Joint Resource Scheduling and Peak Power Shaving for Cloud Data Centers with Distributed Uninterruptible Power Supply

Sultan Alanazi, Mehiar Dabbagh, Bechir Hamdaoui, Mohsen Guizani†, and Nizar Zorba‡
Oregon State University, Corvallis, alanzsu,dabbaghm,hamdaoui@onid.orst.edu
† University of Idaho, Moscow, mguizani@ieee.org
‡ Qatar University, Doha, Qatar, nizarz@qu.edu.qa

***Abstract*—The grid company enforces high penalties for the peak power demands of cloud data centers. These high penalties result in high electricity bill that can be avoided by relying on the servers' Uninterruptible Power Supply (UPS) as a source of energy during peak load periods. This paper proposes a management framework that exploits the distributed UPS batteries in order to minimize the total cluster's electricity bill. Our framework consists of: i) a scheduler that accounts for both the amount of storage energy and the available resource slacks when making workload placement decision, and ii) a power distributor that decides which UPS battery should store energy and by how much in order to increase the amount of energy that can be accessible for shaving the peak during high-demand periods. Several evaluations based on real Google traces show that our proposed framework achieves significant monetary savings.**

***Index Terms*—Energy efficiency, cloud computing, resource management, peak shaving.**

I. Background

According to [1], Google continuously draws 260 Mega Watt of power in order to run their data centers. This amount of power is enough to power 200K homes and translates into an electricity bill of millions of dollars per month. Thus there is clearly a great financial incentive for IT companies to cut down their electricity bill in every possible way in order to reduce their operational expenses and increase their profits.

A cloud data center is normally divided into multiple clusters where each cluster is located in a certain geographical area and is designated a resource manager that orchestrates operating a fleet of thousands of servers. The resource manager receives Virtual Machine (VM) requests from clients each requesting a certain amount of computing resources (e.g. CPU). The resource manager decides which server in the cluster should provide the requested resources for each VM request. Clients normally run some computing jobs on the requested VMs and release the VMs once their jobs complete.

The bill that a cloud data center receives from the grid at the end of the billing cycle (e.g. month) is normally made up of two major components [2]: i) *Energy Charge:* which is proportional to the amount of consumed energy, measured in KWH, within the entire cycle. ii) *Peak Charge*[1]*:* which is proportional to the maximum power, measured in Kilo Watt (KW), drawn within the cycle. The maximum power is usually calculated by first dividing the billing cycle into slots each of 15-minute length, and then measuring the average demanded power for each slot separately. The Peak Charge is then calculated based on the slot with the maximal average demanded power.

Numerous cluster management techniques were proposed to minimize the Energy Charge of the electricity bill [3–7]. The most common approach is to consolidate the VM requests on as few ON servers as possible which allows switching the redundant servers to sleep to save energy [4, 5]. The Best-Fit (BF) heuristic is the most popular VM placement heuristic that tries to achieve this objective [6, 7]. While the BF makes significant Energy Charge reduction compared to a random VM placement strategy, it completely ignores minimizing the Peak Charge which contributes to more than 40% of the electricity bill [8].

There have been few approaches, referred to by *Peak Shaving* techniques, that are proposed to minimize the Peak Charge of the electricity bill. One of the main Peak Shaving approaches that does not cause service degradation is to store energy in batteries during low demand periods so that this stored energy can be used later to (partially) power the cloud data center during high power demand periods. This results in reducing the power drawn from the grid during high power demands, thereby resulting in minimizing the Peak Charge.

Two reasons make energy-storage peak shaving techniques practically applicable in cloud data centers. First, data centers are already equipped with controllable Uninterruptible Power Supply (UPS) batteries for fault-tolerance [9]. Second, the amount of energy that needs to be stored in UPS batteries for fault tolerance is very small compared to the energy storage capacity of those batteries [10]. This is true since during power outages, batteries need to power the data center for only a short duration (around a minute) until the diesel generator starts working. Their remaining capacity can thus be used to store energy for peak shaving purposes while always preserving a small amount of energy to power the data center during the short transition period in case a power outage occurs.

UPS batteries in data centers have two main power distribution topologies: i) *Centralized Topology:* where a large room full of batteries is used to provide power for the whole cluster. Charging those batteries requires converting the grid power

[1]Peak Charge is also called Demand Charge.

from AC to DC whereas discharging the stored energy to the cluster requires converting the power back from DC to AC. The discharged power is then fed to servers where the power supply unit (PSU) of each server converts the AC power to DC to be used by the computing components. ii) *Distributed Topology:* where each server in the cluster is supplied with an independent small UPS battery that is placed internally in the server. The server's PSU converts the AC grid power into DC and stores some amount of energy in the internal battery to be used directly by the server's components when needed.

The distributed topology was more widely adapted by Google and Facebook [11] than the centralized one as it does not suffer from the single-point-of-failure problem, allows the energy storage capacity to grow automatically when adding new servers and reduces the conversion losses by eliminating two redundant conversion stages.

However, the main disadvantage of the distributed topology is the fact that it limits the amount of energy that can be used for peak shaving as the energy stored in each battery can supply power only to its dedicated server. More specifically, while idle or lightly utilized servers in the cluster might have a good amount of stored energy in their batteries, this energy may not be fully accessible for peak shaving as the power demands of those servers is too low. On the other hand, other active servers may have large power demands but not enough energy stored in their batteries to provide the demanded power. This creates challenges for deciding how to assign VMs to servers in the cluster and how to decide how to distribute the stored energy among the distributed UPS batteries.

In this paper, we propose a resource management framework for a cloud cluster with distributive UPS topology. Our framework places the submitted VM requests in a way that reduces both the number of ON servers needed to host the VMs and the amount of stored energy that is inaccessible for peak shaving, which leads into significant reductions in both the Energy Charge and the Peak Charge of the electricity bill when compared to the traditional BF placement heuristic that completely ignores the Peak Charge. To further reduce the amount of inaccessible stored energy, our framework adapts a greedy power distribution strategy that decides based on the power demands of the servers which distributed UPS battery needs to charge (discharge) power and by how much. To summarize, our main contributions are the following. We:

- Propose a placement strategy that reduces both the number of ON servers and the amount of inaccessible stored energy for clusters with distributed UPS batteries.
- Propose a power distribution strategy that decides which UPS batteries should charge to/discharge from while minimizing the amount of inaccessible locked energy.
- Show that a good portion of the cluster's total electricity bill can be reduced by our techniques when compared to existing approaches.

The remainder is organized as follows. Section II introduces our notations. Section III explains our proposed framework. Section IV evaluates our framework based on real Google traces. Finally Section V concludes and provides directions for future work.

II. Notations

We consider a cloud cluster with a distributed UPS topology where each server is dedicated a UPS battery that can supply power only to the server it is attached to. Each server's battery has a maximal energy storage capacity, E_{max}, and a maximal charging and discharging rate, C_{max}. In order to charge/discharge an amount of power, P, some percentage of this power gets lost due to conversion losses. Also, a certain percentage of the stored energy in the battery gets lost over time due to leakage losses.

We consider a time-slotted billing cycle where the billing cycle is divided into n time slots where each slot has a duration of τ minutes. Let $\mathbb{P}$ be the set of all servers, and $\mathbb{P}_{on}$ and $\mathbb{P}_{off}$ be the set of all ON and OFF servers in the cluster ($\mathbb{P} = \mathbb{P}_{on} \cup \mathbb{P}_{off}$). To simplify our notations, the index i will be used to refer to one of the billing cycle's slots, whereas the index j will be used to refer to one of the cluster's servers. The following notations are used throughout:

- $d_{i,j}$ is the power demand for server j during the i^{th} slot. This basically represents the aggregate power demands of all the VMs hosted on server j at time slot i.
- D_i is the power demand of the whole cluster during the i^{th} slot which can be calculated as: $D_i = \sum_{j \in \mathbb{P}} d_{i,j}$.
- $c^-_{i,j}$ is the power that the battery attached to the j^{th} server discharges during the i^{th} slot in order to meet partially or fully the server's power demands.
- C^-_i is the power that our framework decides to discharge from all the batteries in the cluster during the i^{th} slot.
- $c^+_{i,j}$ is the power drawn from the grid to be stored in the battery attached to the j^{th} server during the i^{th} slot.
- C^+_i is the amount of power that our framework draws from the grid and stores (charges) in all the batteries in the cluster during the i^{th} slot.
- $e_{i,j}$ is the amount of energy stored in the battery attached to server j at the beginning of the i^{th} slot.
- $r_{i,j}$ is the amount of energy that the battery attached to server j can store at the beginning of the i^{th} slot. It can be calculated as $r_{i,j} = E_{max} - e_{i,j}$, where E_{max} is the battery's capacity.

III. The Proposed Framework

As illustrated in Fig. 1, our proposed framework has a two-level control structure. First, a scheduler, residing at the top of the framework, that controls where to place new VMs that arrive to the DC. Second, a UPS controller that controls when to charge/discharge the batteries, and how much energy should each UPS battery charge/discharge. We provide next a detailed description of our framework's components:

A. Scheduler

The scheduler follows a **S**lack and **B**attery **A**ware placement strategy which is referred to as (**SBA**) throughout the paper. The SBA strategy basically decides which server a new VM

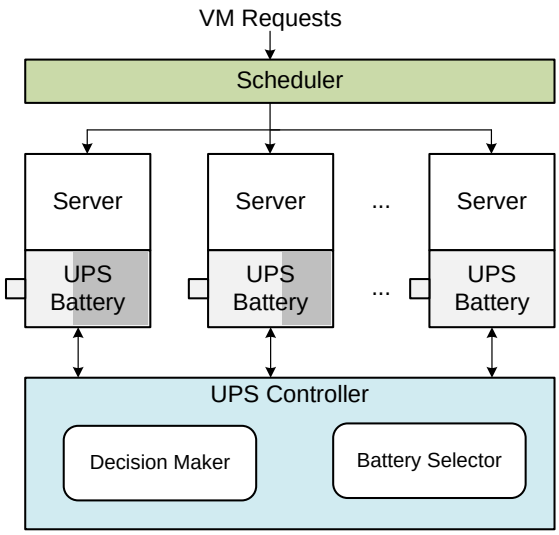

Fig. 1: Proposed Framework.

request should be assigned to based on: i) the power state of the servers (ON/OFF) within the cluster, ii) the resource utilization and capacity of the servers in the cluster, and iii) the amount of energy stored on the servers' batteries. These assignment decisions are made with two objectives in mind: a) minimizing the number of ON servers and b) maximizing the amount of accessible stored energy that can be used for peak shaving. In order to achieve these two objectives, SBA operates as follows:

- SBA places a VM request on an OFF server only if no other ON server in the cluster can fit the VM request. The intuition here is to consolidate the VM requests on fewer ON servers in order to minimize the consumed energy by turning OFF as many redundant servers as possible.

- If the VM must be placed on an OFF server, SBA prefers servers with the largest CPU capacity and with the largest UPS Stored Energy. The intuition behind this preference is the following. First, servers with larger CPU capacity are preferred as they can fit larger VMs in the future without requiring to turn ON extra OFF servers. Second, servers with larger stored energy are preferred as the larger the energy stored, the higher the accessible power that can be used to shave the peak power demands in future. Now in order to consider both the capacity of the servers and their amount of stored energy, SBA calculates a combining score for each OFF server that can fit the VM request and then picks the OFF server with the largest score. The score $S(j)$ for the OFF server j is calculated as[2]:

$$S(j) = \alpha \times S^{cpu}_{Cap}(j) + (1-\alpha) \times S_{UPS}(j)$$

where $S^{cpu}_{Cap}(j)$ and $S_{UPS}(j)$ are respectively the CPU capacity score and the UPS stored Energy score and where α is a tunnable weight that lies within the range [0,1].

[2]In our formulation we considered only a single resource (CPU). However, our framework can be easily extended to handle multiple resources (e.g. Memory and Hard Disk) by basically introducing a weighted utilization score for each one of those resources.

The CPU Capacity score $S^{cpu}_{Cap}(j)$ is calculated as follows:

$$S^{cpu}_{Cap}(j) = C^{cpu}_{j} / C^{cpu}_{max}$$

where C^{cpu}_{j} is the CPU capacity of the j^{th} OFF server and C^{cpu}_{max} is the maximum CPU capacity among all OFF servers. The UPS stored energy score is calculated as:

$$S_{UPS}(j) = E_j / E_{max} \quad (1)$$

where E_j is the energy stored in the battery attached to the j^{th} OFF server and E_{max} is the maximum amount of energy that is currently stored in the battery of any OFF servers within the cluster.

- If multiple ON servers can provide the resource demands for the submitted VM request, then SBA prefers the ON server with the largest CPU utilization and the largest amount of energy stored in the server's UPS. The intuition is as follows. It is better to place the VM on an ON server with high utilization so that larger slacks are left on the remaining servers that have low utilization. This saves energy as it allows the cluster to host VMs with larger CPU demands in the future without the need of switching extra servers from OFF to ON. On the other hand, servers with largest stored energy are preferred as they are the best candidates for future peak shaving. In order to select the server with both larger capacity and larger amount of stored energy, SBA calculates a score for each one of the ON servers that can provide the VM's requested computing resources and picks the server with the highest score to host the submitted VM request. For an ON server j that can meet the VM's demand, the score $S(j)$ is calculated as:

$$S(j) = \alpha \times S^{cpu}_{Util}(j) + (1-\alpha) \times S_{UPS}(j)$$

where $S^{cpu}_{Util}(j)$ is the CPU utilization score and $S_{UPS}(j)$ is again the UPS stored energy score. α is a tunnable weight that lies within the range [0,1].

$S_{UPS}(j)$ is calculated as described in Equation (1), and $S^{cpu}_{Util}(j)$ is calculated as $S^{cpu}_{Util}(j) = U^{cpu}_{j} / U^{cpu}_{max}$, where U^{cpu}_{j} is the CPU utilization for server j and U^{cpu}_{max} is the maximum CPU utilization among all the servers in the cluster. This score basically gives higher preference for the ON server with the higher CPU utilization.

B. UPS Controller

This module manages the UPS batteries that are attached to the servers in the cluster and consists of two sub-modules:

Algorithm 1 Decision Maker(D_i, T)

```
if D_i > T then
    C^- ← D_i − T
    SelectDischargeBattery(C^-)
else
    C^+ ← T − D_i
    SelectChargeBattery(C^+)
end if
```

1) Decision Maker: This sub-module decides when to charge/discharge batteries and by how much and is launched

at the beginning of each time slot i. The Decision Maker is illustrated as a pseudo code (Algorithm 1) and takes as input the DC's power demand, D_i, at the i^{th} slot, and a predefined threshold T. The Decision Maker compares the DC's power demand to the threshold T, and if the demanded power is below the threshold, then the difference is charged into the DC's batteries (Line 2 and 3). Otherwise (if the demanded power is above the threshold), the decision maker tries to discharge the difference from the DC's batteries (Line 5 and 6). The intuition of the Decision Maker algorithm is to charge batteries during low demand periods (i.e., periods during which the DC's power demand is below the threshold), and use this stored energy latter to power partially or fully the DC during high demand periods (i.e., periods during which the DC's power demand is above the threshold), thereby reducing the peak charge and hence minimizing DC's electricity bills.

2) Battery Selector: This sub-module decides which batteries should be charged/discharged (Line 3 and Line 6 of Algorithm 1). We explain next the discharging and charging policies that this sub-module follows:

Discharging Policy: as illustrated in Algorithm 2, the Battery Selector orders the ON servers in an increasing order of their CPU utilization (Line 1) and iterates over the ordered list trying to discharge energy from each server. There is a constraint on the amount of energy that the server's battery can discharge as it is limited by the server's power demand, the current amount of stored energy, the discharge rate and amount of power that the Decision Maker requested to discharge (Line 3). The intuition behind preferring to discharge energy from servers with lowest CPU utilization as they are more likely to become vacant in the future as they hold less workload. Once a server becomes vacant it is switched off to save energy and thus the amount of energy stored on this server becomes inaccessible for peak shaving (this energy is referred to by locked-in energy). Thus in short, the battery selector follows a greedy discharging strategy that aims at minimizing the amount of inaccessible (locked-in) energy.

Charging Policy: as illustrated in Algorithm 3, the Battery Selector charges batteries attached to ON servers with high CPU utilization first as they are less likely to be switched off soon since they have a high workload which reduces the amount of inaccessible stored energy in future. If all batteries attached to ON servers get charged and the power consumption still bellow threshold then the algorithm will start charging batteries attached to servers that are OFF where higher preference is given to the OFF servers with high capacity. The idea behind charging OFF server's is to prepare them for further discharge when they turned ON. In line 5. the battery charge is limited by the amount of energy that the battery can store (as no battery can store more than its capacity), the server's maximal charging rate and the amount of power requested to be charged by the Decision Maker.

Algorithm 2 $SelectDischargeBattery(C^-)$

1: Sort $\mathbb{P}_{on}$ servers in increasing order of their utilization
2: **for each** server j in $\mathbb{P}_{on}$ **do**
3: $c^-_{i,j} \leftarrow \min(d_{i,j},\, e_{i,j}/\tau,\, C_{max},\, C^-)$
4: Discharge $c^-_{i,j}$
5: $C^- \leftarrow C^- - c^-_{i,j}$
6: **if** $C^- == 0$ **then**
7: break
8: **end if**
9: **end for**

Algorithm 3 $SelectChargeBattery(C^+)$

1: Sort $\mathbb{P}_{on}$ servers in decreasing order of their utilization
2: Sort $\mathbb{P}_{off}$ servers in decreasing order of their capacity
3: $\mathbb{P} \leftarrow \mathbb{P}_{on} \cup \mathbb{P}_{off}$
4: **for each** server j in $\mathbb{P}$ **do**
5: $c^+_{i,j} \leftarrow \min(r_{i,j}/\tau,\, C_{max},\, C^+)$
6: Charge $c^+_{i,j}$
7: $C^+ \leftarrow C^+ - c^+_{i,j}$
8: **if** $C^+ == 0$ **then**
9: break
10: **end if**
11: **end for**

IV. Performance Evaluation

In this section, we evaluate our proposed framework based on real Google cluster traces [12]. We limited our evaluations to only a part of the traces spanning a five-day period due to the immense size of the traces (>39GB). We consider Lead Acid batteries whose specifications are summarized in Table I.

TABLE I: LA Battery Specs

Rated Capacity	30Ah
Charge losses	8%
Discharge losses	2%
Leakage per day	0.3%

We compare our proposed framework against the following cluster management schemes:

- **Best Fit (BF) Placement**: places each submitted VM request on the ON server with the least CPU slack that can fit the submitted VM request. If no ON server can fit the VM request, then the VM is placed on the server with the largest capacity.
- **Random Placement**: places a new VM request on a random ON server that can fit the VM. If no ON servers can fit the VM, then a random OFF server is picked to host the submitted VM.

Both of these schemes use our proposed Decision Maker (Algorithm 1) to decide when and how much energy need to be charged/discharged. These schemes, however, use a random Battery Selector which picks a random server's battery among

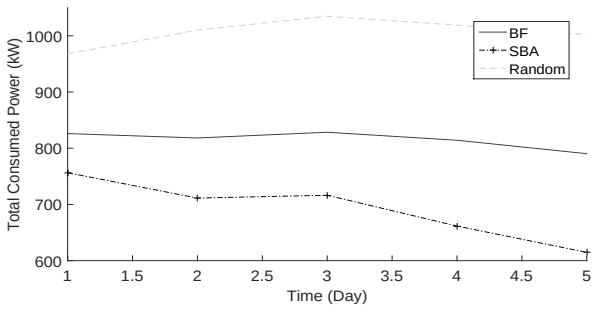

Fig. 2: The power drawn from the grid for Google DC over time for the different schemes.

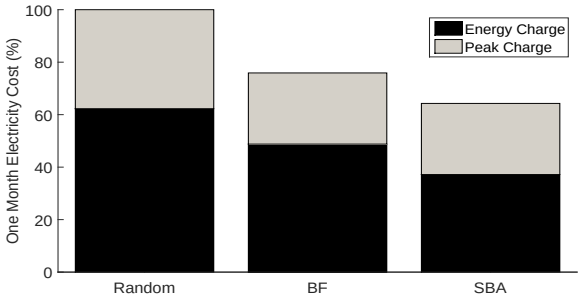

Fig. 3: Google DC's electricity bill for the different cluster management schemes.

the ON servers to discharge/charge energy. An OFF server is selected at random to store energy only if all the ON servers' batteries are full.

A. Energy Consumption

We start first by comparing the Google DC's energy consumption under the different schemes. We follow the model in [13] where the consumed power of an individual server depends on its utilization, U^{cpu}, and is calculated as follows:

$$P_c\left(U^{cpu}\right) = P_{idle} + U^{cpu}\left(P_{peak} - P_{idle}\right) \qquad (2)$$

where $P_{idle} = 200$, and $P_{peak} = 400$ Watts. Also, switching a server from ON to sleep and from sleep to ON incurs an energy consumption of 5510, and 4260 respectively [14]. OFF servers don't consume any power. Figure 2 illustrates the energy consumption of the Google DC over time. Observer that our framework has the lowest power consumption. This proves the efficiency of our framework and highlights how important it is to make efficient placement algorithm along with UPS management control for energy reduction in cloud centers.

B. Electricity Bill

We evaluate next in Fig. 3 the electricity bill of Google DC under the different schemes. We use a real power prices [9], where the Energy Charge and Peak Charge price rates are respectively $0.05\$/kWh$ and $20\$/kW$. The results are normalized with respect to the total electricity bill of the random placement scheme. Observe that our framework achieves the lowest energy cost among other schemes where the total bill is around 35% and 25% less than the random and the BF scheme respectively. This reduction is achieved as our framework consumed less amount of energy and also had a lower peak during the billing cycle.

C. Utilization Gain

Our next comparison is going to be about utilization gain that our framework achieves compared to others. CPU utilization of a server is the summation of its CPU resources that has been reserved for all of its hosted VMs divided by its total capacity. Figure 4 shows the average CPU utilization over time for all of the ON servers in the cluster under the three schemes. Observe that our framework utilization is not better than BF but close to it because during VMs placement our algorithm balances between reducing the number of ON servers and maximizing the accessible stored energy while BF focus only on minimizing number of ON servers. That means even though BF has better utilization, our framework incurs less energy consumption which leads to significant money savings.

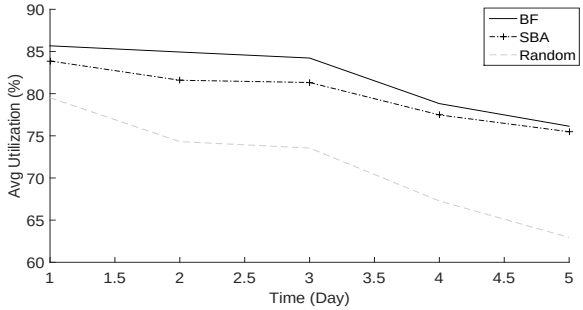

Fig. 4: Google DC's average CPU utilization over time for the different cluster management schemes

V. Conclusion and Future Work

This paper proposes a framework that decides on which server a new submitted VM request should be placed and which server's battery needs to charge/discharge energy and by how much with the aim of minimizing the Energy and Peak charges of the electricity bill. Results based on Google traces show that our proposed framework makes promising reductions in the total electricity bill. For future work, we plan to investigate how to perform workload migrations in order to reduce further the amount of inaccessible locked-in energy so that peak power demands can be shaved further and hence higher electricity bill reductions can be achieved.

VI. Acknowledgment

This work was made possible by NPRP grant # NPRP 5-319-2-121 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

REFERENCES

[1] J. Glanz and P. Sakuma, "Google details, and defends, its use of electricity," *The New York Times*, vol. 8, 2011.

[2] B. Urgaonkar, G. Kesidis, U. Shanbhag, and C. Wang, "Pricing of service in clouds: optimal response and strategic interactions," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 3, pp. 28–30, 2014.

[3] J. Shuja, K. Bilal, S. Madani, M. Othman, R. Ranjan, P. Balaji, and S. Khan, "Survey of techniques and architectures for designing energy-efficient data centers," *IEEE Systems Journal*, 2014.

[4] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "Exploiting task elasticity and price heterogeneity for maximizing cloud computing profits," 2015.

[5] M. Shojafar, S. Javanmardi, S. Abolfazli, and N. Cordeschi, "Fuge: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method," *Cluster Computing*, vol. 18, no. 2, pp. 829–844, 2015.

[6] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, 2012.

[7] W. Song, Z. Xiao, Q. Chen, and H. Luo, "Adaptive resource provisioning for the cloud using online bin packing," *IEEE Transactions on Computers*, 2013.

[8] M. Dabbagh, A. Rayes, B. Hamdaoui, and M. Guizani, "Peak shaving through optimal energy storage control for data centers," in *IEEE International Conference on Communications (ICC)*, 2016.

[9] Di Wang, Chuangang Ren, Anand Sivasubramaniam, Bhuvan Urgaonkar, and Hosam Fathy, "Energy storage in datacenters: what, where, and how much?," in *ACM SIGMETRICS Performance Evaluation Review*, 2012, vol. 40, pp. 187–198.

[10] R. Urgaonkar, B. Urgaonkar, M. Neely, and A. Sivasubramaniam, "Optimal power cost management using stored energy in data centers," in *ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, 2011, pp. 221–232.

[11] V. Kontorinis and et al., "Managing distributed UPS energy for effective power capping in data centers," in *the Annual International Symposium on Computer Architecture (ISCA)*, 2012.

[12] C. Reiss, J. Wilkes, and J. Hellerstein, "Google cluster-usage traces: format+ schema," *Google Inc., White Paper*, 2011.

[13] L. Barroso and U. Holzle, "The case for energy-proportional computing," *Computer Journal*, 2007.

[14] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "Efficient datacenter resource utilization through cloud resource overcommitment," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2015.