Data Center Participation in Demand Response Programs with Quality-of-Service Guarantees

Yijia Zhang
zhangyj@bu.edu
Boston University
Boston, MA

Ioannis Ch. Paschalidis
yannisp@bu.edu
Boston University
Boston, MA

Ayse K. Coskun
acoskun@bu.edu
Boston University
Boston, MA

ABSTRACT

Demand response helps stabilize the power grid and offers opportunities for consumers to reduce their cost by regulating their power consumption to follow market requirements. Regulation service reserves (RSRs) is a specific form of a demand response program, requiring participants to regulate their power to follow a dynamically-changing target that is updated every few seconds. In return, participants' electricity bill is reduced in proportion to the reserves they provide. Data centers are significant power consumers, and they are good candidates to participate in RSRs because they have the flexibility to regulate their power consumption through various strategies. Previous work in this area has proposed power regulation policies that enable data centers to participate in RSRs, but without providing guarantees on the Quality-of-Service (QoS) of the jobs running on the data center. This paper develops a strategy for a data center to provide RSR while offering QoS guarantees, expressed in terms of the sojourn time of jobs. The proposed policy regulates data center power through power capping and job scheduling, following the spirit of a Generalized Processor Sharing (GPS) policy. Parameters in our policy are calculated so as to minimize the electricity cost under QoS constraints. The key in guaranteeing QoS is to determine an acceptable range for policy parameters using a queueing theoretic result for delay. We evaluate our policy in both large-scale simulations as well as real-system experiments on a small cluster. We demonstrate that our policy enables data centers to participate in RSRs and reduces their electricity bill by 14-51% while providing guarantees on the QoS of the jobs.

CCS CONCEPTS

• **Hardware** → **Smart grid**; **Enterprise level and data centers power issues**; • **Information systems** → *Data centers*; • **General and reference** → Design; • **Computing methodologies** → Modeling methodologies.

KEYWORDS

data center, demand response, quality of service, power market

e-Energy'19, June 25–28, 2019, Phoenix, AZ
© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-6671-7/19/06…$15.00
https://doi.org/10.1145/3307772.3328309

ACM Reference Format:
Yijia Zhang, Ioannis Ch. Paschalidis, and Ayse K. Coskun. 2019. Data Center Participation in Demand Response Programs with Quality-of-Service Guarantees. In *Proceedings of ACM International Conference on Future Energy Systems (e-Energy'19).* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3307772.3328309

1 INTRODUCTION

Demand response offers opportunities for electricity consumers to significantly reduce their electricity cost if they can regulate their power consumption to follow market requirements [15]. The aim of demand response programs is to help stabilize the power grid and to balance the demand and the supply side of the power grid. Balancing supply and demand becomes increasingly important because the integration of renewable sources of energy into the grid poses significant challenges. As renewable supplies such as solar and wind energy depend on weather conditions, they are highly volatile and intermittent [22]. Demand response offers a solution to this challenge by absorbing the volatility of energy generation with demand-side regulation [15], and power providers motivate consumers' participation by offering electricity bill reduction [23].

As a specific form of demand response programs, Regulation Service Reserves (RSRs) require participants to regulate their power consumption following a dynamically-changing power target that is updated every few seconds [23]. When participating in RSRs, users first determine their average consumption and the reserves they can provide. Users get more reduction in their electricity cost if they provide larger reserves, i.e., if they are capable of tracking a power target varying in larger range. After the average value and the reserve amount are selected, the real-time value of the target power consumption is calculated based on a signal provided by independent system operators (ISOs). This signal cannot be known in advance, but its statistical features are known. The electricity cost of a user is determined by its average power, the reserves it provides, and the tracking error which quantifies the difference between the target power and the user's actual power usage at every moment. Larger tracking error results in higher cost.

Data centers[1] play an essential role in fulfilling computational tasks in various domains, including analyzing commercial data, supporting online services, conducting scientific research, etc. However, data centers are significant power consumers. In 2014, the power consumption of data centers in US has reached 70 billion kWh, close to 2% of US electrical usage [29]. The top-1 supercomputer in 2018, the Summit system, consumes a peak power of 10 MW [32], which generates a cost of about $24,000 per day. Therefore, the electricity

[1]In this work, we define data centers broadly, including both enterprise and high performance computing data centers.

cost is a burden for data centers, and reducing the electricity cost of data centers will not only benefit existing data centers but also pave the way for future exascale computing systems.

Data centers are good candidates to participate in RSRs because they are capable of regulating their power consumption in a large range through various strategies, including job scheduling [9], server state transition [13], server power capping [28], virtual machine (VM) provisioning [6], etc. For example, to reduce the power consumption of a data center, we can postpone the execution of some jobs by scheduling, or turn some servers into sleep states (or other power-saving states). We can also apply power caps on the processors by limiting their voltage and frequency. For data centers supporting virtual environments, one can also reduce the CPU/memory resource limits of VMs [6, 12]. Previous work has proposed power regulation policies for data centers to participate in RSRs, but without guarantees on the Quality-of-Service (QoS) of jobs [5, 7]. The QoS of jobs, in this work quantified by the jobs' sojourn time in a data center, is a vital index that data center users care about. It is unacceptable for data centers to participate in RSRs at the cost of violating the QoS constraints of jobs.

In this paper, we propose a policy called *QoSG* that enables data centers to participate in RSRs while providing guarantees on the QoS of jobs. Our *QoSG* policy minimizes the electricity cost by selecting the optimal values of the average power consumption and the reserve amount. The policy handles the scheduling of a heterogeneous workload by coordinating separate groups of servers to run different types of jobs following the spirit of a Generalized Processor Sharing (GPS) policy. As each type of jobs differs in their processing time, power profile, and QoS constraint, our policy assigns different weights to each job type. These weights determine the ratio of the number of servers running each type of jobs, and the weights are also optimized to minimize electricity cost under the QoS constraints. The key in providing QoS guarantee is to determine an acceptable range for the policy parameters using a queueing theoretic result from Paschalidis et al. [3, 26], which quantifies the delay in the GPS policy.

To summarize, the contributions of this work are as follows:

- We propose a policy that enables a data center to perform demand response with theoretically-proven guarantees on the QoS of jobs running on the data center (Section 3).
- We evaluate our policy in both large-scale simulations and also with real-system experiments on a small cluster. We demonstrate that our policy reduces the electricity cost by 14-51% while providing guarantees to the QoS of jobs (Section 5).
- By running simulations at different data center and workload settings, we demonstrate that our policy is robust with respect to the size of data centers. We also show the data center and workload parameters that maximize cost savings (Section 5.2).

2 BACKGROUND ON DEMAND RESPONSE AND REGULATION SERVICE RESERVES

Since introduced in the US in 1997, power markets have been designed to encourage power generators / wholesalers to competitively participate in injecting / withdrawing power to the buses of high voltage transmission networks [24]. The generation and consumption of power should be balanced in (near) real-time, otherwise catastrophic events such as blackouts happen. To handle unpredictable disturbances that cause temporary imbalances, independent system operators (ISOs) procure a mix of reserves in advance. Reserve types vary in their dynamic properties. Among them, frequency control reserves are deployed in millisecond intervals; RSRs have few-second lag; operating reserves are in minutes-scale intervals. The value of reserves is higher for those with shorter time periods.

Starting from 2006, one of the largest US ISOs, PJM, has allowed electricity consumers to participate in reserve transactions [27]. Since then, demand side capacity reserves have started to make significant contributions in stabilizing power systems. Among the three types of reserves mentioned above, RSRs are the most attractive ones for data centers to provide, because RSRs require electricity consumers to regulate power consumption at the timescale of seconds, which is achievable for data centers equipped with various power management strategies.

To participate in RSRs, at the beginning of every hour, electricity consumers should determine their average power consumption $\bar{P}$ and the reserve R for this hour. Within this hour, the dynamically-changing power target $P_{target}(t)$ is computed by $\bar{P}$, R, and a signal $y(t)$ provided by ISOs, according to the formula:

$$P_{target}(t) = \bar{P} + y(t)R. \tag{1}$$

Assuming the actual power usage of a consumer at this moment is $P(t)$, we define the tracking error by

$$\epsilon(t) = \frac{|P(t) - P_{target}(t)|}{R}, \tag{2}$$

which quantifies the difference between the power target and the actual power usage. At the end of this hour, the tracking performance is evaluated by computing the average tracking error $\bar{\epsilon}$ over this hour. Then, the electricity cost for this hour can be estimated as

$$Cost = \left(\Pi^P \bar{P} - \Pi^R R + \Pi^\epsilon R\bar{\epsilon}\right) \times 1\text{h}, \tag{3}$$

where Π^P, Π^R, Π^ϵ are fixed monetary cost coefficients determined by the power market. As shown in Eq. (3), consumers will get higher cost reduction if they provide larger reserves by choosing a larger value of R, but they will also be penalized for large tracking error. In addition to the average error penalty, a consumer may lose his contract with ISOs if their instantaneous tracking performance is too poor to meet the tracking error constraints from ISOs. In this paper, we use the following tracking error constraint in a probabilistic form:

$$\text{Prob}[\epsilon > 0.3] < 10\%, \tag{4}$$

which states that the tracking error is only allowed to exceed a threshold of 0.3 for less than 10% of the time.

3 A POLICY WITH QOS GUARANTEES

In this section, we first elaborate on our data center model and the workload model. Then, we give an overview of our *QoSG* policy. After that, we briefly explain the Generalized Processor Sharing (GPS) policy, which is used in developing our *QoSG* policy. We also introduce and generalize a theorem proven by Paschalidis et al. [3, 26], which quantifies the delay in the GPS policy. Finally, we explain how our *QoSG* policy works at runtime in detail, as well as how our policy determines the optimal parameters that minimize the cost under QoS constraints.

3.1 Data Center and Workload Model

The electricity usage of data centers can be decomposed into three parts: servers, cooling systems, and affiliated components including the network, storage, lighting, mechanical, or security systems. As servers consume the majority of power, this paper only focuses on the regulation of server power. We target data centers with homogeneous servers, i.e., all servers are assumed to be identical regarding their power and performance characteristics; however, it is possible to extend our policy to heterogeneous clusters of servers.

We design a policy capable of handling the general situation where a data center serves a heterogeneous workload, which means the workload is a compound of various types of jobs with different processing time, power usage, and QoS constraints. In the following, we assume the types of jobs are known a prior. That is, for each type of jobs, we know their average processing time, power usage, and QoS constraint. We assume there are separate queues for different types of jobs; i.e., jobs of the same type are submitted the same queue. Inside each queue, waiting jobs are served in a First-Come-First-Served (FCFS) manner.

We group the servers in a data center according to their states. A server is either on but not running a job (called "idle") or actively running a job (called "active"). Assuming there are J types of jobs, then, generally, the data center is composed of $J + 1$ groups of servers: one idle group, and one active group for each job type. Servers switch groups from time to time. When an idle server is instructed to run a j^{th} type job, it switches from the idle group to the j^{th} active group. When a server finishes the execution of a job, the server transits back to the idle group.

We assume once jobs start executing, they cannot be interrupted, which is common in high-performance computing systems. In the following, we also assume there is no server consolidation, i.e., a server cannot simultaneously run multiple jobs. Also, we assume all jobs are single-node jobs, i.e., each job takes one server to run. (Again, it is possible to expand the policy for multi-node jobs, which we leave to future work.)

In case the job types are not known in advance, a data center could record and analyze the properties of the jobs running on it in a "data collection period" prior to participating in demand response. The jobs running in this "data collection period" can be classified into several types according to their processing time, power usage, and QoS constraint. Then, the *QoSG* policy proposed in this paper can be applied.

In this paper, we define the QoS degradation of a type-j job, Q^j, as the percentage of extra time used for processing the job compared to its minimum processing time T^j_{min}, according to this formula:

$$Q^j = \frac{T_{so} - T^j_{min}}{T^j_{min}}. \tag{5}$$

Here, the minimum processing time T^j_{min} is defined as the time for processing a type-j job without power caps and without being delayed in the queue. The sojourn time T_{so} is the time from a job's submission to the completion of its execution, including its waiting time in the queue T_{wait} and its actual processing time T_{proc}:

$$T_{so} = T_{wait} + T_{proc}. \tag{6}$$

Thus, a job processed without any delay will have a QoS degradation value of 0; and a job whose sojourn time is 50% longer than its minimum processing time will have a QoS degradation of 0.5.

In this paper, we focus on QoS constraints in the following probabilistic form:

$$\text{Prob}[Q^j \geq Q^j_{thres}] \leq \delta^j \quad (j = 1, 2, \ldots), \tag{7}$$

which states that, among all jobs of type j, the percentage of jobs whose QoS degradation is larger than a threshold Q^j_{thres} should be less than δ^j. In Section 5, we use 10% as the value of δ^j.

3.2 An Overview of our *QoSG* Policy

To enable data centers to participate in RSRs with theoretically-proven QoS guarantees, we design our *QoSG* policy following the spirit of a Generalized Processor Sharing (GPS) policy [25]. When applied to the current context, according to the GPS policy, all the active servers are partitioned into several groups. Each group of servers executes one type of jobs. The number of servers in every group is determined by a set of weights, w_j (with $\sum_{j=1}^{J} w_j = 1$).

To follow the power target in Eq. (1) at runtime, our policy determines whether to start some jobs waiting in the queues as well as what power cap to be applied on each server. To be specific, with a given power target $P_{target}(t)$, our policy first computes the total number of servers that should be actively running jobs, $n(t)$. As we apply the GPS policy, approximately $w_j n$ servers should be running type-j jobs. Consequently, $n(t)$ can be solved by $P_{target} = \sum_{j=1}^{J} n w_j p_j + (N - n)p_{idle}$. Here, N is the number of servers in the data center. p_{idle} is the power consumption of an idle server, and p_j is the power consumption for running a type-j job. For each type of job, if the current number of servers running type-j jobs is less then the requirement determined by the GPS policy, some type-j jobs waiting in the queue will start running to match the requirement.

Since job scheduling alone cannot regulate HPC system power at very high granularity, after the waiting jobs are scheduled, we apply a power cap on each server to track the power target accurately. As there may not always exist a sufficient number of jobs in the queues, which prevents the actual power consumption to reach the target, we solve this mismatch by allowing the data center to have a queue of *standby jobs* with loose QoS constraints at the time-scale of hours/days. When there is a lack of a sufficient number of jobs in the other queues, some *standby jobs* will be started to guarantee good signal tracking.

At the beginning of every hour, our policy selects the optimal $\bar{P}$ (the average power), R (the reserves), as well as the weights w_j that determine how many servers to be allocated for each type of jobs. We select these parameters by solving an optimization problem that minimizes the monetary cost in Eq. (3) under the QoS constraints of jobs. The optimization formulation includes a constraint that guarantees QoS using a theorem proven by Paschalidis et al. [3, 26]. The theorem proves that, in the GPS policy, the number of jobs with large delay decreases exponentially as the delay increases. With

Table 1: Variables Used in the Problem Formulation

Parameter	Description
R	The selected reserve value to participate in RSRs
$\bar{P}$	The selected average power to participate in RSRs
N	The total number of servers in the data center
p_{idle}	The power consumption of a server that is idle
J	The number of job types
p_j	The power consumption of a type-j job
λ_j	The average number of type-j jobs submitted to the data center per second
T_j	The processing time for each type-j job
D^j	The delay of a type-j job, i.e., the time from submission to starting
D^j_{max}	The maximal delay that the majority of type-j jobs should satisfy
Q^j_{thres}	A threshold in the QoS constraint for type-j jobs, in Eq. (7)
δ^j	A threshold in the QoS constraint for type-j jobs, in Eq. (7)
δ^j_D	A parameter calculated by δ^j in Eq. (21)
$A_j(t)$	The random variable representing the amount of work submitted to the j-th queue per second
$B(t)$	The random variable representing the total amount of service provided by the data center per second
$y(t)$	The ISO signal at time t, which is always within $[-1, 1]$
$n(t)$	The number of servers that should be active at time t, calculated by Eq. (12)
α_j	An empirically-determined coefficient in quantifying the probability of large delay, in Eq. (9)
θ^*_j	A coefficient in the exponent quantifying the probability of large delay, in Eq. (9)
w_j	The weights used in the GPS policy

this theorem, the constraints on the QoS of jobs are converted into constraints on $\bar{P}$, R, and w_j.

3.3 Generalized Processor Sharing (GPS) Policy

The Generalized Processor Sharing (GPS) policy is originally proposed to provide balanced performance in network scheduling [25]. It assumes there are several streams of messages submitted to separate queues and processed by a single communication channel. Because these queues share the same channel, the policy partitions the channel's bandwidth and each part serves one of the queues.

With a given set of weights w_j $(j = 1, 2, ...)$ for the queues satisfying $\sum_j w_j = 1$, the GPS policy states that: (1) if all the queues are non-empty, the channel bandwidth will be partitioned according to the weights w_j; (2) if some queues are empty at a certain time, the portion of bandwidth for the empty queues will be shared by the non-empty queues; i.e., the channel bandwidth will be partitioned according to the weights of the non-empty queues. An advantage of the GPS policy is that there exists a minimal service rate for each queue. That is, assuming the channel has a fixed bandwidth B, the effective bandwidth of the j-th queue is at least w_jB, and this effective bandwidth will be larger if there is bandwidth sharing due to the existence of empty queues.

3.4 Generalization of a Theorem Quantifying Delay in GPS Policy

Paschalidis et al. have proven a theorem that quantifies the delay of messages when processed by a stochastic channel following the GPS policy [3, 26]. Their theorem concerns the situation where two streams of messages are submitted to two queues respectively and processed by a single communication channel. Two queues are independent and the number of bits received by each queue in unit time follows a stochastic process. The bandwidth of the channel also varies from time to time following another stochastic process. Assuming the channel applies the GPS policy with fixed weights w_j (here $j = 1, 2$), the theorem quantifies the distribution of delay D_j, i.e., the waiting time for a certain bit of message to be processed in the queue j. They have proven that the portion of bits with large delay decreases exponentially according to

$$\text{Prob}[D_j \geq m] = \alpha_j e^{-m\theta^*_j} \quad (j = 1, 2) \tag{8}$$

as $m \longrightarrow \infty$. Here, α_j is a coefficient to be determined empirically. θ^*_j is a coefficient that can be calculated based on the distribution of the bit arrival and the distribution of the channel bandwidth. This theorem is the key to providing QoS guarantees in our paper, but first, we need to generalize it to multiple-queue scenarios. As mentioned in their paper, a rigorous generalization of their theory to multiple-queue scenarios is particularly hard. The reason is that, to apply the GPS policy in multiple-queue scenarios, we need to analyze every case where some of the queues are empty while others are not. With the increase of the number of queues J, the number of cases increases exponentially with $O(2^J)$, making the analysis unapproachable.

To make the generalization of the theory to multiple-queue scenarios feasible, we make a "decoupling" assumption in our theoretical analysis: we assume different queues are decoupled from each other, so that the bandwidth will always be partitioned according to the weights w_j no matter whether there are empty queues or not. With this assumption, we do not need to analyze the exponentially large number of cases where some queues are empty. It deserves mentioning that this decoupling assumption does not reduce the validity of our theory. That is because if we provide QoS guarantees by applying the GPS policy with this assumption, then applying the original GPS policy should also guarantee QoS because its effective bandwidth is never smaller.

Following the derivations in Ref. [3, 26], we generalize their theorem to multiple-queue scenarios under the decoupling assumption. We prove that the portion of bits with large delay decreases exponentially as $m \longrightarrow \infty$:

$$\text{Prob}[D^j \geq m] = \alpha_j e^{-m\theta^*_j}, \quad (j = 1, 2, ..., J). \tag{9}$$

The coefficients θ^*_j for the j-th queue can be calculated by

$$\theta^*_j = \sup_{\theta \geq 0,\ \Lambda_{GPS,j}(\theta) < 0} -\Lambda_B(-\theta w_j), \tag{10}$$

where the function $\Lambda_{GPS,j}(\theta)$ is defined as

$$\Lambda_{GPS,j}(\theta) = \Lambda_{A_j}(\theta) + \Lambda_B(-\theta w_j).$$

In this equation, Λ_{A_j} and Λ_B are the log moment-generating functions for the random variables $A_j(t)$, $B(t)$. Here, $A_j(t)$ represents the number of bits arriving in queue j per unit time, and $B(t)$ represents the channel bandwidth at time t. "sup" represents the supremum of the expression under the denoted condition. Appendix A describes how we derive Eq. (9) and Eq. (10).

3.5 Our *QoSG* Policy at Runtime

We first elaborate on how our policy schedules jobs and adjusts server power caps at runtime assuming the values of $\bar{P}$, R and the weights in the GPS policy w_j are already selected. The selection of these parameters will be discussed in Section 3.6.

To apply the GPS policy to the data center context, we regard the "channel bandwidth" as equivalent to the amount of service provided by the data center per second. Then, partitioning the bandwidth according to weights w_j is equivalent to partitioning the active servers in the data center into separate groups.

To let the data center's power consumption $P(t)$ follow the power target $P_{target}(t)$ in Eq. (1), at every moment, we control the total number of active servers $n(t)$ at this moment. Assume N is the total number of servers in the data center, p_{idle} is the power consumption of an idle server, and p_j is the power consumption for running a type-j job. According to the GPS policy with our decoupling assumption, there should be $n(t)w_j$ servers running type-j jobs at this moment. Then, the total power consumption of the data center is composed of the power from active servers, $\sum_{j=1}^{J} nw_jp_j$, and the power from idle servers, $(N-n)p_{idle}$. Thus, by matching the power target with the data center's power, we get

$$P_{target} = \bar{P} + y(t)R = \sum_{j=1}^{J} nw_jp_j + (N-n)p_{idle}. \quad (11)$$

We compute the number of servers that should be active at this moment as

$$n(t) = \frac{\bar{P} + y(t)R - p_{idle}N}{\sum_{j=1}^{J} w_jp_j - p_{idle}}. \quad (12)$$

Then, we determine the number of servers that should be in each group according to the weights w_j, and our policy schedules jobs to match those numbers. For example, if there are fewer servers running type-j jobs than there should be, the policy immediately starts some type-j jobs without power caps. On the other hand, if there are more servers running j-type jobs than there should be, the jobs waiting in the corresponding queue will not be processed.

However, because we assume interrupting a job in the middle is not allowed, the actual number of active servers may be larger than $n(t)$. To reduce the impact of this mismatch, our policy applies power caps on all active servers to fine-tune the power consumption. Assume the power of a type-j job can be adjusted from $p_{j,max}$ down to $p_{j,min}$ by applying power caps on servers. Let us denote the current number of active servers processing type-j jobs as n_j. Then, to track the power target better, for each job type j, our policy reduces the power cap on the corresponding servers to $p_{j,cap}$ ($p_{j,min} \le p_{j,cap} \le p_{j,max}$). That $p_{j,cap}$ is determined by letting the equation

$$\omega = \frac{p_{j,cap} - p_{j,min}}{p_{j,max} - p_{j,min}} \quad (13)$$

hold for every job type to ensure fairness. Here, $\omega \in [0,1]$ is a single number calculated by

$$P_{target} = \sum_{j=1}^{J} n_jp_{j,cap} + (N-n)p_{idle}. \quad (14)$$

On the other hand, if there is an insufficient number of jobs in the queues, the actual number of active servers will be smaller than $n(t)$, leading to a mismatch between the actual power and the power target. To handle such "lack of job" situations, we assume that there is a queue of *standby jobs* in addition to the other types of jobs. These *standby jobs* have QoS constraints at the time-scale of hours/days. Because their QoS constraints are not as tight as the other jobs whose QoS constraints are at the time-scale of minutes/seconds, these *standby jobs* will only be processed in these situations to provide extra power consumption to match the power target. In our experiments, we assume the *standby jobs* cannot be interrupted.

Amazon EC2 Spot Instances stand as a good real-life example of the concept of *standby jobs* [1]. Amazon EC2 Spot Instances offer spare computing capacity available in their cloud at steep discounts (around 70% off) compared to normal cloud instances. Jobs running on Spot instances can be interrupted by the cloud service provider with a two-minute in advance warning whenever there are fewer spare instances than required or the dynamically-changing price exceeds a user's predetermined threshold. Although unpredictable interruptions could occur, non-urgent jobs with loose QoS constraints can benefit from this service for its lower cost. Another example is the scavenger queue on the Edison and Cori systems at the National Energy Research Scientific Computing Center (NERSC) [20]. Users with no CPU-time-quota are allowed to submit their jobs to the scavenger queue. This queue has low priority and a job can be terminated by the system after it has already run for 4 hours.

3.6 Determining the optimal $\bar{P}$, R and w_j

Our policy finds the optimal values of the average power $\bar{P}$, the reserves R, and the GPS policy weights w_j by minimizing the data center's cost in Eq. (3). Obviously, we should guarantee that the maximal power target $\bar{P} + R$ and the minimal power target $\bar{P} - R$ are achievable. Therefore, we have the following constraints:

$$\bar{P} + R \le N \cdot \max_j p_j, \quad (15)$$

$$\bar{P} - R \ge N \cdot p_{idle}. \quad (16)$$

Next, we derive the condition for guaranteeing the QoS of jobs. Based on the theorem introduced in Section. 3.4, we know that, to guarantee the QoS constraint for the type-j jobs, i.e.:

$$\text{Prob}[Q^j \ge Q^j_{thres}] \le \delta^j \quad (17)$$

is equivalent to the condition:

$$\text{Prob}\left[\frac{T^j_{wait} + T^j_{proc} - T^j_{min}}{T^j_{min}} \ge Q^j_{thres}\right] \le \delta^j \quad (18)$$

$$\Leftrightarrow \quad \text{Prob}[T^j_{wait} \ge Q^j_{thres}T^j_{min}] \le \delta^j \quad (19)$$

$$\Leftrightarrow \quad \text{Prob}[D^j \ge D^j_{max}] = \alpha_j e^{-D^j_{max}\theta^*_j} \le \delta^j \quad (20)$$

$$\Leftrightarrow \quad \theta^*_j \ge \delta^j_D = -\frac{1}{D^j_{max}} \ln\left(\frac{\delta^j}{\alpha_j}\right). \quad (21)$$

Here, because the actual processing time T^j_{proc} is usually close to the minimum T^j_{min}, we assume they are equal and we transform Eq. (18) into Eq. (19). As the delay D^j in Eq. (9) refers to the waiting time T^j_{wait}, we transform Eq. (19) into Eq. (20) after defining $D^j_{max} = Q^j_{thres}T^j_{min}$. The right hand side of Eq. (20) comes from Eq. (9).

As we mentioned in Section 3.4, to get an explicit expression for θ^*_j, we need the log moment-generating functions that characterize the stochastic processes of job arrival and job execution. To begin with, we denote the amount of service provided by the data center per second as $B(t)$, which corresponds to the "channel bandwidth" in Section. 3.4. Since the servers are homogeneous in the data center, each server provides the same amount of service per second. Let us set the amount of service provided by each server per second as 1, then the total amount of service provided by the data center per second is $n(t)$, where $n(t)$ is the current number of active servers. With this definition of "amount of service", a type-j job that takes T_j seconds to process will require T_j amount of service.

For the J types of jobs submitted to J separate queues, we assume the job arrival time in each queue follows a Poisson process[2], and we denote the parameter for the Poisson process as λ_j, which represents the average number of type-j jobs arriving every second. Then, the amount of work submitted to the j-th queue per second, $A_j(t)$, follows a Poisson distribution whose log moment-generating function is

$$\Lambda_{A_j}(\theta) = \lambda_j(e^{\theta T_j} - 1). \tag{22}$$

Practically, the processing time of a job increases when its power is capped, which makes T_j no longer a fixed number. In the following theoretical part, since server power-capping only plays a fine-tuning role in our policy, we assume the power usage and processing time of jobs are fixed. That is, we take $T_j = T^j_{min}$ and $p_j = p_{j,max}$. To handle situations where the processing time of jobs deviates significantly, we could change the log moment-generating functions accordingly if we know their probability distribution.

As discussed in Section 3.5, to participate in RSRs, our policy at runtime controls the total number of active servers $n(t)$ so as to adjust the data center power to match the power target, and $n(t)$ is derived in Eq. (12) according to the ISO signal $y(t)$.

Although we cannot know the ISO signal $y(t)$ in advance, its statistical features are usually stable. For the ISO signal samples that we use in our evaluation, $y(t)$ generally follows a normal distribution[2] with an average value $\bar{y} = 0$, and a standard deviation $y_\sigma = 0.40$. Then, from Eq. (12), we see that the number of active servers $n(t)$ should also follow a normal distribution with an average value

$$n_\mu = \frac{\bar{P} - p_{idle}N}{\sum_{j=1}^{J} w_j p_j - p_{idle}},$$

and a standard deviation

$$n_\sigma = \frac{y_\sigma R}{\sum_{j=1}^{J} w_j p_j - p_{idle}}.$$

Then, for the random variable $B(t) = n(t)$ that represents the amount of service processed by the data center per second, its log moment-generating function is

$$\Lambda_B(\theta) = n_\mu \theta + \frac{1}{2} n_\sigma^2 \theta^2. \tag{23}$$

Using Eq. (22) and Eq. (23), we calculate θ^*_j that quantifies the distribution of large delay by Eq. (10). Then, Eq. (21) tells us whether a set of values for $\bar{P}$, R, w_j meets the QoS constraints or not.

To summarize, our policy selects the optimal parameters $\bar{P}$, R, w_j that minimize monetary costs under QoS constraints by solving the following optimization problem:

$$\min_{\bar{P}, R, w_j} \left(\Pi^P \bar{P} - \Pi^R R + \Pi^\epsilon R \bar{\epsilon}\right) \times 1\text{h} \tag{24}$$

$$\text{subject to} \quad \theta^*_j \geq \delta^j_D \quad (j = 1, 2, ..., J) \tag{25}$$

$$\delta^j_D = -\frac{1}{D^j_{max}} \ln\left(\frac{\delta^j}{\alpha_j}\right) \tag{26}$$

$$\theta^*_j = \sup_{\theta \geq 0,\ \Lambda_{GPS,j}(\theta) < 0} -\Lambda_B(-\theta w_j) \tag{27}$$

$$\Lambda_{GPS,j}(\theta) = \Lambda_{A_j}(\theta) + \Lambda_B(-\theta w_j) \tag{28}$$

$$\Lambda_{A_j}(\theta) = \lambda_j(e^{\theta T_j} - 1) \tag{29}$$

$$\Lambda_B(\theta) = n_\mu \theta + \frac{1}{2} n_\sigma^2 \theta^2 \tag{30}$$

$$n_\mu = \frac{\bar{P} - p_{idle}N}{\sum_{j=1}^{J} w_j p_j - p_{idle}} \tag{31}$$

$$n_\sigma = \frac{y_\sigma R}{\sum_{j=1}^{J} w_j p_j - p_{idle}} \tag{32}$$

$$\bar{P} + R \leq N \cdot \max_j p_j, \quad \bar{P} - R \geq N \cdot p_{idle} \tag{33}$$

$$\sum_{j=1}^{J} w_j = 1, \quad \bar{P}, R, w_j > 0 \tag{34}$$

After some calculations (see Appendix B), we can simplify Eqs. (25) (27)-(30) into

$$n_\sigma \leq \frac{n_\mu}{\sqrt{2\delta^j_D}} \tag{35}$$

$$w_j \geq \frac{2\delta^j_D T_j}{\left(n_\mu + \sqrt{n_\mu^2 - 2\delta^j_D n_\sigma^2}\right) \ln\left(1 + \frac{\delta^j_D}{\lambda_j}\right)} \equiv w^*_j \tag{36}$$
$$(j = 1, 2, ..., J).$$

Because increasing R creates a larger variation in the power target, QoS degradation increases with the increase of R. Therefore, when $\bar{P}$ is fixed, there exists a maximal value of R, beyond which no choice of weights w_j can meet the QoS constraints. When $\bar{P}$ and R are fixed, increasing w_j is always beneficial to type-j jobs because it allows that type of jobs to be processed with more servers. These observations intuitively explain the existence of an upper limit of n_σ and a lower limit of w_j in Eqs. (35) (36). Based on this, we design an algorithm to solve Eqs. (24~34):

(1) Start with a fixed value of $\bar{P}$.
(2) Use binary search to find the maximal R that guarantees QoS. We start with an initial value for R, and use Eq. (36) to compute the minimal weights w^*_j. If the sum of these minimal weights, $\sum_{j=1}^{J} w^*_j$, is smaller than 1, then it means this value of R is small enough so that there exists a set of weights that meet the QoS constraints of the jobs, consequently we can increase R. On the other hand, if the sum $\sum_{j=1}^{J} w^*_j$ is already larger than 1, then it means no weights exist to meet the QoS constraints, and we should reduce the value of R. After several iterations, we will arrive at the maximal value of R. This maximal R also minimizes the cost function for this fixed $\bar{P}$ because of the negative term $-\Pi^R R$ in Eq. (24). Note that the third term in Eq. (24) related to

[2]Our approach is not limited to job arrivals following a Poisson process or ISO signals being Gaussian. Instead, the distributions of job arrivals and ISO signals can be general and our approach can still be applied. For other forms of distribution, the log moment-generating functions in Eqs. (22)(23) should be calculated accordingly.

Table 2: Characteristics of benchmark applications.

Application Name	Max Processing Time (T^j_{max})	Min Processing Time (T^j_{min})	Max Power ($p_{j,max}$)	Min Power ($p_{j,min}$)
fs	100.8 s	97.6 s	142 W	137 W
sc	53.8 s	52.6 s	221 W	218 W
bt	143.0 s	108.5 s	279 W	241 W
lu	85.3 s	62.8 s	262 W	231 W
mg	161.2 s	132.6 s	309 W	261 W
sp	108.4 s	99.9 s	290 W	260 W
ft	35.2 s	24.9 s	281 W	229 W

tracking error is much smaller and also changes much slower when changing R, compared to the first two terms. With this set of $\bar{P}$, R, and w_j, we run one simulation to get the actual tracking error and use it to compute the cost in Eq. (24).

(3) Loop through different values of $\bar{P}$ within an acceptable range of $\bar{P}$ derived from Eq. (33), and repeat the second step. By comparing the cost from different $\bar{P}$, we arrive at the global minimum and get the optimal choice of $\bar{P}$, R, and w_j. In practice, we apply this algorithm to get the optimal $\bar{P}$, R at the granularity of 1%.

As we mentioned in Section 3.4, the coefficients α_j are determined empirically. Therefore, before we solve the optimization problem, we always run a simulation to determine the coefficients α_j by fitting the QoS distributions with Eq. (9).

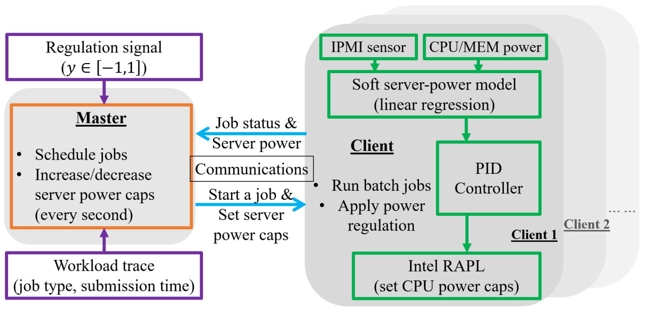

Regulation signal ($y \in [-1,1]$)
Master
Schedule jobs
Increase/decrease server power caps (every second)
Workload trace (job type, submission time)
Job status & Server power
Communications
Start a job & Set server power caps
IPMI sensor
CPU/MEM power
Soft server-power model (linear regression)
Client
Run batch jobs
Apply power regulation
PID Controller
Intel RAPL (set CPU power caps)
Client 1
Client 2
… …

Figure 1: Real-system implementation architecture.

4 EXPERIMENTAL METHODOLOGY

To evaluate our *QoSG* policy, we run simulations at various scales and conduct experiments on a real cluster composed of 12 servers.

In both simulations and real-system experiments, we use computing workloads composed of several types of jobs. These jobs are applications from NAS Parallel Benchmark (NPB) [2] and PARSEC Benchmark [8] suites, which are good representatives of real applications running on high-performance computing data centers. Table 2 summarizes the characteristics of the benchmark applications collected by running them on our servers. Among them, bt, lu, mg, sp, ft are from the NPB. fs and sc are abbreviations for applications facesim and streamcluster from the PARSEC suite.

We randomly create workload traces composed of these benchmark applications for our experiments. For each type-j job, its arrival times are generated as a Poisson process with an arrival rate λ_j. This arrival rate is determined by the data center utilization level $\eta \in [0, 1]$, which represents the average ratio of active servers in the data center. By default, we assume different types of jobs will occupy the data center equally on average, so λ_j is determined by:

$$\lambda_j T^j_{min} = \frac{\eta N}{J} \quad (j = 1, 2, \ldots, J). \tag{37}$$

To compare with our new policy, we implement two policies proposed in previous work, the *Tracking-only* policy [5] and the *EnergyQARE* policy [7], as baselines.

The *Tracking-only* policy uses job scheduling and processor power-capping to control a data center's power to follow the power target. It only focuses on target tracking, ignorant of the QoS of jobs. This policy determines $\bar{P}$ and R heuristically. It selects a $\bar{P}$ equal to the estimated average power of the data center according to its utilization level, i.e., $\bar{P} = \eta N \sum_{j=1}^{J} p_j / J + (1 - \eta) N p_{idle}$. And R is selected so that the minimal (and maximal) possible power targets, $\bar{P} - R$ (and $\bar{P} + R$), are achievable. Thus, $R = \min\{\bar{P} - N \cdot p_{idle}, N \cdot \max_j p_j - \bar{P}\}$.

EnergyQARE monitors the tracking error and jobs' QoS degradation at every moment. Based on whether the tracking error or the QoS degradation at this moment is larger, it either focuses entirely on tracking more accurately without concerning jobs' QoS, or tries to reduce QoS degradation by running more jobs without concerning the tracking error. The *EnergyQARE* policy runs simulations with each pair of $(\bar{P}, R)$ and selects the best $\bar{P}$ and R that minimize the cost under the tracking error constraint.

These two baselines assume the idle servers can transition into sleeping states to further reduce power. Since the servers in our real system do not support sleeping states, we implement the baselines without server state transition.

In the following, we describe our simulation method and our real-system implementation.

Simulation: We implement a simulator in Python. The simulator creates a cluster of servers and keeps track of the servers' power usage. For each server, we assume an idle power of $p_{idle} = 90$ W, and we assume the power usage of jobs follows the values in Table 2. When our policy applies a power cap on a server that is running a type-j job, we assume the processing time of the job increases linearly. That is, when the power cap decreases from $p_{j,max}$ to $p_{j,min}$, the processing time will linearly increase from T^j_{min} to T^j_{max}.

Real-System Implementation: The cluster used in our experiments is composed of 12 Dell PowerEdge M620 blade servers. Each server has two Intel Xeon E5-2650 v2 processors. Each server consumes an idle power of $p_{idle} = 90$ W, and its power can rise to more than 300 W when running certain applications.

For these servers, we monitor their processor and memory power using the *perf* utility [19], and we monitor the total power of a server using IPMI tool [16]. However, the IPMI tool on these servers gives a running average value at a granularity of 4 Watt, which is too coarse to serve our purpose. Therefore, we build a power model to get the server power with higher granularity. This power model takes the processor power P_{proc} and the memory power P_{mem} as input, and gives the total server power P_{server} as output, following a linear relation: $P_{server} = \phi_1 P_{proc} + \phi_2 P_{mem} + \phi_3$.

This power model inherently considers the power contributed by various components in a server, including fans, hard drives, motherboards, etc. We run each benchmark application several times and collect their processor/memory/server power readings, and fit them with the power model. For our servers, the fitting

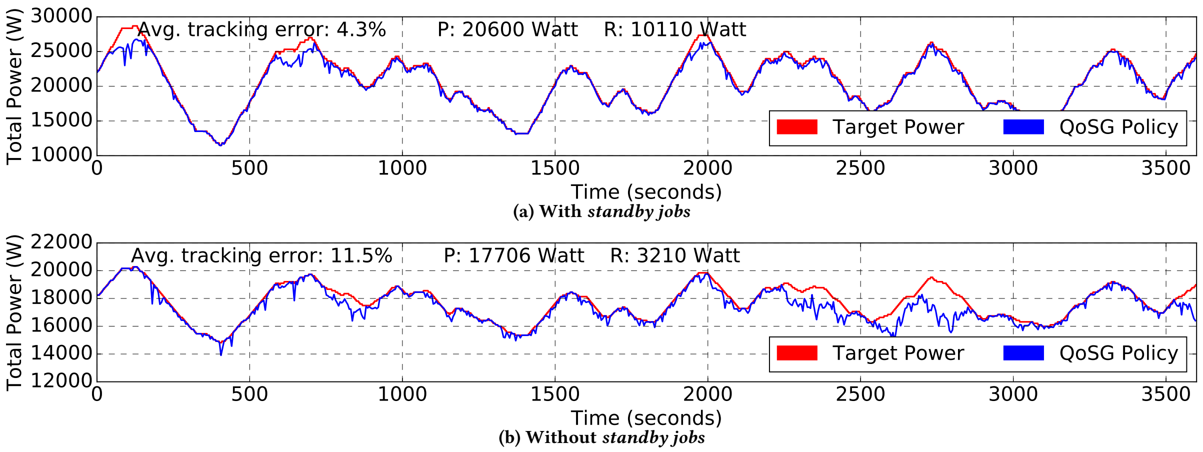


Figure 2: Simulation results for a 100-server data center participating in demand response using our *QoSG* policy. The actual power consumption (blue) of the data center follows the target power (red) closely, with a 4.3% average tracking error.

Table 3: Thresholds Q^j_{thres} in the QoS constraints of jobs.

QoS Constraint Level	fs	sc	bt	lu	mg	sp
Tight	0.65	0.60	0.55	0.45	0.40	0.35
Medium	1.3	1.2	1.1	0.9	0.8	0.7
Loose	2.6	2.4	2.2	1.8	1.6	1.4

provides the following model parameters: $\phi_1 = 1.35$, $\phi_2 = 1.32$, $\phi_3 = 53.4$ W, and this model gives a server power reading at high granularity with an error of less than 2%.

To apply a power cap on a server, i.e., to add a power limit that the server cannot exceed, we use the Intel Running Average Power Limit (RAPL) tool [11]. As RAPL can apply power caps on processors and cannot directly control the total server power, we build a PID controller to apply power caps on servers. Given a server power cap, the PID controller recurrently adjusts the processor power caps by RAPL to let the server power match the cap. On our servers, we explore different parameters for the PID controller and the choice $P = 0.525$, $I = 0$, $D = 0$ works well.

To experiment on this cluster, we let one server be a "master" node to schedule jobs and determine power caps following our *QoSG* policy. The other servers work as "client" nodes to follow the order of the master and run jobs. Every second, the master node sends a message about the scheduling decision to each client and receives a message about their status from each client using the *rabbitmq* [4] tool. The architecture of this system is shown in Fig. 1.

5 RESULTS

We evaluate our *QoSG* policy through both simulations and real-system experiments. In our default setting, we use a data center with $N = 100$ servers (when in simulations), or 12 servers (when in real-system experiments). We assume the workload arrivals follow Poisson processes that maintain an average data center utilization level of $\eta = 50\%$. When using our *QoSG* policy, by default we let one type of job, ft, to be the *standby jobs*, and we assign some type-specific QoS constraints in Table 3 for the other 6 types of jobs. By default, we use the Medium-level QoS constraints. When estimating the electricity cost using Eq. (3), we assume $\Pi^P = \Pi^R = \Pi^\epsilon = 0.1$\$/kWh. In our experiments, we use a historical trace of an ISO signal from PJM. The signal is updated every 4 seconds, and we run each experiment with a length of one hour.

5.1 Evaluation with the Default Setting

Fig. 2(a) shows the result from a one-hour simulation when using the default setting. This simulation uses the optimal values of the average power $\bar{P}$, reserves R, and weights w_j:

$$\bar{P} = 20600 \text{ W},\ R = 10111 \text{ W},\ w_{fs} = 12.8\%,\ w_{sc} = 21.2\%,$$
$$w_{bt} = 12.7\%,\ w_{lu} = 27.1\%,\ w_{mg} = 12.4\%,\ w_{sp} = 14.0\%.$$

These optimal values are derived following the method in Section 3.6. Among all weights, w_{sc} and w_{lu} are larger than the others, because these two types of jobs have tighter absolute tolerance on the sojourn time, computed by $(1 + Q^j)T^j_{min}$.

In Fig. 2(a), the red curve is the target power; the blue curve is the power consumption when applying our policy. The actual power follows the target very well, with an average tracking error of 4.3%. The electricity cost for this hour is \$3931. If not participating in RSRs, the electricity cost will be \$7292. Therefore, our policy reduces the cost by 46.1%. The cumulative distribution function (CDF) of tracking error in Fig. 4 show that our policy meets the tracking error constraint in Eq. (4). Fig. 4 also show the CDF of sp jobs' QoS degradation, which meets the QoS constraint in Eq. (7). Other types of jobs' QoS meet their constraints as well (see Appendix C Fig. 11).

Note that the use of *standby jobs* is not the main reason for the 46.1% cost reduction, because the energy consumed by *standby jobs* only occupies 14.2% of all energy consumed in this hour. When there are no *standby jobs* at all, data centers can still apply our policy, and Fig. 2(b) shows the simulation result. In this case, the cost reduction drops to 14.4%. That is because a smaller $\bar{P}$ is needed to guarantee good signal tracking, and a smaller R is also needed to reduce the power target's variation so as to guarantee job QoS.

Fig. 4 also compares our *QoSG* policy (with *standby jobs*) with the two baselines: the *Tracking-only* policy and the *EnergyQARE* policy. We compare their CDFs for the tracking error and sp jobs' QoS degradation. Because the *Tracking-only* policy only focuses

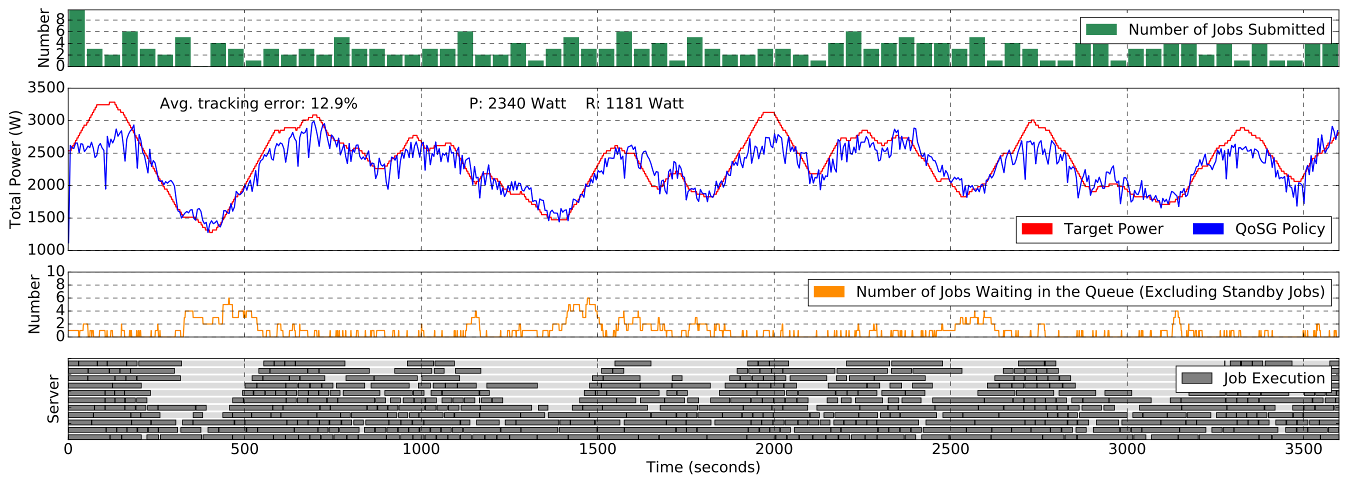


Figure 3: Experiments on a real 12-server cluster using our *QoSG* policy.

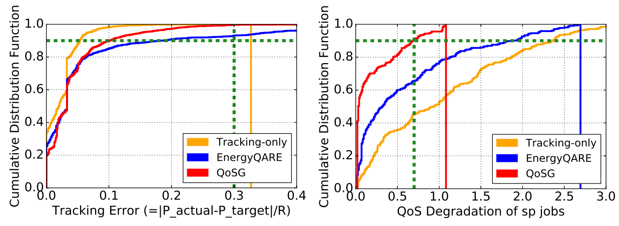


Figure 4: CDFs of QoS degradation and tracking error in simulation with the default setting.

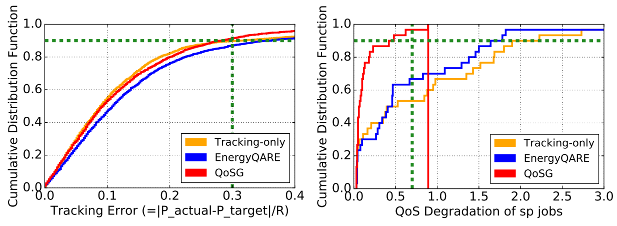


Figure 5: CDFs of QoS degradation and tracking error in real experiments with the default setting.

on tracking and does not consider job QoS, it leads to the smallest tracking error but the largest QoS degradation. The *EnergyQARE* policy balances tracking and QoS, so it reduces the QoS degradation compared to the *Tracking-only* policy. However, as their policy is designed heuristically without any theoretically-proven guarantees, QoS degradation still violates the constraints in this case. Even if we enhance the *EnergyQARE* policy by allowing it to have a queue of *standby jobs*, the QoS constrains of some job types are still violated because that policy does not assign optimally-determined weights to different queues and, thus, cannot balance different types of jobs well. The power-time curves are shown in Appendix C Fig. 13.

We evaluate our proposed policy and the two baselines on a real cluster using the default setting. The cluster has 12 servers including 1 "master" node and 11 "client" nodes. The results using our *QoSG* policy are shown in Fig. 3. The number of jobs submitted to the data center in each time interval is displayed as the green bars. The grey rectangles represent the execution (from start to end) of a job on a certain server. We see that, even with this small cluster, our *QoSG* policy enables this cluster to participate in RSRs and follow the power target well. The yellow curve shows the number of jobs (excluding *standby jobs*) waiting in the queues at every moment. When the target power drops (e.g., at $t = 400$ s), the number of active servers is reduced in order to track the target, so the number of waiting jobs increases. At a few places (e.g., at $t = 100$), the real power does not reach the target. That is because at these moments, all servers are running, so the total power cannot be further increased by starting more jobs. Since this mismatch seldom happens, our policy still meets the tracking error constraint.

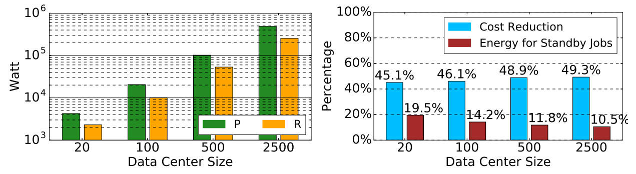


Figure 6: Results for different data center sizes.

For these real-system experiments, Fig. 5 shows the CDFs of the tracking error and the QoS degradation of sp jobs (see Appendix C Fig. 12 for other jobs). They both meet the constraints for *QoSG*, but QoS constraints are not met by using the other policies. The cost reduction using the *QoSG* policy is 42%, which is 38% when using *Tracking-only*, and 37% when using *EnergyQARE*. Compared to simulation, real-system experiments show higher tracking error because, in reality there are variations in the power usage of a job, especially at a job's beginning and completion stage. Appendix C Fig. 14 shows the power-time curves for the baselines. We also simulate the experiment with 12 servers, and the difference between the power consumption in our simulator and that in the real-system is only 5%, which indicates a reasonable accuracy for our simulator.

5.2 Influence of Different Settings

Using simulations, we evaluate the *QoSG* policy with different data center sizes. Fig. 6 shows the optimal $\bar{P}$, R values, the cost reduction, and the ratio of energy consumed by *standby jobs*. As we increase the size from 20 to 2500, the cost reduction remains above 45%, which shows our policy scales well with data center size.

Fig. 7 compares the results when using different QoS constraint levels in Table 3. Because tight QoS constraints are easier to be

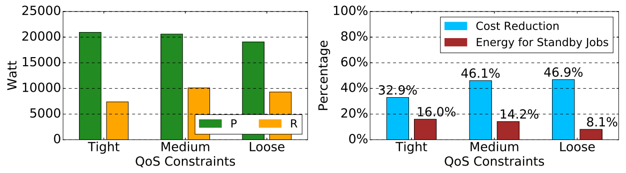


Figure 7: Results for different QoS constraint levels.

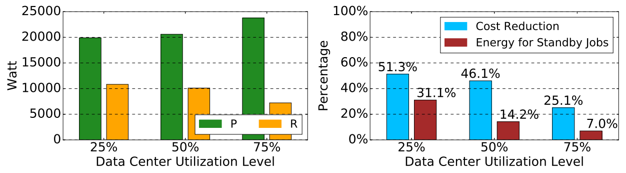


Figure 8: Results for different data center utilization levels.

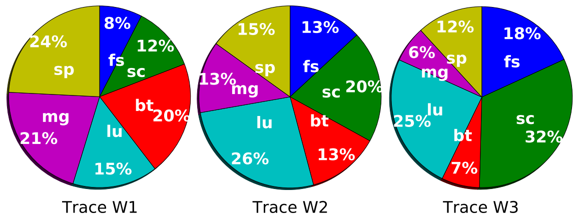


Figure 9: Optimal weights w_j in traces W1, W2, W3.

violated, more conservative selection of $\bar{P}$ and R is required. Therefore, in Fig. 7, tighter QoS constraints lead to larger $\bar{P}$, smaller cost reduction, and more *standby jobs*.

We evaluate the influence of data center utilization level by simulations with the utilization level $\eta = 25\%$, 50% (the default), or 75%. Fig. 8 shows the optimal $\bar{P}$, R, the cost reduction, and the percentage of energy consumed by *standby jobs*. Their power-time curves are in Appendix C Fig. 15. As the utilization level increases, the flexibility of data centers in power regulation decreases because more power becomes necessary in order to guarantee QoS. Therefore, we see the cost reduction drops to 25.1% when $\eta = 75\%$.

In previous experiments, we assume different types of jobs are balanced in the workload, i.e., they occupy the data center equally on average (see Eq. (37)). We evaluate the impact of this factor by comparing two unbalanced workload traces: in trace W1, the jobs with longer processing time (`bt`, `mg`, `sp`) have higher occupancy; in trace W3, the jobs with shorter processing time (`fs`, `sc`, `lu`) have higher occupancy. W2 is the default balanced trace. To be specific, the job arrival rates in these traces satisfy:

$$[\text{W1}] \quad \lambda_1 T_1 : \lambda_2 T_2 : \lambda_3 T_3 : \lambda_4 T_4 : \lambda_5 T_5 : \lambda_6 T_6 = 1:1:3:1:3:3$$

$$[\text{W2}] \quad \lambda_1 T_1 : \lambda_2 T_2 : \lambda_3 T_3 : \lambda_4 T_4 : \lambda_5 T_5 : \lambda_6 T_6 = 1:1:1:1:1:1$$

$$[\text{W3}] \quad \lambda_1 T_1 : \lambda_2 T_2 : \lambda_3 T_3 : \lambda_4 T_4 : \lambda_5 T_5 : \lambda_6 T_6 = 3:3:1:3:1:1$$

Here, indices 1 to 6 refer to `fs`, `sc`, `bt`, `lu`, `mg`, `sp`. Fig. 9 shows the optimal weights w_j when using these workload traces. We see that job types with higher occupancy tend to have higher weights. For example, from W1 to W3, the occupancy of `fs` jobs increases, so their weight increases from 8% to 18%.

We further consider the situation when there are only two types of jobs (including one type of *standby jobs*). Fig. 10 compares the simulation results for workloads of different compositions. Because the power usage of `fs` jobs is the smallest among all jobs in Table 2,

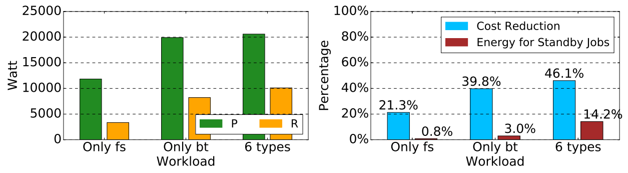


Figure 10: Comparing workloads of different compositions.

there is not much room for power regulation, so the cost reduction in this case is only 21.3%. On the other hand, for `bt` jobs whose power usage is much larger, the cost reduction becomes 39.8%.

In all the results above, we use the same one-hour ISO signals from PJM. To check the robustness against different signals, in Appendix C Fig. 16, we show simulation results (with the same parameters as Fig. 2) using 4 different one-hour samples of the PJM ISO signals. The actual power matches the target well in general. We have verified that different signals lead to similar tracking error, QoS degradation, and cost reduction.

6 RELATED WORK

There have been significant advances in recent years on integrating data centers with power markets. Various power programs including peak shaving [14], dynamic energy pricing [17], and emergency load reduction [33, 35] have been explored. As it is risky for data centers to participate in power market individually due to the uncertainty of their workloads, several works propose methods to enable multiple data centers to collaboratively participate in the power market to mitigate uncertainty [21, 34]. To enable geo-distributed data centers to participate in demand response efficiently, several works propose auction mechanisms, pricing schemes, or control frameworks targeting geo-distributed data centers [31, 36–38].

There is growing interest in data center participation in demand response programs. Some prior work explores strategies for data centers to participate in demand response by joint management of IT workloads with cooling facilities [10], renewable energy sources [18], or energy storage devices [30]. Chen et al. develop a heuristic power regulation policy [5] for data centers to participate in RSRs through job scheduling, processor power capping, and server state transition. Chen et al. also design a QoS-aware policy [7] for data centers to meet RSRs requirement considering the QoS of computing jobs. However, their work cannot provide a theoretical guarantee to data center performance in terms of job QoS. In addition, their work only evaluates their policies in simulation.

7 CONCLUSION

In this paper, we have demonstrated that, when equipped with carefully-designed power regulation policies, data centers benefit from participation in demand response programs with guarantees on the QoS of jobs. We have proposed a policy that regulates data centers' power consumption to follow a power target when providing regulation service reserves. This policy finds the optimal parameters that minimize cost under QoS constraints. Through simulations and real-system experiments, we demonstrate that our policy reduces the electricity cost of data centers by 14-51%.

REFERENCES

[1] Amazon. 2018. Amazon EC2 Spot Instances. https://aws.amazon.com/ec2/spot/?nc1=h_ls

[2] D H Bailey, E Barszcz, J T Barton, et al. 1991. The NAS Parallel Benchmarks. *Proceedings of the 1991 ACM/IEEE Conference on Supercomputing* (1991), 158–165.

[3] D. Bertsimas, I. Ch. Paschalidis, and J. N. Tsitsiklis. 1999. Large deviations analysis of the generalized processor sharing policy. *Queueing Systems* 32, 4 (01 Nov 1999), 319–349.

[4] B. Chapman et al. 2018. RabbitMQ. https://github.com/rabbitmq.

[5] H. Chen, M. C. Caramanis, and A. K. Coskun. 2014. The data center as a grid load stabilizer. *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC* i (2014), 105–112.

[6] H. Chen, C. Hankendi, M. C. Caramanis, and A. K. Coskun. 2013. Dynamic Server Power Capping for Enabling Data Center Participation in Power Markets. In *Intl. Conf. on Computer-Aided Design (ICCAD).*

[7] H. Chen, Y. Zhang, M. C. Caramanis, and A. K. Coskun. 2019. EnergyQARE: QoS-Aware Data Center Participation in Smart Grid Regulation Service Reserve Provision. *ACM Trans. Model. Perform. Eval. Comput. Syst.* 4, 1, Article 2 (Jan. 2019), 31 pages.

[8] B. Christian. 2011. *Benchmarking Modern Multiprocessors.* Ph.D. Dissertation. Princeton University.

[9] T. Cioara, I. Anghel, M. Bertoncini, I. Salomie, D. Arnone, M. Mammina, T. Velivassaki, and M. Antal. 2018. Optimized flexibility management enacting Data Centres participation in Smart Demand Response programs. *Future Generation Computer Systems* 78 (2018), 330 – 342.

[10] L. Cupelli, T. SchÃijtz, P. Jahangiri, M. Fuchs, A. Monti, and D. MÃijller. 2018. Data Center Control Strategy for Participation in Demand Response Programs. *IEEE Transactions on Industrial Informatics* 14, 11 (Nov 2018), 5087–5099.

[11] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le. 2010. RAPL: Memory power estimation and capping. *Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE International Symposium on* (2010), 189–194.

[12] G. Dhiman, G. Marchetti, and T. Rosing. 2009. vGreen: a System for Energy Efficient Computing in Virtualized Environments. In *ISLPED.* ACM, 243–248.

[13] A. Gandhi, M. Harchol-Balter, and M. A. Kozuch. 2012. Are Sleep States Effective in Data Centers?. In *IGCC.* 1–10.

[14] S. Govindan, A. Sivasubramaniam, and B. Urgaonkar. 2011. Benefits and Limitations of Tapping into Stored Energy for Datacenters. In *Proceedings of the 38th International Symposium on Computer Architecture (ISCA).* ACM, New York, NY, USA, 341–352.

[15] J. Hansen, J. Knudsen, and A. M. Annaswamy. 2014. Demand Response in Smart Grids: Participants, Challenges, and a Taxonomy. In *CDC.* 4045–4052.

[16] D. Laurie et al. 2018. An open-source tool for controlling IPMI-enabled systems. https://github.com/ipmitool/ipmitool.

[17] T. N. Le, Z. Liu, Y. Chen, and C. Bash. 2016. Joint Capacity Planning and Operational Management for Sustainable Data Centers and Demand Response. In *Proceedings of the 7th International Conference on Future Energy Systems (e-Energy '16).* 16:1–16:12.

[18] T. N. Le, Z. Liu, Y. Chen, and C. Bash. 2016. Joint Capacity Planning and Operational Management for Sustainable Data Centers and Demand Response. In *Proceedings of the Seventh International Conference on Future Energy Systems (e-Energy '16).* ACM, New York, NY, USA, Article 16, 12 pages.

[19] D. Melo and A. Carvalho. 2010. The new linux perf tools. In *Slides from Linux Kongress*, Vol. 18.

[20] National Energy Research Scientific Computing Center (NERSC). 2019. NERSC Queue Policy. https://docs.nersc.gov/jobs/policy/

[21] L. Niu and Y. Guo. 2016. Enabling Reliable Data Center Demand Response via Aggregation. In *Proceedings of the Seventh International Conference on Future Energy Systems (e-Energy '16).* ACM, New York, NY, USA, Article 22, 11 pages.

[22] C. Novoa and T. Jin. 2011. Reliability centered planning for distributed generation considering wind power volatility. *Electric Power Systems Research* 81, 8 (2011), 1654 – 1661.

[23] New York Independent System Operator (NYISO). 2018. *Ancillary Services Manual, v5.0.* Manual. NYISO. https://www.nyiso.com/manuals-tech-bulletins-user-guides

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

[25] A. K. Parekh and R. G. Gallager. 1993. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-node Case. *IEEE/ACM Trans. Netw.* 1, 3 (June 1993), 344–357.

[26] I. Ch. Paschalidis. 1999. Class-specific quality of service guarantees in multimedia communication networks. *Automatica* 35, 12 (1999), 1951 – 1968.

[27] PJM. 2005. *Integrating Demand and Response into the PJM Ancillary Service Markets.* White Paper. PJM.

[28] S. Reda, R. Cochran, and A. K. Coskun. 2012. Adaptive Power Capping for Servers with Multithreaded Workloads. *IEEE Micro* 32, 5 (Sept 2012), 64–75.

[29] A. Shehabi, S. Smith, N. Horner, I. Azevedo, R. Brown, J. Koomey, E. Masanet, D. Sartor, M. Herrlin, and W. Lintner. 2016. United States Data Center Energy Usage Report. *Lawrence Berkeley National Laboratory, Berkeley, California. LBNL-1005775 Page* 4 (2016).

[30] Y. Shi, B. Xu, B. Zhang, and Di Wang. 2016. Leveraging Energy Storage to Optimize Data Center Electricity Cost in Emerging Power Markets. In *Proceedings of the Seventh International Conference on Future Energy Systems (e-Energy '16).* ACM, New York, NY, USA, Article 18, 13 pages.

[31] Q. Sun, S. Ren, C. Wu, and Z. Li. 2016. An Online Incentive Mechanism for Emergency Demand Response in Geo-distributed Colocation Data Centers. In *Proceedings of the Seventh International Conference on Future Energy Systems (e-Energy '16).* ACM, New York, NY, USA, Article 3, 13 pages.

[32] TOP500. 2018. The Top 500 List. https://www.top500.org/lists/2018/11/

[33] N. H. Tran, C. Pham, S. Ren, Z. Han, and C. S. Hong. 2016. Coordinated Power Reduction in Multi-tenant Colocation Datacenter: An Emergency Demand Response Study. In *ICC.* 1–6.

[34] Z. Yu, Y. Guo, and M. Pan. 2017. Coalitional Datacenter Energy Cost Optimization in Electricity Markets. In *Proceedings of the Eighth International Conference on Future Energy Systems (e-Energy '17).* ACM, New York, NY, USA, 191–202.

[35] L. Zhang, S. Ren, C. Wu, and Z. Li. 2015. A Truthful Incentive Mechanism for Emergency Demand Response in Colocation Data Centers. In *2015 IEEE Conference on Computer Communications (INFOCOM).* 2632–2640.

[36] Z. Zhou, F. Liu, S. Chen, and Z. Li. 2018. A Truthful and Efficient Incentive Mechanism for Demand Response in Green Datacenters. *IEEE Transactions on Parallel and Distributed Systems* (2018), 1–1.

[37] Z. Zhou, F. Liu, and Z. Li. 2016. Bilateral Electricity Trade Between Smart Grids and Green Datacenters: Pricing Models and Performance Evaluation. *IEEE Journal on Selected Areas in Communications* 34, 12 (Dec 2016), 3993–4007.

[38] Z. Zhou, F. Liu, R. Zou, J. Liu, H. Xu, and H. Jin. 2016. Carbon-Aware Online Control of Geo-Distributed Cloud Services. *IEEE Transactions on Parallel and Distributed Systems* 27, 9 (Sep. 2016), 2506–2519.

A DERIVATION OF EQ. (9) AND EQ. (10)

We derive Eq. (9) using Theorem 7 in Ref. [26] and Theorem 7.2 in Ref. [3].

Theorem 7 in Ref. [26] proves that, in a two-class system under the GPS policy, as $m \to \infty$, the delay tail probability can be approximated by

$$\text{Prob}[D^j \geq m] \approx \alpha_j e^{-m\theta_j^*}, \quad (j = 1, 2). \tag{38}$$

Eq. (28) in Ref. [26] gives

$$\theta_1^* = \sup_{\theta \geq 0,\ \Lambda_{GPS,1}(\theta) < 0} \left[\Lambda_{A^1}(\theta) - \Lambda_{GPS,1}(\theta)\right], \tag{39}$$

where $\Lambda_{GPS,1}$ is defined by

$$\Lambda_{GPS,1} = \max\left[\Lambda^{\mathrm{I}}_{GPS,1}(\theta), \Lambda^{\mathrm{II}}_{GPS,1}(\theta)\right], \tag{40}$$

and $\Lambda^{\mathrm{I}}_{GPS,1}(\theta)$, $\Lambda^{\mathrm{II}}_{GPS,1}(\theta)$ are defined in Eqs. (22)(23) in Ref. [26]. Because $\Lambda^{\mathrm{I}}_{GPS,1}(\theta)$ corresponds to the case where the second queue is empty and the effective bandwidth of the first queue is larger than $w_1 B(t)$, we neglect this case following our decoupling assumption. This implies $\Lambda_{GPS,1} = \Lambda^{\mathrm{II}}_{GPS,1}(\theta)$, and

$$\theta_1^* = \sup_{\theta \geq 0,\ \Lambda_{GPS,1}(\theta) < 0} \left[\Lambda_{A^1}(\theta) - \Lambda^{\mathrm{II}}_{GPS,1}(\theta)\right]. \tag{41}$$

In the proof of Theorem 7.2 in Ref. [3], $\Lambda^{\mathrm{II}}_{GPS,1}(\theta)$ is given by

$$\Lambda^{\mathrm{II}}_{GPS,1}(\theta) = \sup_a \sup_{\substack{x_1 - w_1 x_3 = a \\ x_2 \geq w_2 x_3}} \left[\theta a - \Lambda^*_{A^1}(x_1) - \Lambda^*_{A^2}(x_2) - \Lambda^*_B(x_3)\right], \tag{42}$$

where $\Lambda^*(\cdot)$ is the Legendre transform of $\Lambda(\cdot)$, defined by

$$\Lambda^*(a) = \sup_\theta (\theta a - \Lambda(\theta)). \tag{43}$$

$\Lambda(\cdot)$ denote the log-moment generating functions, and $x_1(t)$, $x_2(t)$, $x_3(t)$ are the empirical rates of random process A^1, A^2, B, respectively. Because of the decoupling assumption, the influence of process A^2 can be removed from Eq. (42). Consequently, we get

$$\begin{aligned}\Lambda^{\text{II}}_{GPS,1}(\theta) &= \sup_{a}\sup_{x_1-w_1x_3=a}\left[\theta a-\Lambda^*_{A^1}(x_1)-\Lambda^*_B(x_3)\right]\\ &= \sup_{x_1}\sup_{x_3}\left[\theta x_1-\theta w_1x_3-\Lambda^*_{A^1}(x_1)-\Lambda^*_B(x_3)\right]\\ &= \sup_{x_1}\left[\theta x_1-\Lambda^*_{A^1}(x_1)+\Lambda_B(-\theta w_1)\right]\\ &= \Lambda_{A^1}(\theta)+\Lambda_B(-\theta w_1). \qquad (44)\end{aligned}$$

Combining Eq. (41) and Eq. (44), we conclude at

$$\theta_1^* = \sup_{\theta\ge 0,\ \Lambda_{GPS,1}(\theta)<0} -\Lambda_B(-\theta w_1). \qquad (45)$$

For a multi-queue system, because of our decoupling assumption, Eq. (45) holds for the first queue. Because all queues are equivalent to each other, we can directly generalize Eq. (45) by changing the subscripts and we arrive at Eq. (10).

B DERIVATION OF EQS. (35) AND (36)

To start with, we plug Eq. (30) into Eq. (27):

$$\begin{aligned}&\sup_{\theta\ge 0,\ \Lambda_{GPS,j}(\theta)<0} -\Lambda_B(-\theta w_j) &(46)\\ =&\sup_{\theta\ge 0,\ \Lambda_{GPS,j}(\theta)<0} -n_\mu(-\theta w_j)-\frac{1}{2}n_\sigma^2(-\theta w_j)^2 &(47)\\ =&\sup_{\theta\ge 0,\ \Lambda_{GPS,j}(\theta)<0} -\frac{1}{2}n_\sigma^2 w_j^2\left(\theta-\frac{n_\mu}{n_\sigma^2 w_j}\right)^2+\frac{n_\mu^2}{2n_\sigma^2} &(48)\end{aligned}$$

Whether the maximum point $\frac{n_\mu^2}{2n_\sigma^2}$ of the above quadratic function can be reached depends on whether $\theta=\frac{n_\mu}{n_\sigma^2 w_j}$ meets the conditions $\theta\ge 0,\ \Lambda_{GPS,j}(\theta)<0$.

To evaluate these conditions, we plug Eqs. (29) (30) into Eq. (28) and get

$$\Lambda_{GPS,j}(\theta)=\lambda_j(e^{\theta T_j}-1)-n_\mu\theta w_j+\frac{1}{2}n_\sigma^2\theta^2 w_j^2. \qquad (49)$$

As the second-order derivative $\Lambda''_{GPS,j}(\theta)$ is always positive, the function $\Lambda_{GPS,j}(\theta)$ is convex. As we already know 0 is a root of $\Lambda_{GPS,j}(\theta)$, there will be one positive root if

$$\begin{aligned}&\Lambda'_{GPS,j}(\theta)\big|_{\theta=0}<0 &(50)\\ \Leftrightarrow\quad & w_j>\frac{\lambda_j T_j}{n_\mu} &(51)\\ \Leftrightarrow\quad & n_\mu w_j>\lambda_j T_j, &(52)\end{aligned}$$

and there will be no positive root otherwise. Actually, Eq. (52) is the requirement that the average computing service provided should be large than the average work submitted for the j-th queue. We assume Eq. (52) is satisfied, otherwise the queue length will diverge. Then, there is a positive root for $\Lambda_{GPS,j}(\theta)$, and whether $\theta=\frac{n_\mu}{n_\sigma^2 w_j}$ meets the conditions $\Lambda_{GPS,j}(\theta)<0$ can be converted to

$$\begin{aligned}&\Lambda_{GPS,j}\left(\frac{n_\mu}{n_\sigma^2 w_j}\right)<0 &(53)\\ \Leftrightarrow\quad & \lambda_j\left(e^{\frac{n_\mu T_j}{n_\sigma^2 w_j}}-1\right)<\frac{n_\mu^2}{2n_\sigma^2} &(54)\\ \Leftrightarrow\quad & w_j>\frac{n_\mu T_j}{n_\sigma^2\ln\left(1+\frac{n_\mu^2}{2n_\sigma^2\lambda_j}\right)} &(55)\end{aligned}$$

Now, there are two cases.

Case I: If both Eqs. (51) (55) hold, from Eq. (48) we get

$$\theta_j^*=\sup_{\theta\ge 0,\ \Lambda_{GPS,j}(\theta)<0} -\Lambda_B(-\theta w_j)=\frac{n_\mu^2}{2n_\sigma^2}. \qquad (56)$$

Then, the condition in Eq. (25) is equivalent to

$$\begin{aligned}&\theta_j^*\ge\delta_D^j\quad (j=1,2,\ldots,J) &(57)\\ \Leftrightarrow\quad & n_\sigma\le\frac{n_\mu}{\sqrt{2\delta_D^j}} &(58)\end{aligned}$$

Case II: If Eq. (51) holds but Eq. (55) does not hold, i.e.,

$$w_j\le\frac{n_\mu T_j}{n_\sigma^2\ln\left(1+\frac{n_\mu^2}{2n_\sigma^2\lambda_j}\right)} \qquad (59)$$

assuming θ_0 is the positive root of $\Lambda_{GPS,j}(\theta)$, i.e.

$$\Lambda_{GPS,j}(\theta_0)=\Lambda_{A_j}(\theta_0)+\Lambda_B(-\theta_0 w_j)=0 \qquad (60)$$

then the supremum in Eq. (46) is achieved at θ_0, i.e.,

$$\theta_j^*=\sup_{\theta\ge 0,\ \Lambda_{GPS,j}(\theta)<0} -\Lambda_B(-\theta w_j)=-\Lambda_B(-\theta_0 w_j). \qquad (61)$$

In this situation, we can convert the condition in Eq. (25) by

$$\begin{aligned}&\theta_j^*\ge\delta_D^j\quad (j=1,2,\ldots,J) &(62)\\ \Leftrightarrow\quad & -\Lambda_B(-\theta_0 w_j)\ge\delta_D^j &(63)\\ \Leftrightarrow\quad & \Lambda_{A_j}(\theta_0)\ge\delta_D^j &(64)\\ \Leftrightarrow\quad & \lambda_j(e^{\theta_0 T_j}-1)\ge\delta_D^j &(65)\\ \Leftrightarrow\quad & \theta_0\ge\frac{1}{T_j}\ln\left(1+\frac{\delta_D^j}{\lambda_j}\right)\equiv\kappa &(66)\\ \Leftrightarrow\quad & \Lambda_{GPS,j}(\kappa)\le 0 &(67)\\ \Leftrightarrow\quad & \lambda_j(e^{\kappa T_j}-1)-n_\mu\kappa w_j+\frac{1}{2}n_\sigma^2\kappa^2 w_j^2\le 0 &(68)\\ \Leftrightarrow\quad & \delta_D^j-n_\mu\kappa w_j+\frac{1}{2}n_\sigma^2\kappa^2 w_j^2\le 0 &(69)\\ \Leftrightarrow\quad & \delta_D^j+\frac{1}{2}n_\sigma^2\kappa^2\left(w_j-\frac{n_\mu}{n_\sigma^2\kappa}\right)^2\le\frac{n_\mu^2}{2n_\sigma^2} &(70)\\ \Leftrightarrow\quad & \begin{cases} n_\sigma\le\dfrac{n_\mu}{\sqrt{2\delta_D^j}}\\ w_j\ge\dfrac{2\delta_D^j T_j}{\left(n_\mu+\sqrt{n_\mu^2-2\delta_D^j n_\sigma^2}\right)\ln\left(1+\frac{\delta_D^j}{\lambda_j}\right)}\equiv w_j^*\end{cases} &(71)\end{aligned}$$

In the derivation from Eq. (70) to Eq. (71), we omit another requirement

$$w_j \leq \frac{2\delta_D^j T_j}{\left(n_\mu - \sqrt{n_\mu^2 - 2\delta_D^j n_\sigma^2}\right)\ln\left(1+\frac{\delta_D^j}{\lambda_j}\right)} \tag{72}$$

because it can be proven that Eq. (72) is already implied by Eq. (59).

Combining Case I and Case II, we see that to guarantee the QoS we need

$$\begin{cases} n_\sigma \leq \frac{n_\mu}{\sqrt{2\delta_D^j}} \\ w_j > \frac{\lambda_j T_j}{n_\mu} \\ w_j \geq \min\left\{\frac{n_\mu T_j}{n_\sigma^2 \ln\left(1+\frac{n_\mu^2}{2n_\sigma^2\lambda_j}\right)}, \frac{2\delta_D^j T_j}{\left(n_\mu+\sqrt{n_\mu^2-2\delta_D^j n_\sigma^2}\right)\ln\left(1+\frac{\delta_D^j}{\lambda_j}\right)}\right\} \end{cases}$$

To further simplify the equations above, in the following, I will prove that there is always

$$\frac{n_\mu T_j}{n_\sigma^2 \ln\left(1+\frac{n_\mu^2}{2n_\sigma^2\lambda_j}\right)} > \frac{2\delta_D^j T_j}{\left(n_\mu+\sqrt{n_\mu^2-2\delta_D^j n_\sigma^2}\right)\ln\left(1+\frac{\delta_D^j}{\lambda_j}\right)} \tag{73}$$

and

$$\frac{\lambda_j T_j}{n_\mu} < \frac{2\delta_D^j T_j}{\left(n_\mu+\sqrt{n_\mu^2-2\delta_D^j n_\sigma^2}\right)\ln\left(1+\frac{\delta_D^j}{\lambda_j}\right)}. \tag{74}$$

In fact, defining $\alpha = \frac{n_\sigma^2}{n_\mu^2}$, we get

$$\frac{n_\mu T_j}{n_\sigma^2 \ln\left(1+\frac{n_\mu^2}{2n_\sigma^2\lambda_j}\right)} > \frac{2\delta_D^j T_j}{\left(n_\mu+\sqrt{n_\mu^2-2\delta_D^j n_\sigma^2}\right)\ln\left(1+\frac{\delta_D^j}{\lambda_j}\right)}$$

$$\Leftrightarrow \quad \ln\left(1+\frac{\delta_D^j}{\lambda_j}\right) > \left(1-\sqrt{1-2\alpha\delta_D^j}\right)\ln\left(1+\frac{1}{2\alpha\lambda_j}\right)$$

Define

$$H(\delta_D^j) = \ln\left(1+\frac{\delta_D^j}{\lambda_j}\right) - \left(1-\sqrt{1-2\alpha\delta_D^j}\right)\ln\left(1+\frac{1}{2\alpha\lambda_j}\right), \tag{75}$$

then we need to prove

$$H(\delta_D^j) > 0, \quad \delta_D^j \in (0, \frac{1}{2\alpha}). \tag{76}$$

First, we notice that

$$H(0) = H(\frac{1}{2\alpha}) = 0. \tag{77}$$

As

$$\frac{dH}{d\delta_D^j} = \frac{1}{\delta_D^j + \lambda_j} - \frac{\alpha}{\sqrt{1-2\alpha\delta_D^j}}\ln\left(1+\frac{1}{2\alpha\lambda_j}\right), \tag{78}$$

we have

$$\frac{dH}{d\delta_D^j}\Big|_{\delta_D^j=0} = \frac{1}{\lambda_j} - \alpha\ln\left(1+\frac{1}{2\alpha\lambda_j}\right). \tag{79}$$

Then, define $\gamma = \frac{1}{\lambda_j}$, we get

$$\frac{dH}{d\delta_D^j}\Big|_{\delta_D^j=0} = \gamma - \alpha\ln\left(1+\frac{\gamma}{2\alpha}\right) \equiv I(\gamma). \tag{80}$$

Because $I(0) = 0$ and

$$\frac{dI(\gamma)}{d\gamma} = \frac{\alpha+\gamma}{2\alpha+\gamma} > 0, \tag{81}$$

we know for $\gamma > 0$ there are

$$\frac{dH}{d\delta_D^j}\Big|_{\delta_D^j=0} = I(\gamma) > 0. \tag{82}$$

From Eq. (78), we also have

$$\lim_{\zeta\to\frac{1}{2\alpha}^-} \frac{dH}{d\delta_D^j}\Big|_{\delta_D^j=\zeta} = -\infty. \tag{83}$$

Furthermore, $H(\delta_D^j)$ is a concave function because

$$\frac{d^2H}{d(\delta_D^j)^2} = -\frac{1}{(\delta_D^j+\lambda_j)^2} - \frac{\alpha^2}{(1-2\alpha\delta_D^j)^{3/2}}\ln\left(1+\frac{1}{2\alpha\lambda_j}\right) < 0. \tag{84}$$

Combing Eqs. (77) (82) (83) (84), we get the conclusion $H(\delta_D^j) > 0$.

Similarly, there are

$$\frac{\lambda_j T_j}{n_\mu} < \frac{2\delta_D^j T_j}{\left(n_\mu+\sqrt{n_\mu^2-2\delta_D^j n_\sigma^2}\right)\ln\left(1+\frac{\delta_D^j}{\lambda_j}\right)} \tag{85}$$

$$\Leftrightarrow \quad \frac{1+\sqrt{1-2\alpha\delta_D^j}}{2}\ln(1+\delta_D^j\gamma) < \delta_D^j\gamma. \tag{86}$$

Define

$$L(\gamma) = \frac{1+\sqrt{1-2\alpha\delta_D^j}}{2}\ln(1+\delta_D^j\gamma) - \delta_D^j\gamma, \tag{87}$$

we can simply prove $L(0) = 0$ and $\frac{dL}{d\gamma} < 0$, which results in $L(\gamma) < 0$ for $\gamma > 0$.

Summarizing the discussion above, we get the final form of the conditions to guarantee QoS as

$$\begin{cases} n_\sigma \leq \frac{n_\mu}{\sqrt{2\delta_D^j}} \\ w_j \geq \frac{2\delta_D^j T_j}{\left(n_\mu+\sqrt{n_\mu^2-2\delta_D^j n_\sigma^2}\right)\ln\left(1+\frac{\delta_D^j}{\lambda_j}\right)} \end{cases} \tag{88}$$

C ADDITIONAL FIGURES

Fig. 11, Fig. 12, Fig. 13, Fig 14 are mentioned in Section 5.1. Fig. 15 and Fig. 16 are mentioned in Section 5.2.

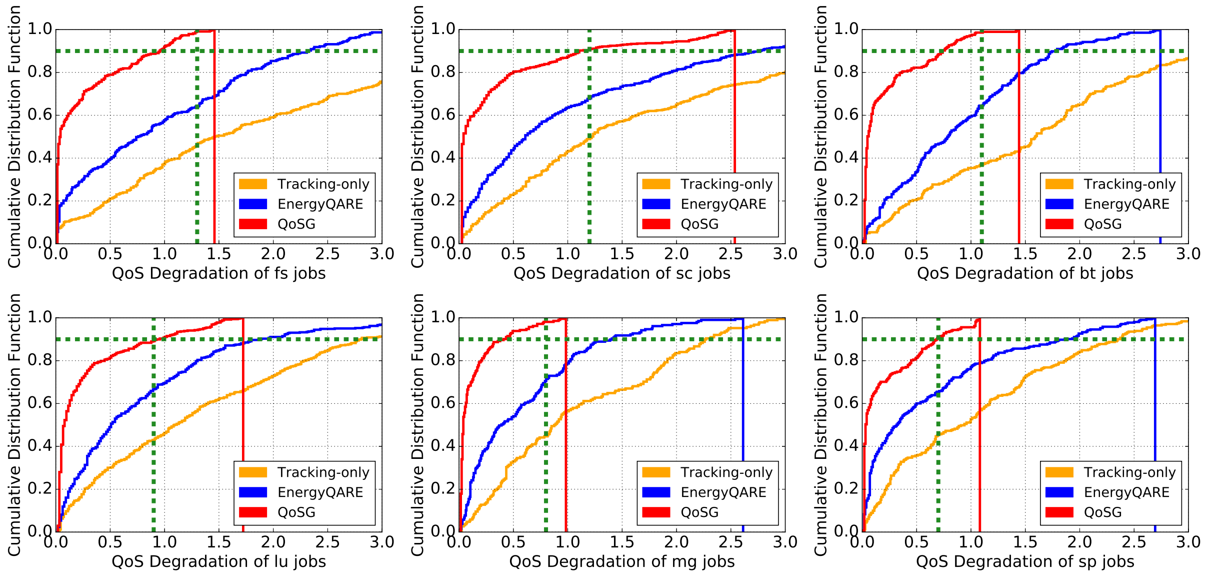


Figure 11: Cumulative distribution functions of the QoS degradation of each type of jobs in simulation. This simulation uses the default setting with 100 servers. The green dashed lines are the thresholds in the QoS constraints in Eq. (7) and Table 3. Our *QoSG* policy guarantees that all job types meet their constraints. The *Tracking-only* policy and the *EnergyQARE* policy violate the QoS constraints in this case.

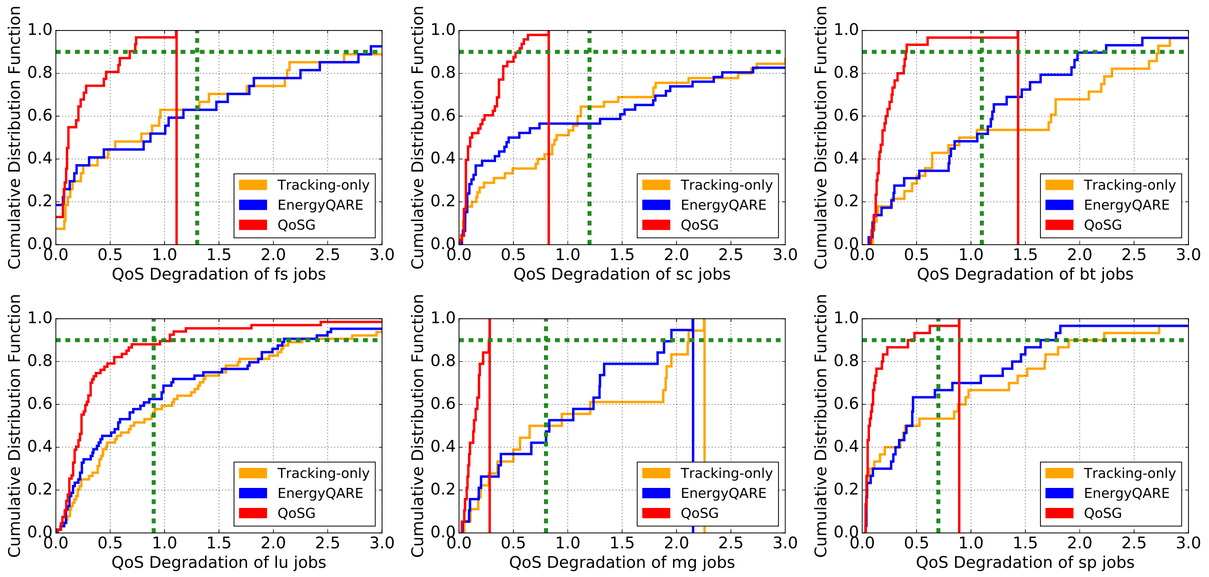
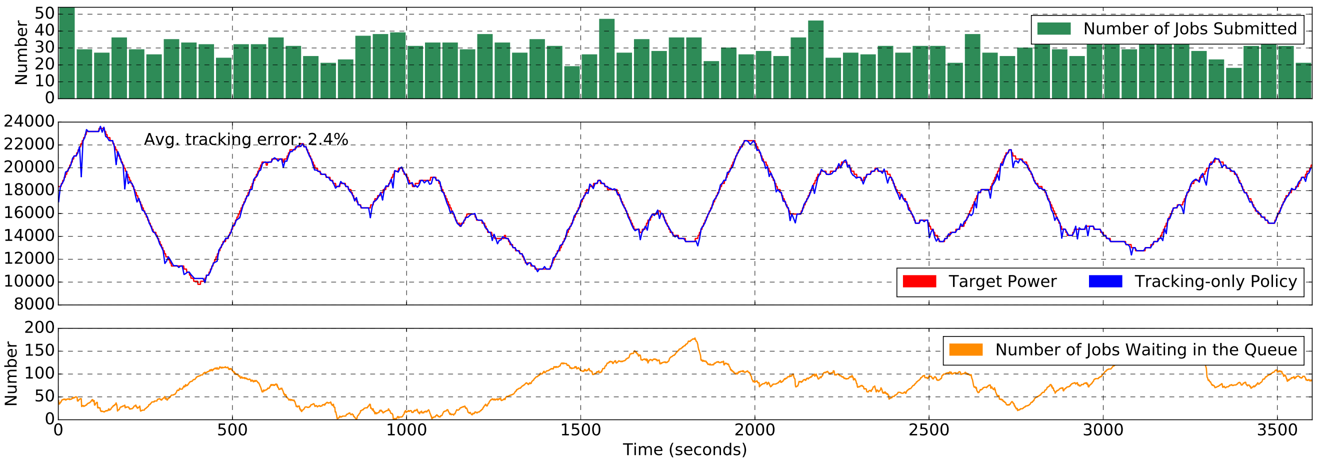
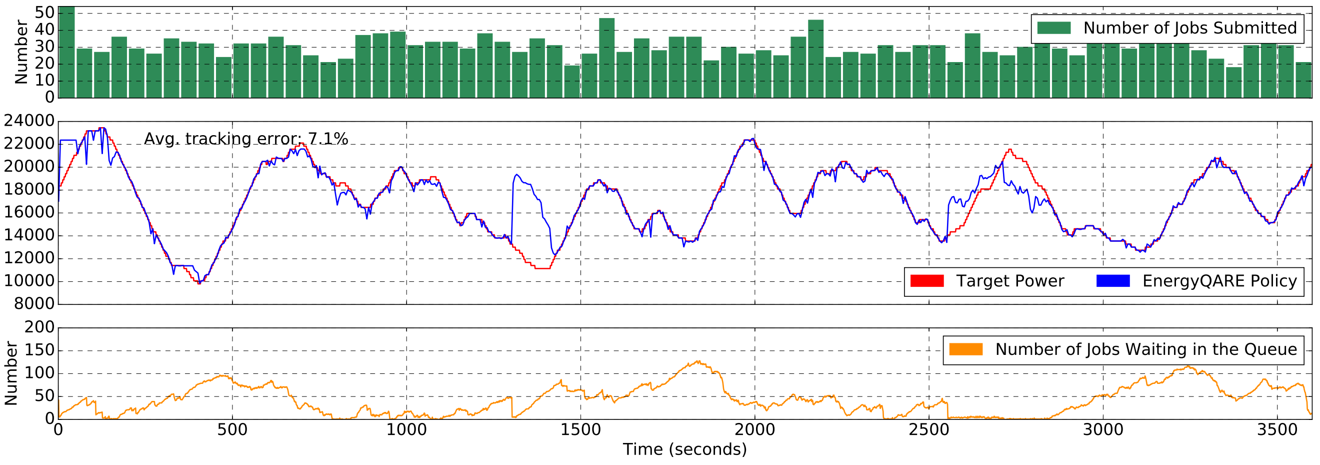


Figure 12: Cumulative distribution functions of the QoS degradation of each type of jobs in a real-system experiment with 12 servers. Our *QoSG* policy guarantees that all job types meet their constraints; meanwhile, the baseline policies cannot.

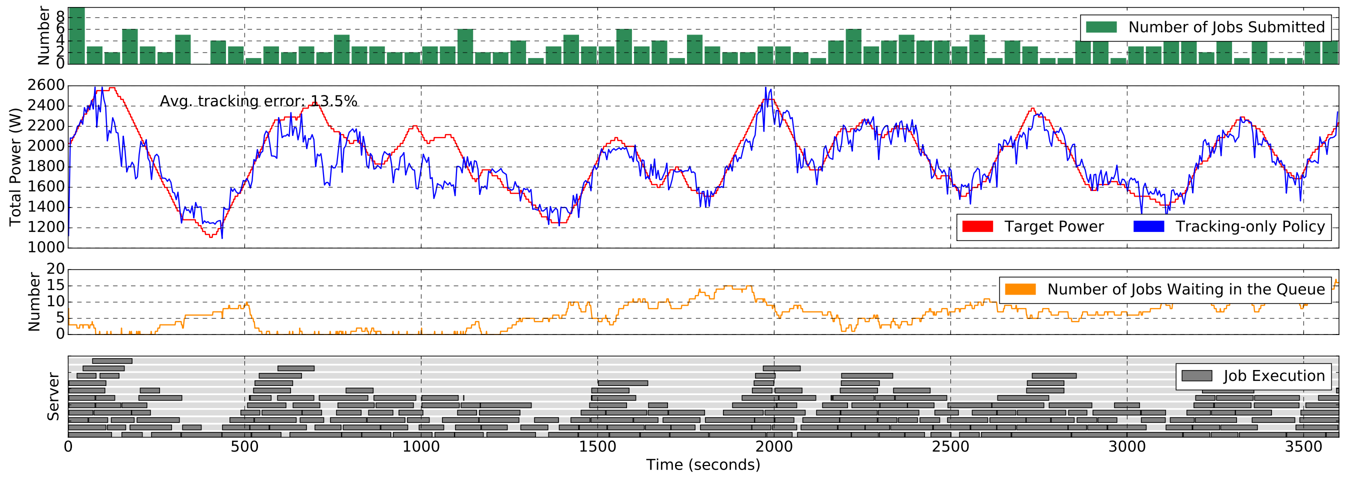


(a) The *Tracking-only* policy

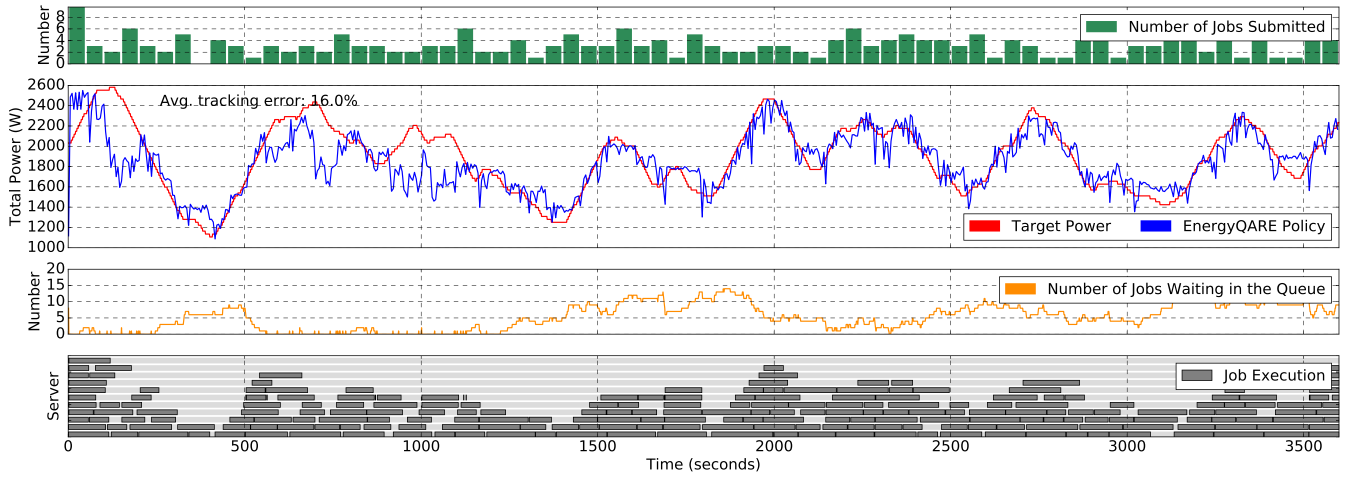


(b) The *EnergyQARE* policy

Figure 13: Power-time curves of the simulations using the two baseline policies with the default setting. The *Tracking-only* policy provides good tracking performance, with an average tracking error of 2.4%. The *EnergyQARE* policy consumes more power than the target at some places (e.g., $t = 1300$) in order to reduce the QoS degradation of the jobs. Therefore, *EnergyQARE* leads to smaller number of jobs waiting in the queues, at the cost of larger tracking error than *Tracking-only*.

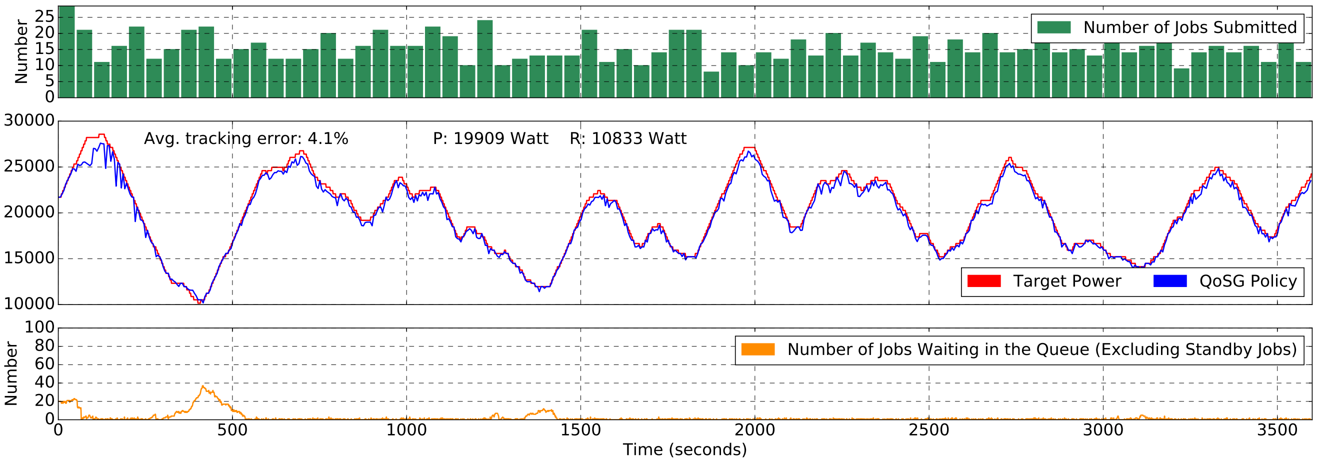


(a) The *Tracking-only* policy

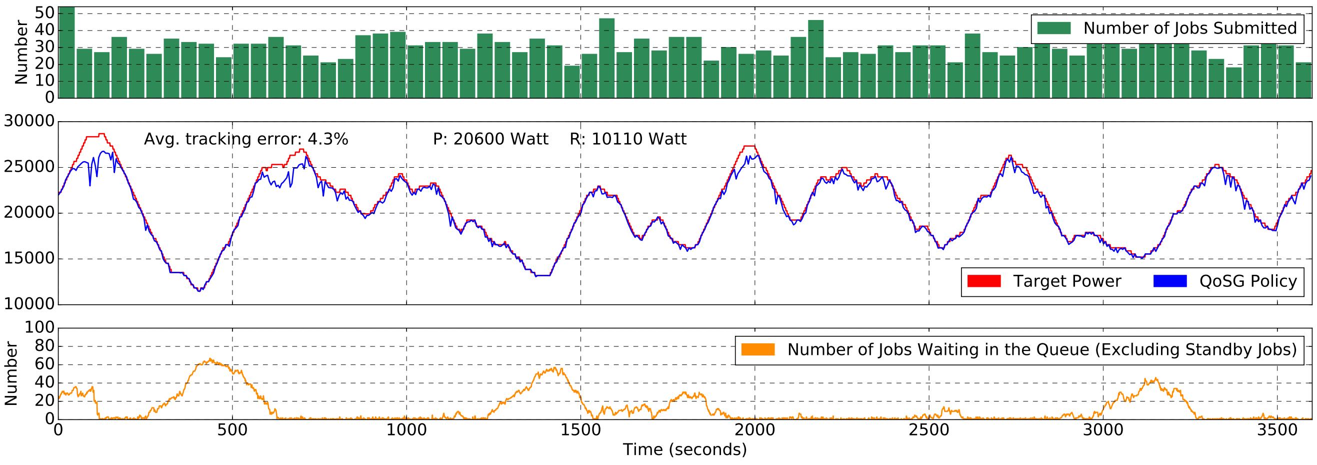


(b) The *EnergyQARE* policy

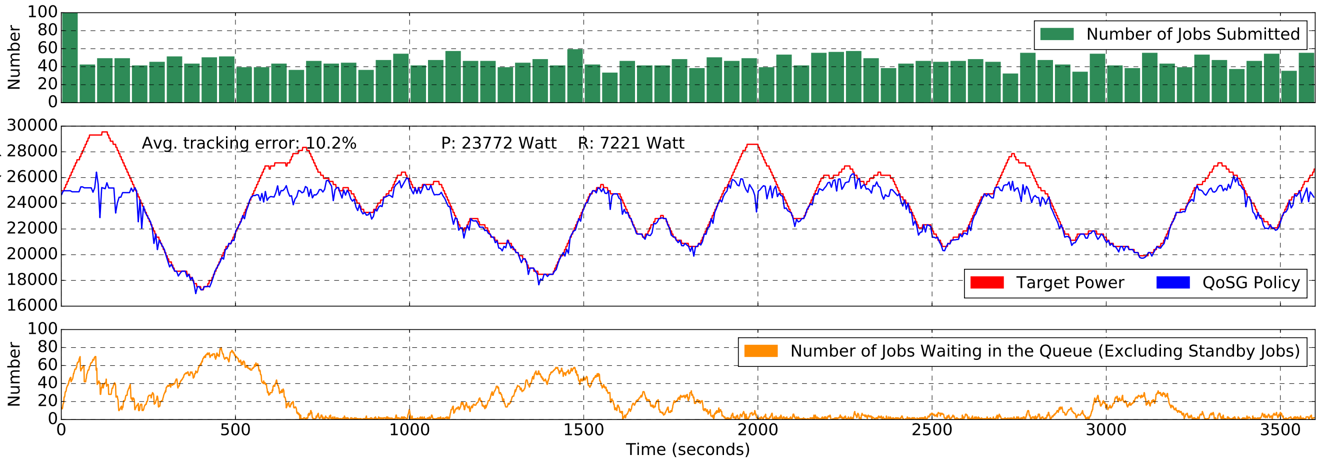
Figure 14: Power-time curves of the real-system experiments using the two baseline policies with the default setting.



(a) At 25% utilization level

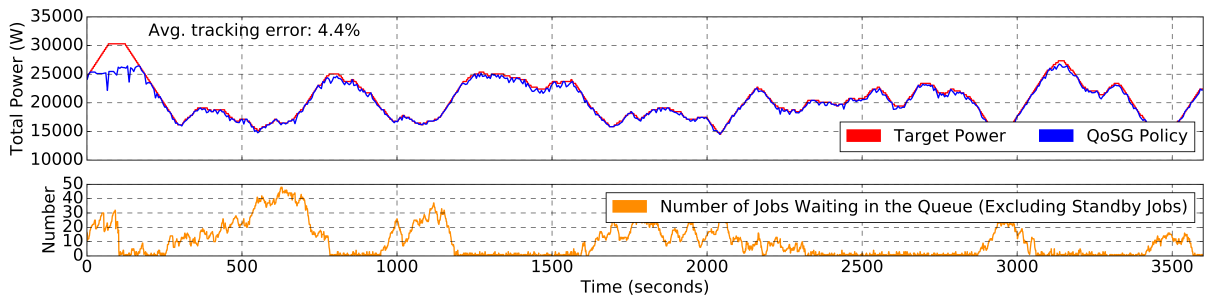


(b) At 50% utilization level

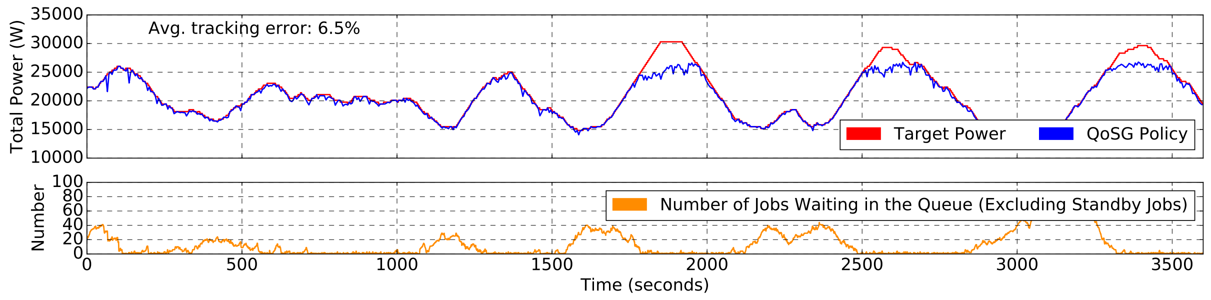


(c) At 75% utilization level

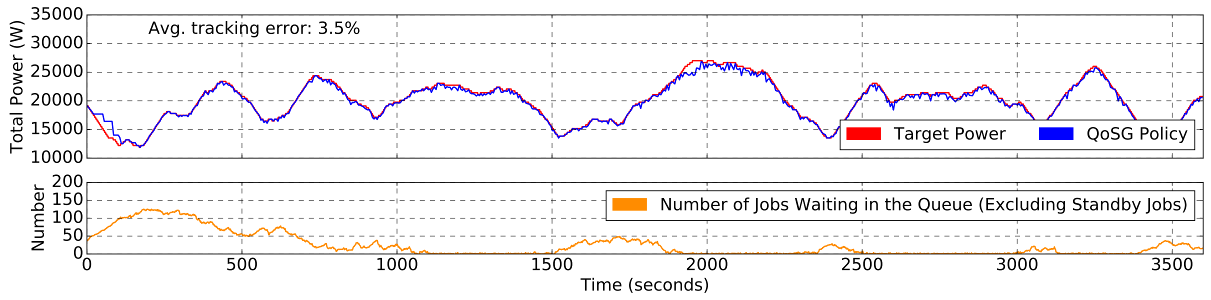
Figure 15: Power-time curves for simulations using our *QoSG* policy at different data center utilization levels. Our policy selects different optimal values for $\bar{P}$ and R according the utilization level.



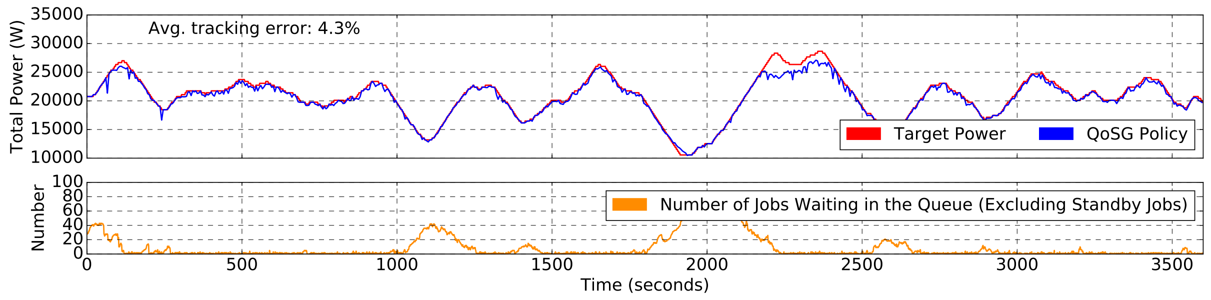
(a) ISO signal sample 1



(b) ISO signal sample 2



(c) ISO signal sample 3



(d) ISO signal sample 4

Figure 16: Results using different one-hour samples of the ISO signal. These results demonstrate that our result is robust to the behavior of ISO signal. In all these simulations, the average tracking error is less than 7%. Our *QoSG* policy meets the tracking error constraint and the job QoS constraints.