Experiments on Data Center Participation in Demand Response Programs

Yijia Zhang, Ozan Tuncer, Athanasios Tsiligkaridis, Michael Caramanis, Ioannis Ch. Paschalidis, Ayse K. Coskun
Boston University, Boston, MA 02215, USA; E-mail: {zhangyj, otuncer, atsili, mcaraman, yannisp, acoskun}@bu.edu

***Abstract*—Regulation service reserves (RSR) are among the demand response programs in emerging power markets. These programs are beneficial to the stability of the power grid. RSR requires participants to regulate their power consumption in a near-real-time manner following a signal broadcast every several seconds, and in return, the participants' electricity cost is reduced as a reward. Data centers[1] are good candidates to participate in RSR owing to their flexibility in managing power consumption and computational demand. In this paper, we demonstrate and evaluate data center participation in RSR through experiments on a 13-server cluster. We implement two regulation policies: a policy that primarily tracks the given power signal closely, and another one that prioritizes quality-of-service (QoS) of the workloads running on the data center. This paper, with policies achieving acceptable accuracy and good performance even on a small cluster, indicates a promising future where full-scale data centers participate in demand response programs.**

I. INTRODUCTION

As data centers consume large amounts of power, data center operators have a strong motivation to reduce this power consumption and, thus, reduce their electricity bill. Regulation service reserves (RSR) have recently emerged as an opportunity for data centers to participate in and reduce their electricity cost by up to 50% [1]. The main purpose of programs like RSR is to stabilize the load of electricity grid. Independent system operators (ISOs), such as PJM [2], are already offering this service to power consumers.

To participate in RSR, data centers are required to regulate their power consumption to follow a signal broadcast by the ISO. This signal, $y(t) \in [-1, 1]$, is a zero-mean random variable updated every 4 seconds. The increments of $y(t)$ in each 4-second cycle are bounded. At the beginning of every hour, data center operators bid for an average power $\bar{P}$ and a maximum power reserve R. Then, the power target for the data center to follow is calculated as $P_{target}(t) = \bar{P} + y(t)R$.

Previous work has demonstrated the benefits of data center participation in RSR, but mostly in simulation (e.g., [3]). In this paper, we implement and evaluate policies for data center demand response on a real system located in the Massachusetts Green High Performance Computing Center (MGHPCC). We compare two policies: the *Tracking-only* policy, which focuses entirely on signal tracking; and the *Quality-of-Service-aware* policy, which balances the tracking performance and the quality-of-service (QoS) of the workload. We implement the two policies on a cluster of 13 servers, and we use NAS Parallel Benchmarks (NPB) [4] as our workloads.

[1]"Data center" in this work refers broadly to a large-scale computer, including those in the cloud or in HPC systems. Our current workloads and experimental setup are more aligned with HPC or grid systems.

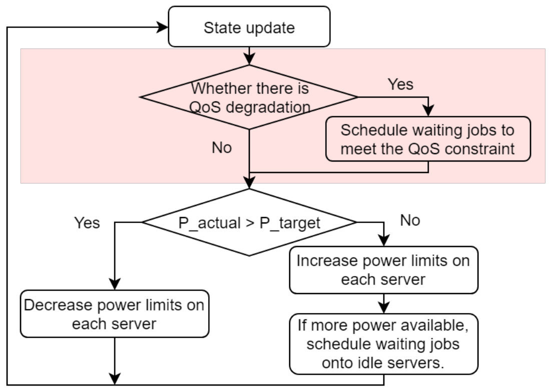

State update
Whether there is QoS degradation
Yes
No
Schedule waiting jobs to meet the QoS constraint
P_actual > P_target
Yes
No
Decrease power limits on each server
Increase power limits on each server
If more power available, schedule waiting jobs onto idle servers.

Fig. 1: Flow chart for the two policies. In the *Tracking-only* policy, the pink regions are excluded.

II. DATA CENTER POWER REGULATION POLICIES

Our policies regulate the power of the cluster through job scheduling as well as processor power capping. These policies assume *a priori* knowledge of minimum execution times and minimum/maximum power consumption of the workloads.

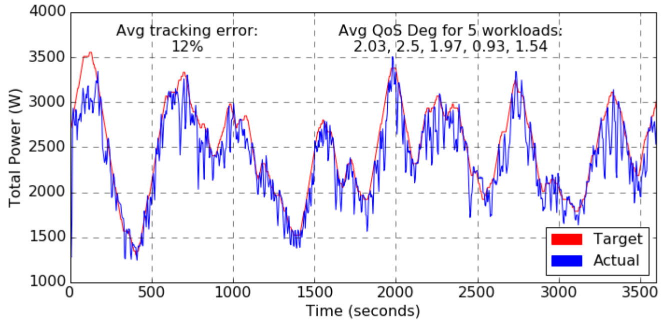
A. The Tracking-Only Policy

The goal of this policy is to let the data center follow the power target closely, and it is similar to the policy proposed in our previous work [3]. Figure 1, when its pink highlighted regions are excluded, represents the *Tracking-only* policy. In every cycle, this policy adjusts the power consumption of the servers proportionally. That is, if $P_{i,min}$ and $P_{i,max}$ represent the min/max power consumption for a server running a job of type i, we assign a power cap as $P_{i,min} + \delta \cdot (P_{i,max} - P_{i,min})$ to that server, where, the same δ ($\in [0, 1]$) is applied to all active servers (i.e., the servers running jobs).

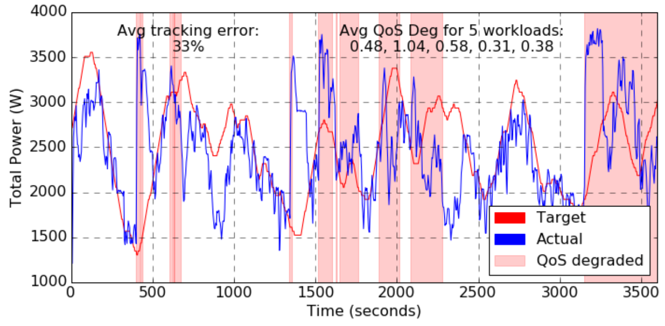
If the actual power is not large enough to meet the target even when all active servers are running at their maximal power, a number of $N = (P_{target} - P_{actual})/(P_{max} - P_{idle})$ idle servers are activated to run the waiting jobs. Here, P_{idle} is the idle server power, and P_{max} is the maximal power consumption of a server when running a job.

B. The QoS-aware Policy

As shown in Fig. 1, this policy prioritizes the QoS of the workloads by including an additional branch to handle QoS degradation. We define the QoS degradation of a job as $Q_i =$



(a) Experiments with the *Tracking-only* policy.



(b) Experiments with the *QoS-aware* policy.

Fig. 2: One-hour experiments on a 13-server cluster using two different regulation policies.

$(T_{system,i} - T_{min,i})/T_{min,i}$, where, $T_{min,i}$ is the minimum execution time of that job when executed without a power cap, and $T_{system,i}$ is defined as the job's waiting time plus actual execution time.

In each cycle, the *QoS-aware* policy determines whether QoS is degraded, i.e., checks if $Q_i > \eta_i$ for some job-specific threshold η_i. When QoS is degraded, we start a number of waiting jobs on idle servers. The number of jobs to start is determined by QoS constraints and the number of jobs waiting in the queue. Apart from this QoS adjustment, the same steps as the *Tracking-only* policy are performed.

III. EXPERIMENTAL METHODOLOGY

We experiment on 13 servers at the MGHPCC. These 13 "workers" run the computational workloads. Another "master" server executes the policies. The "master" communicates with the "workers" through *rabbitmq* [5]. In our experiments, we only take the power of the "workers" into account.

We use Dell PowerEdge M620 blade servers, each equipped with two Intel Xeon E5-2650 v2 processors. We read processor and memory power using the *perf* utility [6], and we read server power using IPMI. As IPMI on these servers only provides a running average with a 4 Watt granularity, we build a linear power model, $P_{server} = \alpha_1 P_{proc} + \alpha_2 P_{mem} + \alpha_3$, to provide a more accurate server power reading. We fit our model by linear regression using measurements from running NPB applications.

We regulate processor power using the Intel Running Average Power Limit (RAPL) [7]. We build a PID controller that enables a server to meet a power target by adjusting the RAPL.

We use five applications from NPB [4] as our jobs. To simulate realistic cases, we generate a job queue using a Poisson process. The parameters of the Poisson process are determined by $\bar{P}$ and the average utilization of the cluster. Here, $\bar{P}$ is selected as $13 \times (50\% \times P_{max} + 50\% \times P_{idle})$ based on the designated 50% utilization level. R is $\min\{13 \times P_{max} - \bar{P}, \bar{P} - 13 \times P_{idle}\}$. These are reasonable choices suggested by our previous work [3]. We set the QoS constraints of all jobs as $\eta_i = 1$. Our experiments use a real 1-hour ISO signal.

IV. RESULTS

Figure 2a shows a 1-hour experiment using the *Tracking-only* policy. The red line is the power target. The blue line is the actual total power consumption of the 13 "worker" servers. The actual power consumption follows the target well. Our tracking error meets a typical requirement from PJM: for more than 90% of the time, the tracking error $\epsilon = |P_{actual} - P_{target}|/R$ should be less than 30%. The electricity bill reduction can be estimated as $(1 - \epsilon_{\text{avg}}) \cdot R/\bar{P} = 47\%$. However, the workloads suffer from QoS degradation as large as 2.5x. In systems with loose QoS constraints, this Tracking-only policy can significantly reduce electricity costs.

Figure 2b shows results using the *QoS-aware* policy. Pink regions mark the cycles where QoS degradation exceeds the constraints, and the QoS-related branch is evoked. As we see, when QoS is degraded, there is a sudden rise in the actual power because multiple jobs start at this time to enhance QoS, which affects tracking performance. With this *QoS-aware* policy, we reduce the QoS degradation significantly, and the largest QoS degradation is 1.04x. The electricity bill reduction in this case is 36%, smaller than that of the *Tracking-only* policy due to larger tracking error.

Our next steps include reducing the tracking error in the *QoS-aware* policy, employing a broader set of applications, and providing solutions that do not require accurate profiling of applications.

ACKNOWLEDGMENT

This work has been partially funded by the Boston University College of Engineering, Dean's Catalyst Award.

REFERENCES

[1] J. Hansen, J. Knudsen, and A. M. Annaswamy, "Demand response in smart grids: Participants, challenges, and a taxonomy," in *CDC*, Dec 2014, pp. 4045–4052.

[2] PJM, "Integrating demand and response into the pjm ancillary service markets," PJM, White Paper, 2005.

[3] H. Chen, M. C. Caramanis, and A. K. Coskun, "The data center as a grid load stabilizer," *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC*, no. i, pp. 105–112, 2014.

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

[5] "Rabbitmq," https://github.com/rabbitmq.

[6] A. C. De Melo, "The new linux perf tools," in *Slides from Linux Kongress*, vol. 18, 2010.

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