AN ABSTRACT OF THE THESIS OF

Sultan AlAnazi for the degree of Master of Science in Computer Science presented on July 25, 2016.

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

Abstract approved: ______________________________

Bechir Hamdaoui

The grid company enforces high penalties for the peak power demands of cloud data centers. These high penalties result in high electricity bills that can be avoided by relying on the servers' Uninterruptible Power Supply (UPS) as a source of energy during peak load periods. This thesis proposes a management framework that exploits the distributed UPS batteries in order to minimize the cluster's total electricity bill. Our framework consists of: *i*) a scheduler that accounts for both the amount of stored 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 during peak periods. Several evaluations based on real Google traces show that our proposed framework achieves significant monetary savings.

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

by

Sultan AlAnazi

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented July 25, 2016
Commencement June 2017

Master of Science thesis of Sultan AlAnazi presented on July 25, 2016.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

Dean of the Graduate School

Sultan AlAnazi, Author

ACKNOWLEDGEMENTS

First of all I would like to thank God for giving me the strength and hope to to achieve my goals. Second, I would like to give a big credit to Mehiar who shares with me this work achievements. I would like also, to give thanks to my dear parents and would like to dedicate this thesis to them for their unlimited support, love and encouragements throughout all the endeavors that I have been through and all my brother and sisters. Also, I would like to thank my dear wife for her patience, love and support during this journey. I would like to give many thanks to all friends who have supported me. I would like to express my sincere gratitude also to my committee members Professor Larry Cheng, Professor Lizhong Chen, Professor Jinsub Kim and Professor Leonard Coop, for accepting to serve as members of the committee of my defense and for the time they devoted to read this thesis. Lastly I would like to express my sincere gratitude and appreciation to my academic advisor Professor Bechir Hamdaoui, this thesis would not have been completed without his guidance and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

Chapter 1: Introduction

Energy cost has become a serious concern for most data centers today. Electricity bill payments contribute to a pretty significant portion of Data center annual operational expenditures (op-ex). As reported by Natural Resources Defense Council (NRDC), 91 billion Kilo-watt-hours of electricity were consumed by US Data centers alone in 2013 [1]. In the future, this amount is projected to increase to roughly 140 billion kilowatt-hours annually by 2020. The result of this increase is going to cost American businesses 13 billion dollar annually in electricity bills.

According to [15], Google continuously draws 260 Mega Watt of power in order to run its 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 bills in every possible way in order to reduce their operational expenses and increase their profits.

Cloud data center is a repository that contains significant amount of computing resources, called physical machines (PMs). These PMs are put together forming multiple groups. Each group is controlled by a management unit called cluster. A cluster can be homogeneous (servers have the same amount of resources) or heterogeneous (servers have different amounts of resources). Each cluster is located in a certain geographical area and is designated a resource manager that orchestrates the operation of a fleet of thousands of servers. These computing resources are offered as services to cloud clients and they will be charged based on their usage. The resource manager receives Virtual Machine (VM) requests from clients, with 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.

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 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 [17] 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.

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 [27]: *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 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.

[1]Peak Charge is also called Demand Charge.

1.1 Related Work

Researchers have proposed several peak shaving techniques as a solution to reduce Electricity prices in data centers such as Dynamic Voltage Frequency Scaling (DVFS)[14], virtual Machine-based power management [21], and online job migration [4, 31]. Although prior work techniques lead to energy savings, they all incur some performance overhead, which is not desirable to cloud clients. The use of batteries can be thought of as another peak shaving approach that reduces energy costs in data centers [16, 19, 22, 28] if exploited properly. They are more practical and they incur no performance overhead. Recently, a power capping technique using heterogeneous battery environment was introduced by [18]. Another work proposed by [30] to reduce power consumption using a power model that maps the workload to its dissipated power. Although both approaches resulted in significant power reduction, they did not consider per-server homogeneously distributed batteries.

There have also been numerous cluster management techniques proposed to minimize the Energy Charge of the electricity bill [3, 6, 8, 10, 12, 13, 24–26]. The most common approach is to consolidate the VM requests on as few ON servers as possible, thus allowing the switching of redundant servers to sleep to save energy [5, 8, 11, 20, 24]. The Best-Fit (BF) heuristic [3, 26] is the most popular VM placement heuristic that tries to achieve this objective via VM placement and scheduling. While the BF makes significant Energy Charge reduction compared to random VM placement strategies, it completely ignores minimizing the Peak Charge which contributes to more than 40% of the electricity bill [9].

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 [29]. 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 [28]. 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.

1.2 Contribution

In this thesis, 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 aims at minimizing both the number of ON server and the amount of inaccessible stored energy for clusters with distributed UPS batteries.
- Propose a greedy power distribution strategy that decides which UPS battery should charge/discharge power 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. Chapter 2 introduces our notations. Chapter 3 explains our proposed framework. Chapter 4 evaluates our framework based on real Google traces. Finally chapter 5 concludes and provides directions for future work.

Chapter 2: System Model and Notation

2.1 Battery Model

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 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} that can't be exceeded. Each battery has also a conversion charging efficiency η_{c^+}, a conversion discharging efficiency η_{c^-} and a leakage efficiency η_l where $0 \leq \eta_{c^+}, \eta_{c^-}, \eta_l \leq 1$. This means that when a server's battery draws a power P from the gird then only $\eta_{c^+}P$ ends up being stored in the battery where as the remaining power gets lost due to conversion losses. Similarly when the battery discharges a power P to be used by the server then only $\eta_{c^-}P$ gets delivered to the server's components whereas the remaining power gets lost due path losses.

2.2 Power and Energy

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 T be a constant threshold value used to cap data center's power consumption. Let $\mathbb{P}$ be the set of all servers in the cluster. Let $\mathbb{P}_{on}$ and $\mathbb{P}_{off}$ be the set of all ON and OFF servers in the cluster respectively where: $\mathbb{P} = \mathbb{P}_{on} \cup \mathbb{P}_{off}$. 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.

For a slot i, the following notations are used:

- $d_{i,j}$ 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: $D_i = \sum_{j \in \mathbb{P}} d_{i,j}$.

- $e_{i,j}$ is the amount of energy stored in the battery attached to server j at the beginning of the i^{th} slot.

- $p_{i,j}$ is the amount of energy the battery attached to server j can possibly have during the i^{th} time slot, before reaching its maximum capacity, which can be calculated as: $p_{i,j} = E_{max} - e_{i,j}$.

- $c^{+}_{i,j}$ is the possible energy charge the Battery attached to server j can possibly have during the i^{th} time slot. This possible charge is limited by the maximal battery capacity $p_{i,j}$, the data center's threshold $t_{i,j}$and the server's maximal charging rate C_{max} as illustrated in the following relation: $a^{+}_{i,j} = \min(p_{i,j},\, t_{i,j},\, C_{max})$.

- $c^{-}_{i,j}$ is the accessible power that can be discharged from the battery attached to server j at the beginning of the i^{th} slot. This accessible power is limited by the server's power demand $d_{i,j}$, the server's stored energy $e_{i,j}$ and the server's maximal discharging rate C_{max} as illustrated in the following relation: $a^-_{i,j} = \min(d_{i,j},\, e_{i,j}/\tau,\, C_{max})$.

- C^-_i is the amount of power that our framework decides to discharge from all the batteries in the cluster during the i^{th} slot.
- C^+_i is the amount of power that our framework decides to charge from all the batteries in the cluster during the i^{th} slot.

Chapter 3: The Proposed Framework

As illustrated in Fig. 3.1, 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:

3.1 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 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

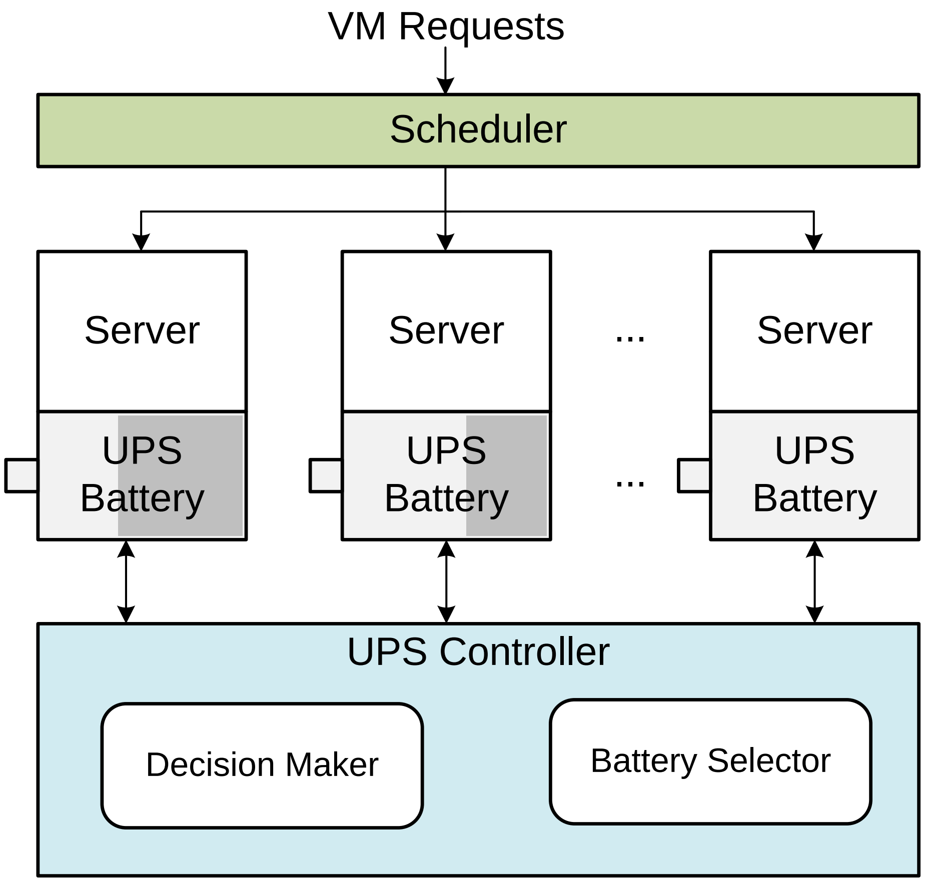

Figure 3.1: Proposed Framework.

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 follows[1]:

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

Where $S_{Cap}^{cpu}(j)$ 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].

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

$$S_{Cap}^{cpu}(j) = C_j^{cpu} / C_{max}^{cpu}$$

[1]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.

where C_j^{cpu} is the CPU capacity of the j^{th} OFF server and C_{max}^{cpu} is the maximum server's CPU capacity among the OFF servers in the cluster.

On the other hand, the UPS stored energy score is calculated as follow:

$$S_{UPS}(j) = E_j/E_{max} \tag{3.1}$$

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 server within the cluster.

- If multiple ON server can provide the resource demands for the submitted VM request, then SBA prefers the ON server with the larger CPU utilization and with the larger 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 larger amount of stored energy in their attached UPS battery are preferred as they hold a larger amount of energy that can be used for peak shaving in the future. In order to select the server with larger capacity and with larger amount of stored energy in the attached UPS battery, 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 a server j that is ON and that can provide the VM's demands. the score $S(j)$ is calculated as follows:

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

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

While $S_{UPS}(j)$ is calculated as was previously described in equation (3.1), the CPU Utilization score $S^{cpu}_{Util}(j)$ is calculated as follows:

$$S^{cpu}_{Util}(j) = U^{cpu}_{j} / U^{cpu}_{max}$$

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.

3.2 UPS Controller

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

3.2.1 Decision Maker

Algorithm 1 Decision Maker(D_i, T)

```
1: if $D_i > T$ then
2:     $C^- \leftarrow D_i - T$
3:     $SelectDischargeBattery(C^-)$
4: else
5:     $C^+ \leftarrow T - D_i$
6:     $SelectChargeBattery(C^+)$
7: end if
```

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 decisions maker is illustrated as a pseudo code (Algorithm 1) and takes as input the DC's power demand at the i^{th} slot D_i, and a predefined threshold T. The decision maker compares the DC's power demand to a constant threshold T. 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 (periods during which the DC's power demand is below the threshold). This stored energy is latter used to power partially or fully the DC during high demand periods (periods during which the DC's power demand is above the threshold) which reduces the peak charge and hence minimizes the monthly electricity bill.

3.2.2 Battery Selector

This sub-module decides which batteries among those attached to the servers in the cluster 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' battery. 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 decisions 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
```

Chapter 4: Performance Evaluation

In this chapter, the experiments and evaluations are conducted based on real Google cluster traces [23] released in November 2011 and Lead Acide(LA) batteries. Tables 4.1 and 4.2 summarize these traces.

Table 4.1: Description of Google Trace

Characterization of the trace	value
Duration of Traces	29 days
Number of servers	> 12K
Number of task requests	> 50M
Compressed size of data	39GB

Table 4.2: Configuration of PMs within the Google trace

Number of PMs	Architecture	CPU
11659	A	0.50
798	B	1.00
126	C	0.25

LA batteries are the most commonly used batteries in data centers due to their ability to produce high current at lower cost. The following table summaries LA battery specifications:

Table 4.3: LA Battery Specs

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

To evaluate our framework efficiency, we compare our proposed technique (referred to as **SBA** in the graphs) with the two following schemes:

- **Best Fit (BF) Placement**: BF makes placement decisions to task requests in a way that reduces the number of ON servers in the cluster. It approaches this reduction by placing a new task on a server that is ON, fits the VM, and has the least CPU slack. In cases where no ON servers can fit the VM, then the OFF server with the maximum CPU capacity would be switched ON.
- **Random Placement**: This heuristic makes random placement decision whenever a new VM request is submitted. It places the new VM randomly on a server that is ON and can fit the VM. If no ON servers can fit the VM, then the algorithm will place it randomly on one of the OFF servers that can fit the new VM.

The results of our framework evaluation will be discussed in terms of energy cost, power consumption and utilization gain.

4.1 Active PMs

The total number of active PMs plays an important role in reducing electricity bill. one way to minimize the number of ON PMs is to keep as much large CPU capacity PMs ON as possible. We can observe that our framework in Figure 4.1 has the lowest number of ON PMs almost all the time. SBA achieves this reduction because it keeps more large CPU capacity PMs ON than BF and Random.

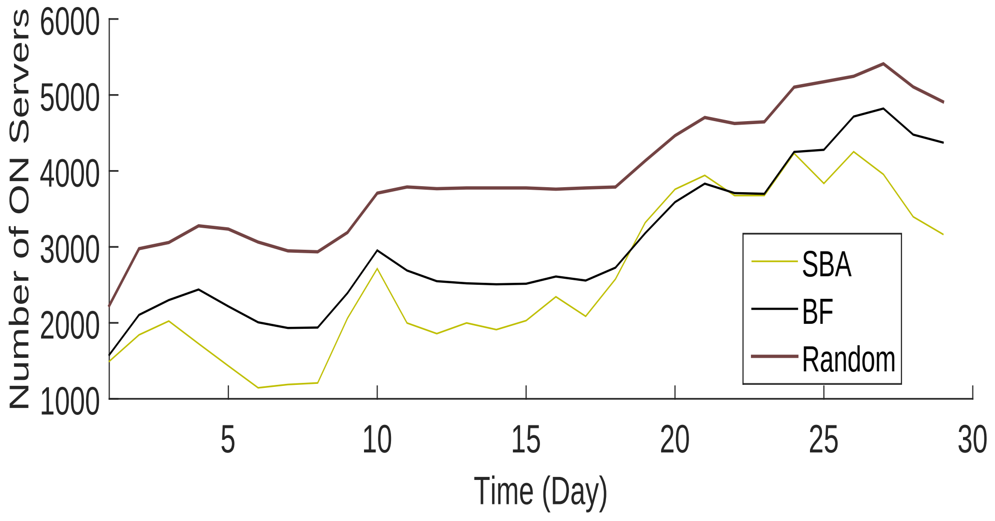

Figure 4.1: Number of PMs over time for the different placement schemes

4.2 Power Demand

In this section we compare a cluster's power demand under the different schemes. This comparison evaluates the power demand without using UPS batteries to show how efficient the scheduler is for the three frameworks. Figure 4.2 shows the power demand of three framework. We can see that our scheduling algorithm leads to the lowest power demand compared to BF and Random. It achieves this reduction because of utilizing ON PMs that have large CPU capacity more than the other schemes. Utilizing large CPU capacity PMs reduces the total number of ON PMs in the cluster, hence reducing power demand. Next to show the efficiency of the frameworks in general, we evaluate the energy consumption using UPS batteries to partially supply data center with power during high demand periods under the different schemes.

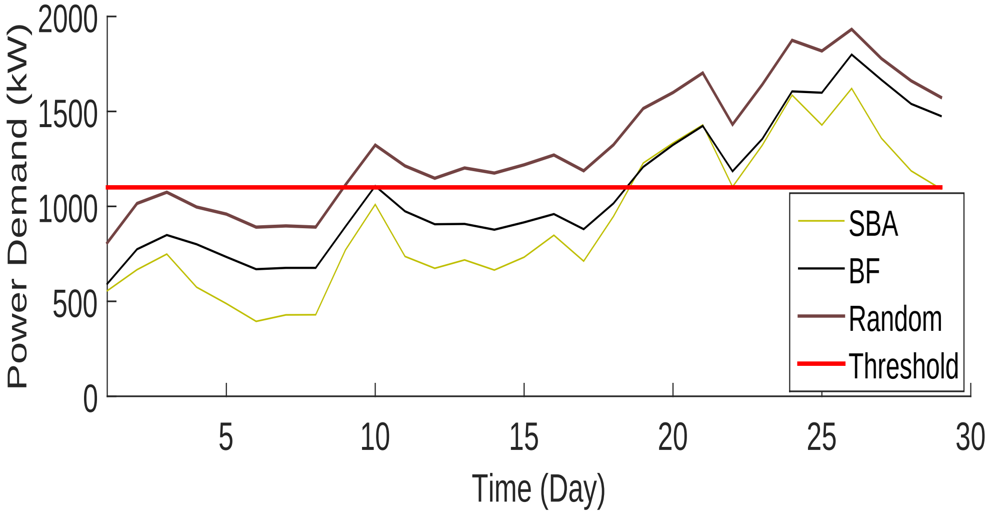

Figure 4.2: Power demand over time for the different placement schemes

4.3 Locked-in Power

Locked-in power is the amount of power that is not accessible for power supply during peak periods due to the nature of UPS topology where a battery can only supply power to it's attached server. When servers are OFF, there attached batteries are not accessible (locked-in). Figure 4.3 shows the locked-in power over time for the different placement schemes. We can see that our heuristic (SBA) has the maximum locked-in power during the first half of the month, however it has a sharp decrease around the other half. The reason for this is that SBA has the lowest number of active PMs which means more OFF servers and more locked-in power, also during the first Fifteen days as depicted in Figure 4.2, the cluster power demand is bellow the threshold where batteries are not yet used to supply data center with power. Nevertheless when cluster demand exceeds the threshold, the locked-in power is minimized significantly by our heuristic compared to the other scheme.

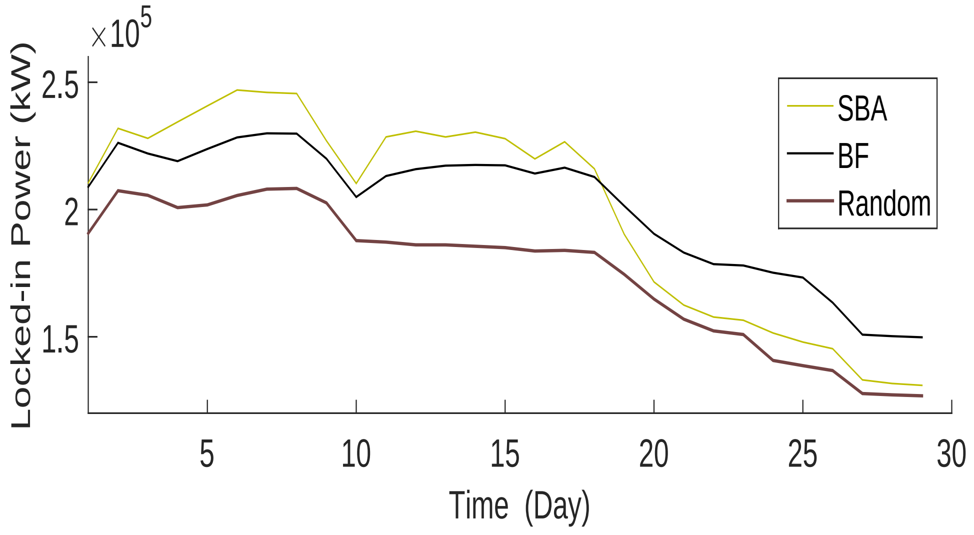

Figure 4.3: Locked-in Power over time for the different placement schemes

4.4 Energy Consumption

Now we compare the Google DC's energy consumption under the different schemes. We follow the model in [2] where the consumed power of an individual server depends on its utilization, U^{cpu}, and it is calculated as follows:

$$P_c\left(U^{cpu}\right) = P_{idle} + U^{cpu}\left(P_{peak} - P_{idle}\right) \tag{4.1}$$

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 Jules respectively [7]. OFF servers do not consume any power. Figure 4.4 illustrates the energy consumption of the Google DC over time. Observer that that the SBA 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.

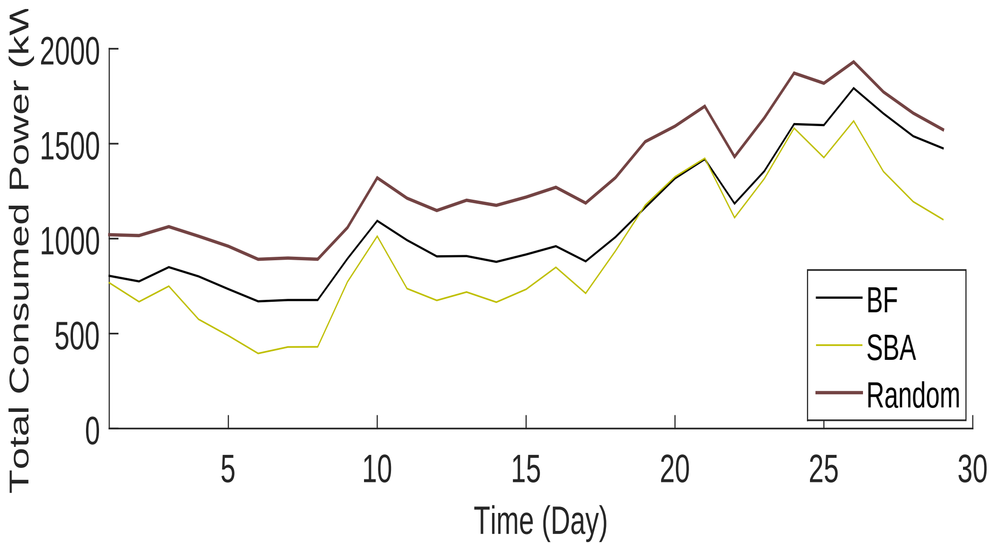

Figure 4.4: Data Center Power Consumption over time with different placement scheme

4.5 Electricity Bill

We evaluate next in Fig. 4.5 the electricity bill of Google DC under the different schemes. We use a real power prices [29], where the energy charge price is 0.05\$/kWh whereas peak charge is 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 20% and 10% 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.

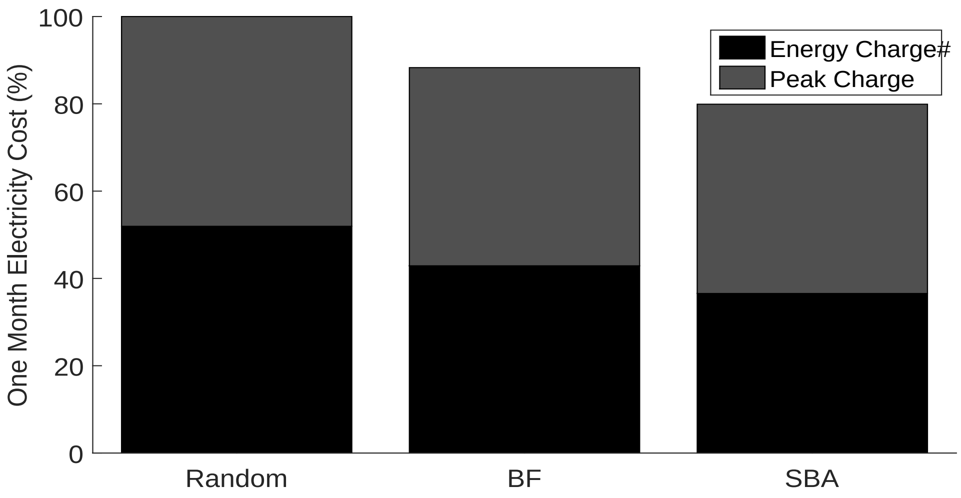

Figure 4.5: Total Energy Cost for the different placement schemes normalized w.r.t Random Placement

4.6 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 been reserved for all of its hosted VMs divided by its total capacity. Figure 4.6 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 because it has more large CPU capacity servers ON than BF. That means even though BF has better utilization, Our framework incur less energy consumption which leads to significant money savings.

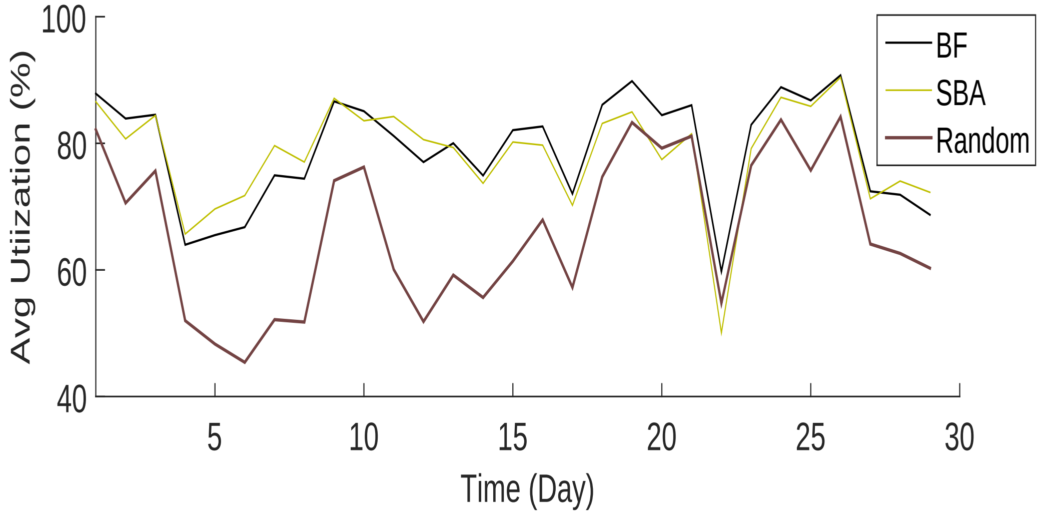

Figure 4.6: Average Utilization over time for the different placement schemes

Chapter 5: Conclusion

We propose in this paper a framework that efficiently controls UPS batteries to maximize the total electricity cost savings. The results of evaluating our proposed framework on a real Google cluster traces shows how efficient our framework compared to other schemes in terms of energy cost savings. For future work, we plan to eliminate battery locked energy, that limits shaving long duration peaks, by applying workload migration among servers.

Bibliography

[1] America's data centers consuming and wasting growing amounts of energy. `https://www.nrdc.org/resources/americas-data-centers-consuming-and-wasting-growing-amounts-energy`. Accessed: 2015-02-06.

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

[3] 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.

[4] Niv Buchbinder, Navendu Jain, and Ishai Menache. Online job-migration for reducing the electricity bill in the cloud. In *International Conference on Research in Networking*, pages 172–185. Springer, 2011.

[5] Jeffrey S Chase, Darrell C Anderson, Prachi N Thakar, Amin M Vahdat, and Ronald P Doyle. Managing energy and server resources in hosting centers. *ACM SIGOPS Operating Systems Review*, 35(5):103–116, 2001.

[6] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes. Efficient datacenter resource utilization through cloud resource overcommitment. In *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 330–335, April 2015.

[7] 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.

[8] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes. Exploiting task elasticity and price heterogeneity for maximizing cloud computing profits. *IEEE Transactions on Emerging Topics in Computing*, 2016.

[9] 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.

[10] Mehiar Dabbagh, Bechir Hamdaoui, Mohsen Guizani, and Ammar Rayes. Energy-efficient cloud resource management. In *INFOCOM Workshops*, pages 386–391, 2014.

[11] Mehiar Dabbagh, Bechir Hamdaoui, Mohsen Guizani, and Ammar Rayes. Release-time aware vm placement. In *2014 IEEE Globecom Workshops (GC Wkshps)*, pages 122–126. IEEE, 2014.

[12] Mehiar Dabbagh, Bechir Hamdaoui, Mohsen Guizani, and Ammar Rayes. Energy-efficient resource allocation and provisioning framework for cloud data centers. *IEEE Transactions on Network and Service Management*, 12(3):377–391, 2015.

[13] Mehiar Dabbagh, Bechir Hamdaoui, Mohsen Guizani, and Ammar Rayes. An energy-efficient VM prediction and migration framework for overcommitted clouds. *IEEE Transactions on Cloud Computing*, 2016.

[14] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power provisioning for a warehouse-sized computer. In *ACM SIGARCH Computer Architecture News*, volume 35, pages 13–23. ACM, 2007.

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

[16] Sriram Govindan, Anand Sivasubramaniam, and Bhuvan Urgaonkar. Benefits and limitations of tapping into stored energy for datacenters. In *Computer Architecture (ISCA), 2011 38th Annual International Symposium on*, pages 341–351. IEEE, 2011.

[17] 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.

[18] Shen Li, Shaohan Huneycutt, Shiguang Wang, Siyu Gu, Chenji Pan, and Tarek Abdelzaher. Wattvalet: Heterogenous energy storage management in data centers for improved power capping. In *11th International Conference on Autonomic Computing (ICAC 14)*, pages 273–279, 2014.

[19] D. Meisner, C. M. Sadler, L. A. Barroso, W. D. Weber, and T. F. Wenisch. Power management of online data-intensive services. In *Computer Architecture (ISCA), 2011 38th Annual International Symposium on*, pages 319–330, June 2011.

[20] Ripal Nathuji and Karsten Schwan. Virtualpower: coordinated power management in virtualized enterprise systems. In *ACM SIGOPS Operating Systems Review*, volume 41, pages 265–278. ACM, 2007.

[21] Ripal Nathuji, Karsten Schwan, Ankit Somani, and Yogendra Joshi. Vpm tokens: virtual machine-aware power budgeting in datacenters. *Cluster computing*, 12(2):189–203, 2009.

[22] Darshan S Palasamudram, Ramesh K Sitaraman, Bhuvan Urgaonkar, and Rahul Urgaonkar. Using batteries to reduce the power costs of internet-scale distributed networks. In *Proceedings of the Third ACM Symposium on Cloud Computing*, page 11. ACM, 2012.

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

[24] 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*, 18(2):829–844, 2015.

[25] 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.

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

[27] B. Urgaonkar, G. Kesidis, U. Shanbhag, and C. Wang. Pricing of service in clouds: optimal response and strategic interactions. *ACM SIGMETRICS Performance Evaluation Review*, 41(3):28–30, 2014.

[28] 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*, pages 221–232, 2011.

[29] 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*, volume 40, pages 187–198, 2012.

[30] Quan Zhang and Weisong Shi. Ups-aware workload placement in enterprise data centers. *IEEE Computer Magazine*, 2015.

[31] Yanwei Zhang, Yefu Wang, and Xiaorui Wang. Capping the electricity cost of cloud-scale data centers with impacts on power markets. In *Proceedings of the 20th international symposium on High performance distributed computing*, pages 271–272. ACM, 2011.