud Computing Environment," in *International Conference on ICT in Business Industry and Government*, 2017.

[7] A. Fayyaz, M. U. S. Khan, and S. U. Khan, "Energy Efficient Resource Scheduling through VM Consolidation in Cloud Computing," in *International Conference on Frontiers of Information Technology*, 2015, pp. 65–70.

[8] R. Zhou, Z. Li, and C. Wu, *An Emergency Demand Response Mechanism for Cloud Computing*. ACM, 2016.

[9] L. Zhang, S. Ren, C. Wu, and Z. Li, "A Truthful Incentive Mechanism for Emergency Demand Response in Colocation Data Centers," in *IEEE INFOCOM*, 2015.

[10] C.-Y. Hsieh, J.-G. Park, and N. Dutt, "Memory-aware Cooperative CPU-GPU DVFS Governor for Mobile Games," in *Embedded Systems For Real-time Multimedia*, 2015.

[11] X. Mei, Q. Wang, and X. Chu, "A Survey and Measurement Study of GPU DVFS on Energy Conservation," *Digital Communications and Networks*, 2016.

[12] R. Zhou, Z. Li, C. Wu, and Z. Huang, "An Efficient Cloud Market Mechanism for Computing Jobs With Soft Deadlines," *IEEE/ACM Transactions on Networking*, vol. PP, no. 99, pp. 793–805, 2017.

[13] Q. Sun, C. Wu, Z. Li, and S. Ren, "Colocation Demand Response: Joint Online Mechanisms for Individual Utility and Social Welfare Maximization," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3978–3992, 2016.

[14] S. Mittal and J. S. Vetter, "A survey of methods for analyzing and improving gpu energy efficiency," *Acm Computing Surveys*, vol. 47, no. 2, pp. 1–23, 2014.

[15] R. Nath and D. Tullsen, "The CRISP Performance Model for Dynamic Voltage and Frequency Scaling in a GPGPU," in *International Symposium on Microarchitecture*, 2015, pp. 281–293.

[16] R. B. Myerson, "Optimal Auction Design," *Mathematics of Operations Research*, vol. 6, no. 1, pp. 58–73, 1981.

[17] A. Gopinathan, Z. Li, and C. Wu, "Strategyproof Auctions for Balancing Social Welfare and Fairness in Secondary Spectrum Markets," in *INFOCOM, 2011 Proceedings IEEE*, 2011, pp. 3020–3028.

[18] N. Cherfi and M. Hifi, "A Column Generation Method for the Multiple-choice Multi-dimensional Knapsack Problem," *Computational Optimization and Applications*, vol. 46, no. 1, pp. 51–73, 2010.

[19] *technical report*, https://www.dropbox.com/s/s9ifpldv4rsfpx8/main0.pdf?dl=0.

[20] *California demand response auction*, https://www.platts.cn/latest-news/electric-power/portland-maine/california-puc-approves-demand-response-auction-21844252.

[21] *Google cloud computing*, http://www.chinanews.com/it/it-itxw/news/2009/09-10/1860250.shtml.