Energy efficient scheduling in data centers

Paul, Debdeep; Zhong, Wende; Bose, Sanjay K

2015

Paul, D., Zhong, W., & Bose, S. K. (2015). Energy efficient scheduling in data centers. 2015 IEEE International Conference on Communications (ICC), 5948-5953.

<https://hdl.handle.net/10356/80296>

<https://doi.org/10.1109/ICC.2015.7249270>

© 2015 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The published version is available at: [<http://dx.doi.org/10.1109/ICC.2015.7249270>].

Downloaded on 26 Apr 2023 18:11:50 SGT

Energy Efficient Scheduling in Data Centers

Debdeep Paul, Wen-De Zhong

School of Electrical and Electronic Engineering
Nanyang Technological University

Singapore

debdeep1@e.ntu.edu.sg, EWDZhong@ntu.edu.sg

Sanjay K. Bose

Department of Electronics and Electrical Engineering
Indian Institute of Technology, Guwahati

India

skbose@iitg.ernet.in

Abstract—The explosive growth of the Internet has resulted in tremendous growth of the data centers which primarily serve as the cloud computing backbones. We consider the practical architecture of a real data center where different types of services are implemented in different tiers. We use Model Predictive Control (MPC) based energy aware scheduling algorithms to address the resource allocation problem for deferrable jobs in such a tiered architecture, where the switching cost is accounted for in two different ways. We compare the performance of these two algorithms with respect to the important performance parameters of cumulative electricity cost, cumulative renewable usage and cumulative number of switchings. We discuss the results to provide some insightful understanding on these issues.

I. INTRODUCTION

The past decade has witnessed enormous growth in Internet technology and its penetration in services and applications. This is continuing with the advent of innovative technologies such as Mobile Computing, Internet of Things (IoT), Big Data and other applications. This prolific increase in the amount of traffic over the Internet also requires the data centers to grow rapidly in step. A consequence of this is that the Cloud Service Providers (CSP), such as Google, Facebook and Amazon, incur a massive electricity costs to support their operations at this enormous scale. In 2011, Google officially revealed that the data centers owned by them continuously draw almost 260 million watts of power [1], which translates into an annual electricity bill of about \$138M [2]. Geographical Load Balancing can reduce power costs by effectively leveraging the spatial variation in the price of electricity between different locations. A few notable works presenting efficient algorithms to implement electricity cost aware geographical load balancing are [2, 3]. The work in [3] combines the objective of power cost reduction with greater integration of renewable energy into the grid. With the increasing importance of operating in an environmentally sustainable fashion, different CSPs are contributing towards reducing their carbon footprint. A recent report [4] presented a detailed statistical analysis on the energy footprint practices of big CSPs around the world. In [5], the authors presented a fairly rigorous discussion about the opportunities and challenges to integrate renewables in powering data centers. They concluded that data centers exhibit two inherent properties which make them particularly suitable to exercise Demand Response (DR) strategies to

integrate time varying renewable energy. These are i) they are large consumers of electricity and ii) they deal with deferrable jobs, which only have to be serviced before their stipulated deadlines. The amount of time by which a deferrable job can be deferred is referred to as its *slack* [5]. This can provide cost savings and more renewable integration if intelligent job scheduling is incorporated. In this work, we harness the slack of deferrable jobs. We apply control theoretic tools to implement energy efficient resource scheduling algorithms to achieve electricity cost reduction and increase the use of renewable energy in a data center. The main challenge to dynamically vary the number of active servers as per the amount of jobs is that the transition from low power state (OFF) to high power state (ON) incurs significant switching cost. The various sources of this cost are discussed in [6]. The stochastic nature of the job arrival process makes *dynamic right sizing* [6] a challenging problem. The authors of [6] present an efficient online algorithm to dynamically control the number of active servers based on the number of jobs, with little knowledge about the future behaviour of job arrivals. In addition, they highlight the necessity of understanding the theoretical basis of energy efficient online algorithms [7].

Lastly, and most importantly, CSPs incur a massive amount of capital cost to build a data center. As a thumb rule the scientists from Google reported empirically that the capital expenditure (Capex) incurred by the CSP to provide 1 watt of power to a data center ranges between \$10 to \$20 [8]. For example, in 2007 Microsoft purchased two 10-MW data centers in California for \$200 million. Many such statistics are presented in [8]. In this work we present an algorithm to achieve Capex reduction on top of operational expenditure (Opex) reduction while increasing renewable usage. In summary, our contributions are:

1. We explicitly investigate the opportunity of peak server reduction which directly reduces capital costs.
2. We tackle the issue of resource scheduling for deferrable jobs through control theoretic methods and propose techniques for energy aware scheduling.
3. We present a practical tiered design model of a data center to capture the requirement of allocating servers equipped with different functions to appropriate jobs.

The organization of the rest of the paper is as follows. Section II describes the mathematical model of a data center with a tiered architecture which serves as a base for implementing energy efficient scheduling algorithms. We

propose two optimization problems for server provisioning at different tiers, based on a discrete time model. The first optimization problem is for joint minimization of a combination of Capex, Opex and switching costs. The second optimization problem aims to minimize the Opex and switching costs. Section III describes the simulation set up and presents and discusses the simulation results. Section IV concludes the paper.

II. SYSTEM MODEL

A. Model for tiered architecture of a data center

Today's data centers provide services to a broad spectrum of applications with workloads which have very different functional requirements. In reality, the different functionalities are implemented in different tiers [9]. Depending on the nature of services provided by the CSP and the data center, the deployment of the different tiers may differ from one CSP to another. Most commonly implemented tiers of a commercial data center supporting web services are web, application and data base tiers [9]. Typically, a set of servers equipped with specific functions which provide similar services and are connected to each other by communication links would constitute a tier. Earlier works for enhancing energy efficiency assumed a simple flat architecture for a data center, which is not what would be generally found in practice. Typically, a data center consists of a load dispatcher, which directs each incoming jobs to the appropriate tier(s). Data centers serve two kinds of jobs, i) non-deferrable jobs and ii) deferrable jobs (which must be serviced before stipulated deadlines). For the non-deferrable jobs, the server provisioning strategies are obtained by applying suitable *queueing theoretic* models [6]. There will also be Service Level Agreements (SLA) between the CSP and its clients to define various quality of service (QoS) parameters and their specifications. A number of effective strategies to tackle the dynamic server provisioning issue for non-deferrable jobs are proposed and implemented in [3, 6, 9]. Very little work has however been done on optimum server provisioning for deferrable jobs, which is the focus of this paper.

B. MPC based optimization framework

We address the issue of server provisioning for deferrable jobs by leveraging a technique prevalent in control engineering, called Model Predictive Control (MPC) which is also known as Receding Horizon Control (RHC). A fairly detailed discussion on its analytical background and implementation of MPC based algorithms under different application settings are provided in [10]. We choose MPC because there is a fundamental trade-off between electricity cost reduction, renewable integration and switchings. On one hand if electricity cost or renewable integration is the prior focus then we can implement an algorithm which schedules the job to match renewable generation and harnesses the temporal diversity of electricity price to achieve cost reduction. In this case, however, the number of switchings can be arbitrarily high. On the other hand, if the CSP is unsure about energy savings by turning off extra servers then it can

choose to keep all the servers always ON. Under MPC, the optimization problem is solved at each timeslot to compute the decision variables over the entire window and the actions for the first timeslot are applied to the system. After applying this action, the problem instance changes. The optimization problem is then solved again in the next timeslot with the updated data set.

We consider a discrete time model whose timeslot length is the time at which the CSP updates the server provisioning decision. There is no switching of servers within a single timeslot. We denote the deferrable jobs as tasks. Each task requires a certain amount of jobs to be executed at various tiers. To avoid notational complexity, we call the deferrable tasks as deferrable jobs and assume that the tasks are preemptive. We capture the power required by cooling, power distribution systems, and other supporting systems through the parameter called Power Usage Effectiveness (PUE). PUE is defined as the ratio of the total power consumed to the power consumed by the IT system. This is the most popular direct method of measuring the energy efficiency of a data center [8].

C. Joint Minimization of Capex and Opex (CapexOpexMin Procedure)

The power consumption of a commercial server generally satisfies an affine relationship (linear plus a constant) with the normalized server load [6]; i.e., if the load is ρ , then the power consumption $\varepsilon(\rho)$ is of the form

$$\varepsilon(\rho) = \alpha_0 + \alpha_1 \rho; 0 \leq \rho < 1 \quad (1)$$

where α_0 and α_1 are constants whose values depend on the power characteristics of a server. The work of [6] rigorously established that a uniform load balancing among the active servers achieves an optima with respect to a linear combination of energy cost, delay cost (which captures the cost of revenue loss due to delay) and switching cost. We augment the work of [6] by considering other crucial aspects such as renewables, electricity price, and most importantly, the deferrable jobs based on the tiered architecture of the data center. We assume the data center consists of J tiers and j is the index to denote a particular tier. The vectors are represented with an $(\rightarrow)$ above and are J dimensional column vectors unless mentioned otherwise.

The entries of the vector $\vec{n}_\tau$ are $n_{\tau,j}$, are the total number of servers provisioned at tier j for timeslot τ . Similarly $\lambda_{\tau,j}$ is the total amount of jobs (sum of deferrable and non-deferrable) executed at tier j in timeslot τ . Also, $\vec{\lambda}_\tau$ is the vector whose entries are $\lambda_{\tau,j}$. The *total server energy consumption* ($P_{\tau,j}^S$) for timeslot τ at tier j is given by:

$$P_{\tau,j}^S = n_{\tau,j} \varepsilon\left(\frac{\lambda_{\tau,j}}{n_{\tau,j}}\right) = \alpha_0 n_{\tau,j} + \alpha_1 \lambda_{\tau,j}, \forall \tau, j \quad (2)$$

We denote the PUE of the data center in timeslot τ as θ_τ . For the whole data center considering all the tiers, the *total energy consumption* (P_τ^{Tot}) for timeslot τ is given by:

$$P_\tau^{Tot} = (\alpha_0 \vec{1}^T \vec{n}_\tau + \alpha_1 \vec{1}^T \vec{\lambda}_\tau) \theta_\tau \quad (3)$$

where $\bar{1}^T$ is the J dimensional row vector with all entries 1. We assume that the data center operator owns the renewable energy source and therefore there is no marginal cost for the renewable [11]. Hence the cost of electricity for timeslot τ (C_τ^E) is given by:

$$C_\tau^E = p_\tau \{P_\tau^{Tot} - r_\tau\}^+ \quad (4)$$

where $\{x\}^+ = \max(x, 0)$, p_τ and r_τ are the price of electricity and the amount of renewable energy for τ , respectively. The renewable energy r_τ is measured in the number of servers that can be powered by the available renewable energy.

In addition, we incorporate the *variance* and *peak* of the number of servers provisioned at the individual tiers into the objective function. The optimization problem we propose here differs from the subsequent one in the formulation of the switching cost. In the present case, we capture the switching cost as the variance of the number of servers provided in the various tiers and scale it by a factor β .

The total cost of switching across the prediction horizon, C_{Total}^{Var} is given by:

$$C_{Total}^{Var} = \frac{1}{T} \sum_{j=1}^J \sum_{\tau=1}^T \{n_{\tau,j} - \frac{1}{T} (\sum_{\tau=1}^T n_{\tau,j})\}^2 \quad (5)$$

This captures the variance of the number of servers provisioned at each of the tiers. It turns out that capturing the switching cost through the variance imposes a tighter control over switching and strikes a good trade-off between electricity cost reduction, renewable incorporation and switching cost. We explicitly incorporate the cost for peak provisioning at each tier and then sum it up across all the tiers. We denote the peak provisioning cost by C_{Total}^{Peak} , which is expressed as:

$$C_{Total}^{Peak} = \sum_{j=1}^J \max_{\tau} (n_{\tau,j}), \tau \in \{1, 2, \dots, T\} \quad (6)$$

We formulate the optimization problem as -

Minimize

$$\sum_{\tau=1}^T C_\tau^E + \beta C_{Total}^{Var} + \gamma C_{Total}^{Peak} \quad (7)$$

With the appropriate choices of β and γ , we control the relative priority among the power cost, variance (of the number of servers provisioned) and peak server provisioning capital cost. Our purpose is to get more insights on how to design the optimal strategy for server provisioning.

We now present the constraints of the optimization problem. The total amount of jobs executed $\lambda_{\tau,j}$ is the sum of non-deferrable jobs $\lambda_{\tau,j}^D$ and deferrable jobs $\lambda_{\tau,j}^{ND}$,

$$\lambda_{\tau,j} = \lambda_{\tau,j}^D + \lambda_{\tau,j}^{ND}, \forall \tau, j \quad (8)$$

Similarly, the total number of servers provisioned $n_{\tau,j}$ is the sum of ($n_{\tau,j}^D$) servers for non-deferrable jobs and ($n_{\tau,j}^{ND}$) servers for deferrable jobs

$$n_{\tau,j} = n_{\tau,j}^D + n_{\tau,j}^{ND}, \forall \tau, j \quad (9)$$

If $\phi(\cdot)$ is a function that determines the number of servers required to serve the non-deferrable jobs given by $\lambda_{\tau,j}^{ND}$ then

$$\phi(\lambda_{\tau,j}^{ND}) = n_{\tau,j}^{ND}, \forall \tau, j \quad (10)$$

Since, this is computed outside the optimization problem $\phi(\cdot)$ can take any closed form function or can even be given by an iterative algorithm as in [9]. This would then be outside the purview of energy aware scheduling, which deals with deferrable jobs. For the relationship between the amount of deferrable jobs and the number of servers to serve them, we assume a linear relationship as in (11) for simplicity. This model can handle any function to compute the number of servers given the amount of jobs as long as the function is convex.

$$n_{\tau,j}^D \geq 1.1 \lambda_{\tau,j}^D, \forall \tau, j \quad (11)$$

To deal with events such as breakdown, component failure and prediction uncertainty, we choose to perform an over-provisioning by 10% as given in (11). The deferrable tasks are indexed by ω . If $n_{\tau,j}^{D,\omega}$ is the number of servers allocated to task ω , at tier j in timeslot τ and if $n_{\tau,j}^D$ is the total number of servers provisioned for deferrable tasks at j in τ , then

$$\sum_{\omega \in \Omega_\tau} n_{\tau,j}^{D,\omega} = n_{\tau,j}^D, \forall \tau, j \quad (12)$$

where Ω_τ denotes the set of unfinished tasks in timeslot τ .

Similarly, for jobs we enforce,

$$\sum_{\omega \in \Omega_\tau} \lambda_{\tau,j}^{D,\omega} = \lambda_{\tau,j}^D, \forall \tau, j \quad (13)$$

In practice there can be some lower bound and upper bound on the number of servers which can be provided for ω at j on τ . Typically, the lower bound is zero unless there are some strict requirements on the execution of certain tasks. The upper bound arises if some tasks are required to be serviced within a specific set of servers with special functions. This may also arise due to security concerns or data locality issues. These two limits are imposed by:

$$n_{\tau,j}^{D,\omega,(-)} \leq n_{\tau,j}^{D,\omega} \leq n_{\tau,j}^{D,\omega,(+)}, \forall \tau, j \text{ and } \forall \omega \in \Omega_\tau \quad (14)$$

Likewise, for the amount of jobs belonging to task ω , there are similar constraints,

$$\lambda_{\tau,j}^{D,\omega,(-)} \leq \lambda_{\tau,j}^{D,\omega} \leq \lambda_{\tau,j}^{D,\omega,(+)}, \forall \tau, j \text{ and } \forall \omega \in \Omega_\tau \quad (15)$$

We sum up the lower bounds on the number of servers across all the active tasks to obtain the lower bound on the total number of servers to be allocated for the deferrable tasks. This can formally be represented as:

$$\sum_{\omega \in \Omega_\tau} n_{\tau,j}^{D,\omega,(-)} = n_{\tau,j}^{D,(-)}, \forall \tau, j \quad (16)$$

where $n_{\tau,j}^{D,(-)}$ is the lower bound on $n_{\tau,j}^D$.

Similarly, for the upper bounds we have:

$$\sum_{\omega \in \Omega_\tau} n_{\tau,j}^{D,\omega,(+)} = n_{\tau,j}^{D,(+)}, \forall \tau, j \quad (17)$$

where $n_{\tau,j}^{D,(+)}$ is the upper bound on $n_{\tau,j}^D$.

To ensure $n_{\tau,j}^D$ is within these bounds we enforce:

$$n_{\tau,j}^{D,(-)} \leq n_{\tau,j}^D \leq n_{\tau,j}^{D,(+)}, \forall \tau, j \quad (18)$$

Similarly, for the deferrable tasks,

$$\sum_{\omega \in \Omega_\tau} \lambda_{\tau,j}^{D,\omega,(-)} = \lambda_{\tau,j}^{D,(-)}, \forall \tau, j \quad (19)$$

$$\sum_{\omega \in \Omega_\tau} \lambda_{\tau,j}^{D,\omega,(+)} = \lambda_{\tau,j}^{D,(+)}, \forall \tau, j \quad (20)$$

$$\lambda_{\tau,j}^{D,(-)} \leq \lambda_{\tau,j}^D \leq \lambda_{\tau,j}^{D,(+)}, \forall \tau, j \quad (21)$$

Let a_ω and d_ω be the respective arrival and deadline timeslots of the deferrable task ω , where it requires L_j^ω amount of jobs to be serviced at tier j during this period. This constraint is formally represented as:

$$\sum_{\tau=a_\omega}^{d_\omega} \lambda_{\tau,j}^{D,\omega} = L_j^\omega, \forall j \text{ and } \forall \omega \in \Omega_\tau \quad (22)$$

Another constraint is to ensure that for each timeslot the total number of servers provisioned at each tier is less than the total number of servers at that tier (N_j). This is formally represented as:

$$n_{\tau,j}^{ND} + n_{\tau,j}^{D,(+)} \leq N_j, \forall \tau, j \quad (23)$$

Nevertheless, $n_{\tau,j}^{D,(+)}$ can be less than $N_j - n_{\tau,j}^{ND}$ if some particular task(s) need(s) to be executed within a specific set of servers. This need may arise if the task(s) have additional security concerns or if there are data locality issues (special storage requirements).

In this optimization problem, $\vec{n}_\tau^D$ and $\vec{\lambda}_\tau^D$ are the *decision variables*. It is worth mentioning that both the objective function and the feasible set are *convex* in the *decision variables* and hence the optimization problem is *convex*. Therefore, efficient, reliable, polynomial time algorithms to solve it do exist [12]. (In case the original problem is not convex, it may still be possible to transform it into a convex optimization problem with acceptable relaxation to deal with the non-convexity in the optimization problem [12]. Since the above mentioned optimization problem is indeed convex in our case, therefore an optimal solution can be derived by expanding as per KKT (Karush–Kuhn–Tucker) conditions [12].

After determining $\vec{n}_\tau^D$ and $\vec{\lambda}_\tau^D$, we provision servers to the tasks following an Earliest Deadline First (EDF) scheduling policy, which is a classical scheduling policy and widely implemented in computer systems. It has been shown that, if the requirement of the resources does not exceed the service capacity of the system, then EDF meets the respective deadlines of all the tasks [13].

D. Minimization of operational cost only (OpexMin Procedure)

In this case the electricity cost remains the same as *CapexOpexMin*, which is given by:

$$C_\tau^E = p_\tau \{P_\tau^{Tot} - r_\tau\}^+ \quad (24)$$

There has been substantial amount of previous works showing that the cost for switching the server from OFF to ON state is dominant as compared to ON to OFF transition [6]. The switching cost for one timeslot (C_τ^S) is:

$$C_\tau^S = \vec{1}^T (\vec{n}_\tau - \vec{n}_{\tau-1})^+ \quad (25)$$

We formulate and solve the following optimization problem:

$$\text{Minimize} \quad \sum_{\tau=1}^T C_\tau^E + \delta \sum_{\tau=1}^T C_\tau^S \quad (26)$$

subject to the same set of constraints as *CapexOpexMin*, as given from (8) to (23). To control the relative priority between power cost and switching cost we introduce a weighting factor δ . The algorithm for solving the above problem and provisioning the servers at different tiers is referred to as *OpexMin*. This optimization problem is also convex and is therefore also computationally tractable.

III. SIMULATION SETUP AND RESULTS

A. Model parameters

In this Section we present the details of the simulation setup and the parameter choices. We implemented the algorithms, *CapexOpexMin* and *OpexMin* in CVX [14], which is a freely available modeling package for solving disciplined convex optimization problems. At each timeslot, *CapexOpexMin* takes approximately 3 seconds and *OpexMin* takes approximately 2 seconds to converge in MATLAB 7.11.0.

The duration of each time slot is 10 min. The length of the prediction horizon is 24 timeslots and hence the window length is 4 hours. It is well established that renewable energy and jobs can be predicted a couple of hours ahead with an acceptable precision [15, 16]. Electricity price is obtained from [17], which is the real time electricity price in Singapore. The job traces are obtained from a commercial server cluster owned by Facebook, which is the same trace as in [18]. We consider a data center with a total of 120,000 servers, distributed in three tiers of 10%, 40% and 50% servers (of the total servers in the data center), respectively.

We take, $\alpha_0 = 0.6$ and $\alpha_1 = 0.4$, since typically a commercial server consumes 60% of its maximum power and that this grows almost linearly with the increasing load [19]. We choose $\delta = 6$, $\beta = 1.2 \times 10^{-4}$ and $\gamma = 100$. The work in [6] proposed that the cost of one OFF to ON transition for one server is equivalent to powering a server for one hour, which translates to 6 timeslots in our model. We explored different values of β from 10^{-4} to 0.1 and found that a number of the order of 10^{-2} gives a good trade-off between electricity cost reduction, renewable usage and the number of switchings. For γ , which determines the weight of peak server reduction (7), 100 happens to be a good choice for striking a reasonable balance between electricity cost and the number of switchings.

B. Comparison between CapexOpexMin, OpexMin and their respective oracle counterparts

We present here the performance of *CapexOpexMin* and *OpexMin* with respect to the three important parameters - Cumulative Electricity Cost (CEC), Cumulative Renewable Consumption (CRC) and Cumulative Number of Switchings (CNS) performed. Our goal is to get insightful understanding

on the effect of careful formulation of the optimization problem. We observe in Fig. 1 that *CapexOpexMin* achieves 13.8% peak (in number of servers) reduction as compared to *OpexMin*. As per empirical calculation, this would translate into a capital cost reduction of US\$170 million to US\$340 million, if the data center consists of 120,000 servers [8]. We then compare and analyze *OpexMin* and *CapexOpexMin* with their respective *oracle optimum*. The *oracle optimum* is the one with perfect knowledge of the future realizations of all the model parameters. Obviously, the performance of *oracle optimum* will be better than its MPC counterpart since the latter uses the knowledge of the model parameters only over the window length. We conclude that *CapexOpexMin* is fairly conservative in switching servers from ON to OFF and therefore *OpexMin* can provide a superior match with the renewables by performing more frequent switching. Fig. 1 presents the number of servers under the two algorithms *OpexMin* and *CapexOpexMin* and their *oracle* counterparts. In Fig.2 we observe that for CEC there is approximately 20.6% saving for *CapexOpexMin* over *OpexMin*. However the Capex and cumulative electricity cost reductions are achieved at the cost of less renewable integration. In Fig.2, we observe that the CEC incurred by *OpexMin* does not increase between timeslots 222 to 250 (for both MPC and *oracle* cases). This happens because the comparison is under the condition that the same amount of jobs is executed for the different algorithms and *OpexMin* schedules and executes deferrable jobs earlier as compared to *CapexOpexMin*. It turns out that after 222 timeslots, the brown energy consumed by *OpexMin* is practically zero (only green energy is used). In terms of CRC we see *CapexOpexMin* harnesses 11.9% less green energy than *OpexMin* (Fig. 3). In this case we observe that CRC is continuously increasing under both the algorithms. In terms of CNS (measured by the number of times a server turned from OFF to ON), the performances of these two algorithms are dramatically different. We observe that CNS under *CapexOpexMin* is only 30.1% (close to one third) compared to *OpexMin* (Fig. 4). The most important fact is that, with respect to *all the three parameters* considered here, *CapexOpexMin* is *less sensitive to window size* than *OpexMin*. This indicates that *CapexOpexMin* provides more robustness against forecast or modeling errors. We find that the CEC under *OpexMin Oracle* is 34% less than that of its MPC variant. In contrast, for *CapexOpexMin*, MPC incurs about 3% less CEC than its *oracle* counterpart. The primary reason for this is that *CapexOpexMin* is more conservative in switching and its *oracle* counterpart achieves 56% improvement in CNS as compared to its MPC counterpart. For *OpexMin*, increasing the window size dramatically improves the switching performance which is even superior than *Oracle CapexOpexMin*. As shown in Fig. 4, the *Oracle OpexMin* performs only 6.7% CNS as compared to its MPC counterpart. This again confirms that the *OpexMin* algorithm is quite sensitive to the size of the prediction horizon. We observe that CNS keeps on increasing almost throughout and there is no prominent sluggishness (Fig. 4). In summary, we observe that in this setting, *CapexOpexMin* gives a

reasonable balance between electricity cost, renewable integration and switching and is *robust* on model parameters. We believe our findings provide crucial insights both on the importance of carefully formulating the optimization problem and the information that can be obtained from it for efficiently operating the system over time. The results are summarized in TABLE II.

Fig. 1. Comparison of no. of servers provisioned under different schemes

Fig. 2. Comparison of cumulative electricity cost under different schemes

TABLE II
SUMMARY OF RESULTS IN PERCENTAGE

	CEC	CRC	CNS
<i>CapexOpexMin</i> over <i>OpexMin</i>	21% (↓)	12% (↓)	70% (↓)
<i>OpexMin Oracle</i> over <i>MPC</i>	34% (↓)	2.17% (↑)	93.3% (↓)
<i>CapexOpexMin Oracle</i> over <i>MPC</i>	-3% (↓)	12.7% (↑)	56.3% (↓)

Fig. 3. Comparison of cumulative green energy consumption under different schemes

Fig. 4. Comparison of cumulative no. of switchings under different schemes

IV. CONCLUSIONS

In this paper we focused on energy aware scheduling of deferrable jobs and considered the issue of determining the optimal strategy for server provisioning in data centers to reduce the capital and operational cost and to increase renewable energy usage by efficient scheduling of deferrable jobs. We proposed and compared two scheduling strategies for the deferrable jobs through the MPC based optimal control method. The primary objective is to reduce peak server provisioning cost (Capex) in addition to reducing the operational cost (Opex). Importantly, we considered the switching cost, for toggling a server in the resource scheduling problem, which makes the problem more challenging. We formulated and solved convex optimization problems to tackle these issues. Since, convex optimization problems are

computationally tractable, our proposed methods are scalable even if the problem size is quite large and this will facilitate the implementation of the proposed algorithms in the actual data centers. We also investigated the sensitivity of our algorithms with respect to the size of the prediction horizon. We have observed that the algorithm for minimizing the combination of Opex and Capex performs better than the other algorithm.

REFERENCES

- [1] G. James. (2011). *Google Details, and Defends, Its Use of Electricity*. Available: http://www.nytimes.com/2011/09/09/technology/google-details-and-defends-its-use-of-electricity.html?_r=0
- [2] A. Qureshi, R. Weber, H. Balakrishnan, J. Gutttag, and B. Maggs, "Cutting the electric bill for internet-scale systems," in *Proc. ACM SIGCOMM Computer Communication Review*, vol. 39, pp. 123-134, 2009.
- [3] Z. Liu, M. Lin, A. Wierman, S. H. Low, and L. L. Andrew, "Greening geographical load balancing," in *Proc. ACM SIGMETRICS* pp. 233-244, 2011.
- [4] C. Gary, "Clicking Clean: How Companies are Creating the Green Internet," 2014.
- [5] A. Krioukov, C. Goebel, S. Alspaugh, Y. Chen, D. E. Culler, and R. H. Katz, "Integrating renewable energy using data analytics systems: challenges and opportunities," *IEEE Data Eng. Bull.*, vol. 34, pp. 3-11, 2011.
- [6] M. Lin, A. Wierman, L. L. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," *IEEE/ACM Transactions on Networking (TON)*, vol. 21, pp. 1378-1391, 2013.
- [7] S. Albers, "Energy-efficient algorithms," *Communications of the ACM*, vol. 53, pp. 86-96, 2010.
- [8] L. A. Barroso and U. Hölzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines," *Synthesis lectures on computer architecture*, vol. 4, pp. 1-108, 2009.
- [9] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, "An analytical model for multi-tier internet services and its applications," in *Proc. ACM SIGMETRICS* vol. 33, pp. 291-302, 2005.
- [10] E. F. Camacho and C. B. Alba, *Model predictive control*: Springer, 2013.
- [11] A. Subramanian, M. J. Garcia, D. S. Callaway, K. Poolla, and P. Varaiya, "Real-time scheduling of distributed resources," *IEEE Trans. Smart Grid*, vol. 4, pp. 2122-2130, 2013.
- [12] S. Boyd and L. Vandenberghe, *Convex optimization*: Cambridge university press, 2009.
- [13] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM (JACM)*, vol. 20, pp. 46-61, 1973.
- [14] M. Grant, S. Boyd, and Y. Ye. (2014). *CVX: Matlab software for disciplined convex programming (2.1 ed.)*. Available: <http://cvxr.com/cvx/>
- [15] P. P. Varaiya, F. F. Wu, and J. W. Bialek, "Smart operation of smart grid: Risk-limiting dispatch," *Proceedings of the IEEE*, vol. 99, pp. 40-57, 2011.
- [16] N. Singh and S. Rao, "Ensemble learning for large-scale workload prediction," *IEEE Trans. Emerging Topics in Computing*, vol. 2, pp. 149 - 165, 2014.
- [17] *Energy Market Company(EMC), Singapore*. Available: <https://www.emcsg.com/MarketData/PriceInformation>
- [18] Z. Liu, I. Liu, S. Low, and A. Wierman, "Pricing data center demand response," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, pp. 111-123, 2014.
- [19] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *IEEE computer*, vol. 40, pp. 33-37, 2007.