Energy Efficient Geographical Load Balancing via Dynamic Deferral of Workload

Muhammad Abdullah Adnan*, Ryo Sugihara† and Rajesh K. Gupta*
*University of California San Diego, CA, USA †Amazon.com

***Abstract*—With the increasing popularity of Cloud computing and Mobile computing, individuals, enterprises and research centers have started outsourcing their IT and computational needs to on-demand cloud services. Recently geographical load balancing techniques have been suggested for data centers hosting cloud computation in order to reduce energy cost by exploiting the electricity price differences across regions. However, these algorithms do not draw distinction among diverse requirements for responsiveness across various workloads. In this paper, we use the flexibility from the Service Level Agreements (SLAs) to differentiate among workloads under bounded latency requirements and propose a novel approach for cost savings for geographical load balancing. We investigate how much workload to be executed in each data center and how much workload to be delayed and migrated to other data centers for energy saving while meeting deadlines. We present an offline formulation for geographical load balancing problem with dynamic deferral and give online algorithms to determine the assignment of workload to the data centers and the migration of workload between data centers in order to adapt with dynamic electricity price changes. We compare our algorithms with the greedy approach and show that significant cost savings can be achieved by migration of workload and dynamic deferral with future electricity price prediction. We validate our algorithms on MapReduce traces and show that geographic load balancing with dynamic deferral can provide 20-30% cost-savings.**

I. Introduction

Increasing energy prices and ability to dynamically track these price variations due to enhancements of the electrical grid raise the possibility of utilizing "cloud computing" for energy efficient computing. Energy efficiency in the cloud has been explored recently in [1], [2], [3], [4], [5]. While these explorations have suggested a number of hardware and software techniques for energy-savings considering different aspects, one non-conventional perspective is to utilize the predetermined service level agreements (SLAs) for energy efficiency. Often the specification of SLAs contains some flexibility which could be exploited to improve the performance and efficiency [6], [7]. One of the important performance metric for cloud-based services is latency in which service providers get a lot of interest. This paper leverages the latency requirements for energy efficient computing in the cloud.

Naturally, energy efficiency in the cloud has been pursued in various ways including the use of renewable energy [8], [9] and improved scheduling algorithms [2], [5], [10], etc. Among them, improved scheduling algorithm is a promising approach for its broad applicability regardless of hardware configurations. The idea of utilizing SLA information to improve performance and efficiency is not entirely new. Recent work explores utilization of application deadline information for improving the performance of the applications (e.g. see [6], [7]). But the opportunities for energy efficiency remain unexplored. In this paper, we utilize the flexibility from the Service Level Agreements (SLAs) for different types of workload to reduce energy consumption.

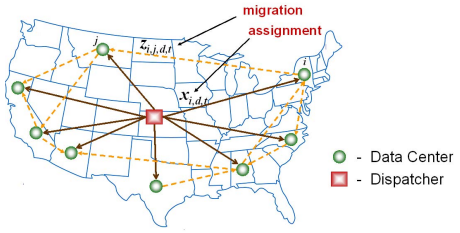


Fig. 1. Geographical Load Balancing.

We consider the problem of geographical load balancing in the cloud (Figure 1). In cloud computing each center of execution (data centers) are usually located in different geographic locations which are often in different time zones. Due to the increase in cost of energy, the electric billing companies have different pricing rates for electricity at different locations and at different times of the day. Hence load balancing decisions should take into account the current time zones and locations of data centers during task assignment for minimizing the total cost of energy consumption in the cloud. In this paper, we investigate and analyze how the pricing of energy at different times of the day along with 'task migration' account for the decisions for task assignment and deferral in the cloud. We use deadline information to defer some tasks so that we can reduce the total cost for energy consumption for executing the workload depending on time and location.

The contribution of this paper is twofold. First, we present a simple but general model for geographical load balancing and provide an offline formulation for solving the problem with deadline requirements. For each time slot, the formulation determines the assignment of workload to data centers and the migration of workload between data centers to adapt with the dynamic electricity price variation.

Second, we design an online algorithm for geographical load balancing considering migration and prediction error. The algorithm uses migration to improve the performance in case of prediction errors. We show that no online algorithm has constant competitive ratio with respect to the offline algorithm because of uncertainty in the electricity price variation. This allows us to compare our online algorithm with a simpler online algorithm without migration and prediction error to determine a bound on the cost based on the prediction error. We then prove that an online algorithm with migration gives better cost savings than the online algorithm without migration with future electricity price prediction. We validated our model by experiments using MapReduce traces as dynamic workload and found 20-30% total cost savings.

The rest of the paper is organized as follows. Section II presents the model that we use to formulate the optimization

978-0-7695-4755-8 2012

DOI 10.1109/CLOUD.2012.45

and gives the offline formulation. In Section III, we present the online algorithm for determining workload assignment and migration dynamically for uniform and nonuniform deadline. Section IV shows the experimental results. In Section V, we describe the state of the art research related to geographical load balancing and Section VI concludes the paper.

II. Model Formulation

In this section, we describe the model we use for geographical load balancing via dynamic deferral. The assumptions used in this model are minimal and this formulation captures many properties of current geographical load balancing and workload characteristics.

A. Workload Model

We consider a workload model where the total workload varies over time. The time interval we are interested in is $t \in \{0, 1, \ldots, T\}$ where T can be arbitrarily large. In practice, T can be a year and the length of a time slot τ could be as small as milliseconds for service requests (e.g. HTTP) or as large as several minutes for batch-like jobs (e.g. MapReduce). Let L_t be the amount of workload released at time slot t. The workload L_t can contain short jobs and long jobs. If the length ℓ of a job is greater than time slot length τ then we decompose the job into small pieces ($\leq \tau$) each of which is released after the execution of the preceding piece. Thus long jobs are decomposed into small jobs. Hence we do not distinguish each job, rather deal with the total amount of workload.

In our model, each job has a deadline D (in terms of number of slots) associated with it, where D is a nonnegative integer. A job released at time t, needs to be executed within time slot $t + D$. The value of D can be zero for interactive jobs and large for batch-like jobs. If the job is large and decomposed into smaller pieces, then we need to assign deadline to each individual piece. If the long job is preemptive then we assign deadline $\lfloor D/\ell \rfloor - 1$ to each of the small pieces and for a non-preemptive job, we assign deadline of $D - \ell$ to the first piece and deadlines of zeros to the other pieces. In this section, we consider the case of uniform deadlines, that is, deadline is uniform for all the jobs, followed by non-uniform deadline case in Section IIIE. Since the deadline D is uniform for all the jobs, the total amount of work L_t must be executed by the end of time slot $t + D$.

We consider a large computing facility ("cloud"), consisting of n data centers. At each time t, the total workload L_t arrive at a central dispatcher from which load balancing decisions are made. We assume that the arriving workload cannot be stored at the dispatcher, i.e., the workload arriving at the beginning of time t needs to be dispatched to the data centers after the assignment of workload for each data center is determined. After the load balancing decisions are made at the dispatcher, the jobs can be stored at each data center to be executed at a suitable time before deadline. We are not concerned about the computation capability (homogeneous/heterogeneous) inside each data center rather we focus on load distribution considering data centers as computation units. The total computation capacity M_i in data center i is fixed and given for $1 \leq i \leq n$. We normalize L_t by the processing capability of the data centers i.e. L_t denotes the computation units required to execute the workload at time t.

Let $x_{i,d,t}$ be the portion of the released workload L_t that is assigned to be executed at data center i at time slot $t + d$. Let $x_{i,t}$ be the total workload assigned to be executed at time t to data center i and x_t be the total assignment at time t. Then $0 \leq x_{i,t} \leq M_i$ and

$$\sum_{d=0}^{D} x_{i,d,t-d} = x_{i,t} \text{ and } \sum_{i=1}^{n} x_{i,t} = x_t$$

We assume that the energy prices vary unpredictably depending on time and location. The energy (electricity) costs may vary in time, yet remain fixed within time slot length τ. The workload assigned to one data center can be migrated to other data centers in order to reduce the total energy consumption. Let $z_{i,j,d,t}$ be the amount of workload that is migrated at time t from data center i to be executed at data center j at time $t + d$. Then $z_{i,j,t} = \sum_{d=0}^{D} z_{i,j,d,t}$, is the total amount of workload that is migrated form data center i to j at time t. We assume that there is a cost associated with each migration and all migrations are done as soon as the migration decisions are made. Often there are high bandwidth links between data centers. Hence we can assume that migration time is negligible with regard to the time interval τ i.e. migration does not incur any delay. For service requests, τ is small as well as migration time is negligible and for batch-like jobs, τ is large in the range of minutes and migration time is in the range of seconds. Therefore with respect to τ, migration time is small (negligible).

Since some portion of the assigned workload is migrated to other data centers, the workload that is executed at time t at data center i is the sum of the assigned workload and the net migrated-in workload as denoted by

$$y_{i,t} = x_{i,t} + \sum_{j=1}^{n} \sum_{d=0}^{D} z_{j,i,d,t-d} - \sum_{j=1}^{n} \sum_{d=0}^{D} z_{i,j,d,t-d} \quad (1)$$

Since the released workload within $[1, T]$ needs to be finished within T time slots, the total assignment and execution is equal to the total released workload over T time slots as given by the following equation.

$$\sum_{t=1}^{T} \sum_{i=1}^{n} y_{i,t} = \sum_{t=1}^{T} \sum_{i=1}^{n} x_{i,t} = \sum_{t=1}^{T} L_t \quad (2)$$

Thus there are two important decisions here: (i) determining $x_{i,d,t}$, assignment of workload to the data centers, and (ii) determining $z_{i,j,d,t}$, the amount of migrated workload during each time slot t.

B. Cost Model

The goal of this paper is to minimize the operation cost in the cloud which is the sum of the energy costs for executing workload at the data centers and the costs for migrating jobs between data centers.

Energy cost: To capture the geographic diversity and variation of energy costs over time, we let $C_{i,t}(y_{i,t})$ denote the *energy cost* for executing workload $y_{i,t}$ in data center i at time slot t. We assume that $C_{i,t}(y_{i,t})$ is a nonnegative, (weakly) convex increasing function as used in [5]. Note that the function itself can change with time, which allows for time variation in energy prices. The simplest example for the cost function for a time slot is an affine function which is the common model for the energy cost for typical data centers:

$$C_{i,t}(y_{i,t}) = \alpha_i + \beta_{i,t} y_{i,t}$$

where α_i and $\beta_{i,t}$ are constants for data center i and time slot t (e.g. see [2]) and $y_{i,t}$ is the executed workload to data center i at time t. Note that in this model, the load dependent component of the function only depends on time, because the real time electricity price varies with real time demand.

Migration cost: The *migration cost* is the cost for migrating workload from one data center to another which accounts for bandwidth cost and energy consumed by the intermediate devices. The migration cost is proportional to the amount of migrated workload which is represented by,

$$B_{i,j}(z_{i,j,t}) = b_{i,j} z_{i,j,t}$$

where $b_{i,j}$ is constant for migration from data center i to j regardless of the migration time [5]. Note that there are also bandwidth costs associated with the arrival of jobs into the cloud (e.g. from the central dispatcher) and leaving the cloud (in case of job departure). However these costs are constant and do not depend on the migration control, and thus can be easily incorporated without changing the problem formulation.

C. Optimization Problem

Given the models above, the goal of geographical load balancing is to choose the migrating jobs $z_{i,j,d,t}$ and the dispatching rule $x_{i,d,t}$ to minimize the total cost during $[1, T]$, which is captured by optimization (3). In this formulation, constraint (3b) represents that the total assignment should be equal to the total released workload and constraint (3c) represents that the total workload that is migrated from data center i to be executed at time t cannot exceed the assigned workload to be executed at time t at data center i. Note that by constraint (3c), we ensure that the amount of work done in a data center is at least the net migrated-in work. We now prove that there is an optimal solution of optimization (3) where there is no migration. We have the following lemma.

$$\min_{x_t, z_t} \sum_{t=1}^{T} \sum_{i=1}^{n} C_{i,t}(y_{i,t}) + \sum_{t=1}^{T} \sum_{i=1}^{n} \sum_{j=1}^{n} b_{i,j} z_{i,j,t} \tag{3a}$$

$$\text{s.t.} \sum_{i=1}^{n} \sum_{d=0}^{D} x_{i,d,t} = L_t \quad \forall t \tag{3b}$$

$$\sum_{k=0}^{D-d} \sum_{j=1}^{n} z_{i,j,d+k,t-k} \leq \sum_{k=0}^{D-d} x_{i,d+k,t-k}, \ \forall i, \forall d, \forall t \tag{3c}$$

$$y_{i,t} \leq M_i \qquad \forall i, \forall t \tag{3d}$$

$$x_{i,d,t} \geq 0, z_{i,j,d,t} \geq 0 \qquad \forall i, \forall j, \forall d, \forall t. \tag{3e}$$

Lemma 1: In every optimal solution of optimization (3), either $z_{i,j,d,t} = 0$, for all time slots t, and deferral d or $b_{i,j} = 0$, for all (i, j).

Proof: Suppose for a contradiction that the optimal solution O contains $z_{i,j,d,t} > 0$, and $b_{i,j} > 0$, for some (i, j). Then we can construct another optimal solution O' where $z'_{i,j,d,t} = 0$, $\forall i, j, d, t$ by making $x'_{i,t} = y_{i,t}$. Then $y'_{i,t} = y_{i,t}$, $\forall i, t$ and $\sum_{t=1}^{T} \sum_{i=1}^{n} C_{i,t}(y_{i,t}) = \sum_{t=1}^{T} \sum_{i=1}^{n} C_{i,t}(y'_{i,t})$. The objective value of the solutions $obj(O') < obj(O)$ because $\sum_{t=1}^{T} \sum_{i=1}^{n} \sum_{j=1}^{n} b_{i,j} z'_{i,j,t} = 0 < \sum_{t=1}^{T} \sum_{i=1}^{n} \sum_{j=1}^{n} b_{i,j} z_{i,j,t}$ since $b_{i,j} > 0$. This contradicts the assumption that O is an optimal solution. ■

Corollary 2: There exists an optimal solution of optimization (3) where $z_{i,j,d,t} = 0$, $\forall i, j, d, t$.

Hence migration is unnecessary when all the information about workload and energy price are known in advance. However, when all the information are not available then migration becomes important as investigated in the next section.

Since the operating cost $C_{i,t}(\cdot)$ is an affine function, the objective function is linear as well as the constraints. Hence it is clear that the optimization (3) is a linear program. Note that the workload $x_{i,t}$ in the formulation is not considered to be integer. This is acceptable because the number of requests in data centers at each time slot is in the range of thousands and we can round the resulting assignment with minimal increase in cost. If all the future costs and workload were known in advance, then the problem could be optimally solved as a linear program. However our basic assumption is that electricity prices change in an unpredicted manner depending on time and location. Therefore, we tackle the optimization problem as an online optimization problem.

III. Online Algorithm

In this section we consider the online case, where at any time t, we neither have information about the future workload $L_{t'}$ for $t' > t$, nor have knowledge about future electricity prices. The workload released at time t can be delayed to be executed in future time slots if the cost for execution at future time slots is less than the current cost. We apply optimization on the current and delayed workload and distribute them in future time slots so that the total cost for execution and migration is minimized subject to the future predicted prices. In the online algorithm, we decouple the migration decision from the assignment decision and apply optimization in two levels: (i) dispatcher level and (ii) data center level. The dispatcher makes decision about the assignment of the incoming workload to data centers based on predicted future electricity prices. Then the data centers make decision on adjusting the execution of the workload in current and future time slots and the migration of workload between data centers in case of prediction errors.

A. Electricity Price Prediction Model

In this section we illustrate our model for predicting future price of electricity. Since only the load proportional component of energy cost depends on time, we need to predict $\tilde{\beta}_{i,t}$. Then the predicted energy cost function will be

$$\tilde{C}_{i,t}(y_{i,t}) = \alpha_i + \tilde{\beta}_{i,t} y_{i,t}$$

The load dependent electricity price $\beta_{i,t}$ is announced by the utility at location i at the beginning of each time slot t and is kept constant during the duration of that time slot. However future prices will change independently of past prices according to some known probability density function. Predicting electricity prices is difficult because price series present such characteristics as nonconstant mean and variance and significant outliers. We model the prediction noise by a Gaussian random variable with zero mean and variance to be estimated. In other words, we model future prices within a 24-hour time-frame by Gaussian random variables with known means, which are the predicted prices, and some estimated variance. The mean for the Gaussian distribution is predicted by the widely used moving average method for time series. The variance for the Gaussian distribution is estimated from the history by the weighted average price prediction filter proposed in [11]. In this model, variances are predicted by

linear regression from the previous prices from yesterday, the day before yesterday and the same day last week. By using two different methods for mean and variance, we exploit both the temporal and historical correlation of electricity prices. To facilitate the future price prediction, we denote the set of the time slots in a 24-Hour time frame by $\mathcal{K} \subset T$. Let $\tilde{\mu}_i^{\kappa}[\chi]$ and $\tilde{\sigma}_i^{\kappa}[\chi]$ be the predicted means and standard deviations for each time slot κ on day χ for geographical location i. Then the mean of the prediction model for Gaussian distribution is obtained as follows:

$$\tilde{\mu}_i^{\kappa} = \varepsilon_0 + \sum_{j=0}^{D} \varepsilon_{\kappa-j}\beta_{i,\kappa-j}, \quad \forall i \in n, \forall \kappa \in \mathcal{K}$$

Here, ε_j are the coefficients for the moving average method which can be estimated by training the model over the previous day prices. The variance parameter $\tilde{\sigma}_i^{\kappa}[\chi]$ is estimated from the history using the following equation:

$$\tilde{\sigma}_i^{\kappa}[\chi] = k_1\sigma_i^{\kappa}[\chi-1] + k_2\sigma_i^{\kappa}[\chi-2] + k_7\sigma_i^{\kappa}[\chi-7], \quad \forall i \in n, \forall \kappa \in \mathcal{K}$$

Here, $\sigma_i^{\kappa}[\chi-1]$, $\sigma_i^{\kappa}[\chi-2]$ and $\sigma_i^{\kappa}[\chi-7]$ denote the previous standard deviation values σ_i^{κ} on yesterday, the day before yesterday and the same day last week, respectively. The coefficients for the weighted average price prediction filter k_1, k_2 and k_7 are selected from [11].

B. Optimization for Dispatcher

The dispatcher makes decision on the assignment of the workload to the data centers based on the current electricity prices and future price predictions. The following optimization applied at the dispatcher determines the assignment of workload $x_{i,d,t}$ to data centers for $0 \le d \le D$ and $1 \le i \le n$.

$$\min_{x_{i,d,t}} \quad \sum_{i=1}^{n}\sum_{d=0}^{D} C_{i,t}(x_{i,d,t-d}) + \sum_{i=1}^{n}\sum_{k=1}^{D}\sum_{d=k}^{D} \tilde{C}_{i,t+k}(x_{i,d,t+k-d}) \tag{4a}$$

$$\text{subj. to} \quad \sum_{i=1}^{n}\sum_{d=0}^{D} x_{i,d,t} = L_t \tag{4b}$$

$$0 \le \sum_{d=s-t}^{D} x_{i,d,s-d} \le M_i \quad \forall i, t \le s \le t+D. \tag{4c}$$

where $\tilde{C}_{i,t'}()$ is the predicted cost function at time $t' > t$ for data center i and $x_{i,d,t''}$ is the unexecuted workload at data center i that was assigned at time $t'' < t$ to be executed at time $t'' + d$ where $t - t'' \le d \le D$. Note that greedy method can also be applied to compute the optimum assignment for the dispatcher.

C. Optimization for Data Centers

The predicted electricity prices at time t may contain prediction errors which may lead to some badness in the assignment. Data centers can migrate workload between each other to adjust the assignment to overcome prediction errors for minimizing the total cost in the later time slots. The adjustment is made by applying an optimization on the schedule for the unexecuted workload for the current and future time slots. For each data center i, this optimization makes decision on how much workload to execute at time t, how much to defer to execute later and how much to migrate to other data centers. Note that the workload released at or before t, cannot be delayed to be assigned after time slot $t + D$. Hence we minimize the total cost by applying optimization on the already released but unexecuted (delayed) workload over the interval $[t, t + D]$. We have two versions of the online optimization at data centers. First we formulate the optimization without considering migration. Then the more general case with migration is considered.

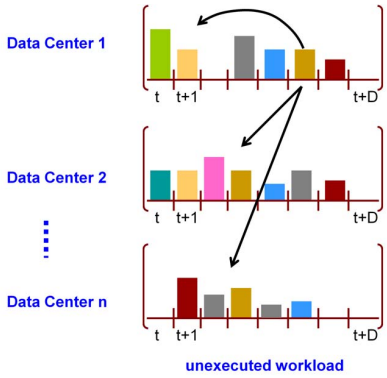


Fig. 2. Optimization on unexecuted (assigned) workload at time t by transferring workload to previous time slots (curved arrow) and by migration to other data centers (straight arrow).

1) Formulation without Migration: We start with a formulation for the online case by considering load balancing without migration. Although there is no migration, still the data centers can improve the assignment by executing the delayed workload early in previous time slots without violating deadline as shown by the curved arrow in Figure 2. Let $u_{i,t}$ and $w_{i,t}$ denote the assigned (delayed) and executed workload at time t at data center i, respectively. Initially $u_{i,t} = w_{i,t} = 0$, $\forall t$. Then the values of $w_{i,s}$ for $t \le s \le t + D$ are obtained at each time t by applying the optimization (5). And the values of $u_{i,s}$ for $t \le s \le t + D$ are updated each time t from the computed $w_{i,s}$. The following optimization determines the current and future execution variables $w_{i,s}$ for $t \le s \le t+D$.

$$\min_{w_{i,t}} \quad \sum_{i=1}^{n} C_{i,t}(w_{i,t}) + \sum_{i=1}^{n}\sum_{s=t+1}^{t+D} \tilde{C}_{i,s}(w_{i,s}) \tag{5a}$$

$$\text{subj. to} \quad \sum_{s=t}^{t+D} w_{i,s} = \sum_{s=t}^{t+D} u_{i,s} \qquad \forall i \tag{5b}$$

$$\sum_{r=t}^{s} w_{i,r} \ge \sum_{r=t}^{s} u_{i,r} \qquad \forall i, t \le s < t+D \tag{5c}$$

$$0 \le w_{i,s} \le M_i \qquad \forall i, t \le s \le t+D. \tag{5d}$$

Here the constraints (5b) and (5c) ensure that the assigned workload $u_{i,s}$ can only be moved to an earlier time slot $t \le t' \le s$ and thus does not violate deadline.

2) Formulation with Migration: We can utilize the migration of workload between data centers to correct the prediction errors (to some extent) made by the dispatcher at an early slot during dispatching. Let $z_{i,j,d,t}$ denote the migration of

workload from data center i at time t which will be executed at data center j at time $t+d$. Then the values of $u_{i,s}$ for $t \le s \le t+D$ are updated each time t from $w_{i,t}$, $z_{i,j,d,t}$ and $z_{j,i,d,t}$'s with the rule in equation (6).

$$u_{i,s} = \begin{cases} x_{i,D,t} & \text{if } s = t+D, \\ w_{i,s} + x_{i,s-t,t} + \sum_{j=1}^{n} z_{j,i,s-t+1,t-1} & \\ \quad - \sum_{j=1}^{n} z_{i,j,s-t+1,t-1} & \text{if } t \le s < t+D. \end{cases} \tag{6}$$

Then applying the optimization (7), the values for $w_{i,s}$ and $z_{i,j,d,s-d}$ are determined for $t \le s \le t+D$. Then the workload that is executed including the migration at data center i at time t is $y_{i,s}$, for $t \le s \le t+D$, which is determined by the following equation

$$y_{i,s} = w_{i,s} + \sum_{j=1}^{n}\sum_{d=0}^{D} z_{j,i,d,s-d} - \sum_{j=1}^{n}\sum_{d=0}^{D} z_{i,j,d,s-d}$$

The optimization (7) applied at time t determines the current and future execution variable $w_{i,s}$ for $t \le s \le t+D$ and the current migration $z_{i,j,d,t}$ for $0 \le d \le D$.

$$\min_{w_{i,t}, z_{i,j,d,t}} \sum_{i=1}^{n} C_{i,t}(y_{i,t}) + \sum_{i=1}^{n}\sum_{s=t+1}^{t+D} \tilde{C}_{i,s}(y_{i,s}) + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{d=0}^{D} b_{i,j} z_{i,j,d,t} \tag{7a}$$

$$\text{subj. to} \quad \sum_{i=1}^{n}\sum_{s=t}^{t+D} y_{i,s} = \sum_{i=1}^{n}\sum_{s=t}^{t+D} u_{i,s} \tag{7b}$$

$$\sum_{i=1}^{n}\sum_{r=t}^{s} y_{i,r} \ge \sum_{i=1}^{n}\sum_{r=t}^{s} u_{i,r}, \ t \le s < t+D \tag{7c}$$

$$\sum_{j=1}^{n} z_{i,j,s-t,t} \le w_{i,s} \qquad \forall i, t \le s \le t+D \tag{7d}$$

$$y_{i,s} \le M_i, w_{i,s} \ge 0 \qquad \forall i, t \le s \le t+D \tag{7e}$$

$$z_{i,j,d,t} \ge 0 \qquad \forall i, \forall j, \forall d. \tag{7f}$$

Here the constraints (7b) and (7c) ensure that the assigned workload $u_{i,s}$ can be migrated to other data centers and can only move to an earlier time slot $t \le t' \le s$ and thus does not violate deadline as shown by arrows in Figure 2. Constraint (7d) ensures that the amount of migration does not exceed the unexecuted workload. Then the actual workload that is executed at time t at data center i is,

$$y_{i,t} = w_{i,t} + \sum_{j=1}^{n} z_{j,i,0,t} - \sum_{j=1}^{n} z_{i,j,0,t} \tag{8}$$

In summary, at the beginning of each time slot, we apply the optimization (4) at the dispatcher and then the tasks are assigned to the data centers. Then the optimization (7) for the data centers, is applied globally to determine the assignment and migration of previously released unexecuted workload. Then the migration takes place and the amount of execution for each data center is determined by equation (8). After that each of the data centers execute that amount of workload.

D. Analysis of the Algorithm

We now analyze the performance of the online algorithm. We first prove that there does not exist any online algorithm with constant competitive ratio with respect to the offline formulation (3).

Lemma 3: No online algorithm has constant competitive ratio with respect to the offline formulation (3).

Proof: Prediction error degrades the performance of the online algorithm, hence w.l.o.g. we assume that the online algorithm does not have any prediction error i.e. $\epsilon = 0$. We prove the claim by adversary method i.e. we consider an adversary who presents the online algorithm with several different instances. Suppose we have only one data center $n = 1$ with capacity M. The time slots are $\{0, 1, \ldots, T\}$. The uniform deadline is $D < T$. And the cost function parameters β_t are $\beta_0 = K \cdot \beta_D$ and $\beta_t \gg K \cdot \beta_D$ for $t \in \{0, 1, \ldots, T\} - \{0, D\}$. Now for determining the assignments $x_{d,t}$, we consider two cases:

Case 1: $\{x_{D,0} \neq 0\}$
In this case, suppose the workload released at time $t = 0$ and $t = 1$ are L_0 and L_1 respectively and $L_0 = L_1 = M$. Then the online assignment vector has $x_{d,1} > 0$ for some $0 \le d \ne D-1$ where $\beta_{d+1} \gg K \cdot \beta_D$ which can be arbitrarily large. Hence the competitive ratio becomes unbounded.

Case 2: $\{x_{D,0} = 0\}$
In this case, we construct an adversary input by making $L_0 = M$ and $L_1 = 0$. The offline algorithm chooses $x^*_{D,0} = L_0$ but the online algorithm chooses $x_{0,0} = L_0$. Since $\beta_0 = K.\beta_D$, the competitive ratio for this case is $\sim K$. Since K is arbitrary, the competitive ratio is not bounded by a constant. ■

Since the performance of any online algorithm cannot be bounded with respect to offline algorithm, we compare the online algorithm with a simple online algorithm without migration and without any prediction error. We call such an online algorithm as A and the online algorithm with migration (described in Section IIIC2) as A^m_ϵ. We denote the online algorithm with prediction error but without migration (described in Section IIIC1) as A_ϵ. Basically we are going to compare the performance of A and A^m_ϵ. We denote the total cost from an algorithm A as $cost(A)$. We have the following lemma.

Lemma 4: $cost(A^m_\epsilon) \le cost(A_\epsilon)$.

Proof: Let $y^m_{i,t}$ and $y_{i,t}$ be the workload executed at time t by algorithms A^m_ϵ and A_ϵ respectively. In the algorithm A^m_ϵ, the workload assigned to a time slot t can only move to earlier time slot $t' \le t$ as illustrated by constraints (7b) and (7c). Hence $\sum_{i=1}^{n} y_{i,t} \le \sum_{i=1}^{n} y^m_{i,t}$. Therefore $\Delta y = \sum_{s=t+1}^{t+D}\sum_{i=1}^{n} y^m_{i,s} - \sum_{s=t+1}^{t+D}\sum_{i=1}^{n} y_{i,s} \ge 0$. That means we have Δy more workload to execute in later slots $s > t$ for A_ϵ than A^m_ϵ and due to optimization at A^m_ϵ, $\sum_{i=1}^{n} C_{i,t}(\Delta y) + \sum_{i=1}^{n}\sum_{j=1}^{n} b_{i,j}(\Delta y) \le \sum_{i=1}^{n} \tilde{C}_{i,s}(\Delta y)$ for any $t+1 \le s \le t+D$. Since both the algorithms use same energy cost functions, we have $cost(A^m_\epsilon) \le cost(A_\epsilon)$. ■

According to Lemma 4, incorporating migration into the online algorithm reduces the total cost of execution than A_ϵ. Using this lemma, we now bound the cost for algorithm A^m_ϵ with respect to A by the prediction error ϵ as stated in the following theorem.

Theorem 5: $cost(A^m_\epsilon) \le (1+\epsilon) \cdot cost(A)$.

Proof: We first show that $cost(A_\epsilon) \le (1+\epsilon) \cdot cost(A)$. Then by lemma 4, the theorem holds. If there were no

prediction error i.e. $\epsilon = 0$, $cost(A_\epsilon) = cost(A)$. If there is a prediction error $\epsilon > 0$ then suppose the electricity price predicted for time t by A_ϵ is $\tilde{\beta}$, whereas the actual price used in A is β. Then $\tilde{\beta} - \epsilon \leq \beta \leq \tilde{\beta} + \epsilon$. Then $\frac{cost(A_\epsilon)}{cost(A)} = \frac{\alpha+\tilde{\beta}y}{\alpha+\beta y} \leq 1 + \frac{\epsilon}{\beta} \leq (1+\epsilon)$, where y is the executed workload. ■

Suppose the prediction error follows the Gaussian distribution with standard deviation σ. Then the probability that the prediction error is bounded by ϵ is given by the Chebyshev's inequality

$$Pr(|C - \tilde{C}| \geq \epsilon) \leq \frac{\sigma^2}{\epsilon^2}$$

By Theorem 5 the cost savings from the online algorithm depends on the price variation and the quality of prediction.

E. Nonuniform Deadline

The algorithm described above can be easily extended for nonuniform deadline where the deadline requirement is not same for all the workload. In this case the workload can be decomposed according to their associated deadline. Suppose $L_{d,t} \geq 0$ be the portion of the workload released at time t and has deadline d for $0 \leq d \leq D$ where D is the maximum deadline. Then we have $\sum_{d=0}^{D} L_{d,t} = L_t$. Then the constraints for L_t in the offline formulation (3b) and the online formulation (4b) can be replaced by the following constraint:

$$\sum_{i=1}^{n}\sum_{k=0}^{d} x_{i,k,t} = \sum_{k=0}^{d} L_{k,t}, \qquad 0 \leq d \leq D$$

Then the same algorithm can be applied to get solutions for nonuniform deadline.

IV. Experimental Results

In this section, we seek to evaluate the cost incurred by the algorithms A, A_ϵ and A_ϵ^m relative to the optimal solution in the context of workload generated from realistic data.

A. Experimental Setup

We aim to use realistic parameters in the experimental setup and provide conservative estimates of the cost savings resulting from optimal geographical load balancing.

Electricity Price: There are two types of electricity markets: Wholesale Market and Retail Market. Due to the high consumption of electricity in data centers, they usually purchase electricity from the wholesale markets [2]. Electricity price varies on a 5-minute or 15 minute basis in real time wholesale electricity market. Electricity price in this market exhibit significant volatility with high frequency variation [3].

We run our simulations for four data centers geographically located in four different locations. We choose distant locations for our experiments. We choose the locations near those power grids whose real time electricity prices are publicly available. We used the publicly available data from electricity markets from Independent System Operator New England (ISO-NE) [12], New York Independent System Operator (NYISO) [13], Electric Reliability Council of Texas (ERCOT) [14] and Electricity Market of New Zealand (NZ) [15]. We took the locational based marginal prices (LBMP) from the 5 minute spot markets for three days (15th, 14th and 8th February, 2012) and ran our experiments on the prices of 15th February using the prices for 14th and 8th for prediction of future

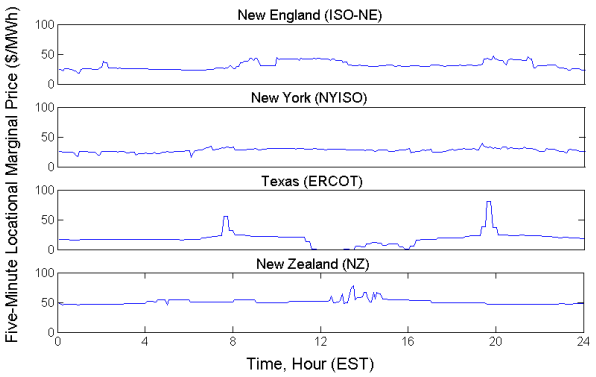
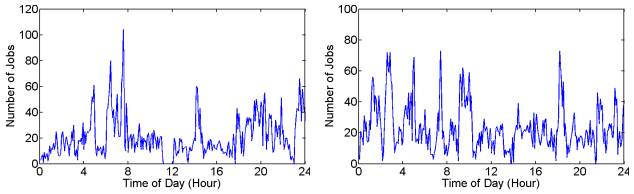


Fig. 3. Illustration of five minute locational marginal electricity prices in real time market on 15th February, 2012 for four different regions (a) New England (ISO-NE), (b) New York (NYISO), (c) Texas (ERCOT), (d) New Zealand (NZ).

prices. We use the four locations to have both temporal and geographical variation of electricity prices e.g. the time zones of New York, New England, Texas and New Zealand are GMT-5, GMT-7, GMT-6 and GMT+13 respectively. The variation of electricity prices for different locations are plotted in Figure 3 with Eastern Standard Time (EST). These graphs indicate significant spatio-temporal variation in electricity prices.

Workload Description: We use two publicly available MapReduce traces as examples of dynamic workload. The MapReduce traces were released by Chen et al. [16] which are produced from real Facebook traces for one day (24 hours) from a cluster of 600 machines. We count the number of different types of job submissions over a time slot length of 5 minutes and use that as a dynamic workload (Figure 4) for simulation. The two samples we use represent strong diurnal properties and have variation from typical workload (Workload A) to bursty workload (Workload B). We use time slot length of 5 minutes because the electricity prices vary with an interval of 5 minutes. In practice, load balancing decisions can be made more frequently with slot length size in the range of seconds. We then assign deadline for each job in terms of the number of slots the job can be delayed. For the case of uniform deadline, we vary deadline D from $1-12$ for the simulation. This is realistic because MapReduce workloads have deadlines in the range of minutes as deadlines from 8-30 minutes for these workloads have been used in the literature [17]. For the Non-uniform case, we use k-means clustering to classify the MapReduce workload into 10 groups based on the total sizes of map, shuffle and reduce bytes. The characteristics of each group are depicted in Table I where smaller jobs dominate the workload mix. This kind of clustering has been used by Chen et al. for classifying the workload. For each class of jobs we assign a deadline from $1-10$ slots such that smaller class (batch jobs) has larger deadline and larger class (interactive jobs) has smaller deadline.

Cost benchmark: Currently geographical load balancing for data centers typically does not use deferral of workload for load balancing [2], [5]. Often the load balancing decisions are made dynamically using greedy method based on current electricity prices without dynamic deferral. Clearly we could be energy efficient if we consider deferral of some of the tasks and use migration to adapt with the variation of electricity



(a) Workload A (b) Workload B

Fig. 4. Illustration of the traces for dynamic workload used in the experiments.

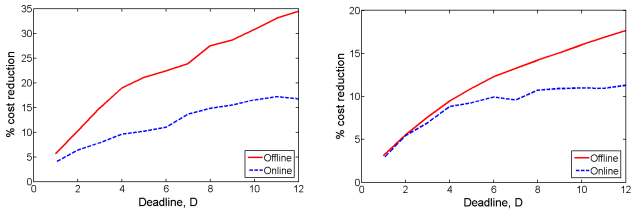
TABLE I
CLUSTER SIZES AND DEADLINES FOR WORKLOAD CLASSIFICATION BY K-MEANS CLUSTERING FOR NONUNIFORM DEADLINE

Cluster	Workload A		Workload B		Deadline
	#Jobs	GB	#Jobs	GB	#slots
1	4878	0.22	5632	0.32	1
2	496	3.13	513	5.85	2
3	196	9.90	170	18.62	3
4	113	23.49	100	39.09	4
5	80	49.59	106	56.52	5
6	49	85.07	44	99.23	6
7	48	146.67	26	160.90	7
8	19	286.36	29	350.62	8
9	13	620.01	11	659.30	9
10	2	8104.52	7	1294.19	10

prices. We compare the total cost from the offline and online (A_ϵ^m) algorithms with the greedy strategy as proposed by Qureshi et al. [3] and evaluate the cost reduction. We also compare the total cost for the online algorithms A_ϵ and A_ϵ^m.

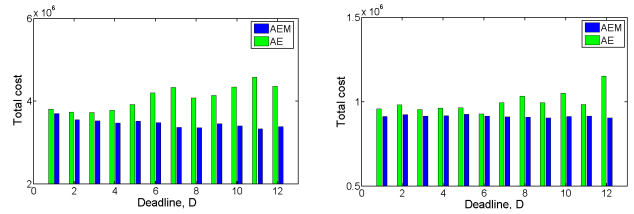
Cost function parameters: The cost function parameter $\beta_{i,t}$ is determined using current electricity price and $\tilde{\beta}_{i,t'}$, for $t' > t$, are determined using the electricity price prediction models described in Section IIIA. For our simulations, we use load independent parameter $\alpha_i = 0$, for all i. The values for $b_{i,j}$ are determined proportional to the geographic distance between data centers i and j. Since the workload cannot be migrated from source to source, we use $b_{i,i}$ to be a large number. Depending on the nature of the workload we varied the total capacity of the data centers because the algorithms keep on assigning the workload to the data center with the lowest cost until the data center is overloaded. Choosing a maximum capacity value to be less than the peak value allows us to visualize the cut off for the assignment. For both the workload, we use capacity $M_i = 50$ for all i.

The future electricity prices $\tilde{\beta}_{i,t}$ for the next D time slots are randomly generated from Gaussian Distributions because of their high unpredictability and the volume (D) of generation as described in Section IIIA. We use the same mean but different variances for the generation in each time slot. We use the optimal daily coefficients for the price prediction filter from [11] for estimating $\tilde{\sigma}_i^\kappa[\chi]$. Since we use the electricity prices for Wednesday (15th February, 2012), we choose $k_1 = 0.837$, $k_2 = 0$ and $k_7 = 0.142$. For the previous standard deviation values ($\sigma_i^\kappa[\chi-1]$, $\sigma_i^\kappa[\chi-7]$), we use the past standard deviation of electricity prices for D slots on those days such that $\sigma_i^\kappa[\chi-1] := std(\beta_{i,\kappa}[\chi-1], \beta_{i,\kappa-1}[\chi-1], \ldots, \beta_{i,\kappa-D}[\chi-1])$ and $\sigma_i^\kappa[\chi-7] := std(\beta_{i,\kappa}[\chi-7], \beta_{i,\kappa-1}[\chi-7], \ldots, \beta_{i,\kappa-D}[\chi-7])$; where $std(\cdot)$ denotes the standard deviation. The mean $\tilde{\mu}_i^\kappa[\chi]$ is computed from the moving average of the prices for D previous slots on the current day χ.



(a) Workload A (b) Workload B

Fig. 5. Impact of deadline on cost reduction by the offline and the online algorithm A_ϵ^m in comparison to greedy algorithm.



(a) Workload A (b) Workload B

Fig. 6. Comparison of cost incurred by the online algorithms A_ϵ^m, A_ϵ (AEM and AE in figure) for different deadlines.

B. Experimental Analysis

We now evaluate and analyze the cost savings provided by the offline and online algorithms.

Uniform Deadline: We compare the cost reduction for the offline and the online algorithm A_ϵ^m with the greedy method without dynamic deferral. The cost reduction for the online and offline algorithms for different deadlines are depicted in Figure 5. These curves show that dynamic deferral can provide around 30% cost savings for deadlines of 12 slots (1 hour) and even for one slot we can get ∼5% cost savings. Figure 6 illustrates the comparison of the total cost from algorithms A_ϵ and A_ϵ^m for different deadlines. From this figure, we can see that the total cost from the algorithm A_ϵ^m is always less than the total cost from the algorithm A_ϵ, as claimed in Lemma 4. As the deadline increases the total cost from the algorithm A_ϵ increases since the prediction error becomes significant for predicting more distant values (electricity prices) while the total cost from A_ϵ^m is reduced due to migration and the flexibility of dynamic deferral.

Nonuniform Deadline: We evaluate the cost savings for nonuniform deadline assigning different deadlines on the classification of the workload as illustrated in Table I. For conservative estimates of deadline requirements (1-10), we found 15.64% cost reduction for Workload A and 9.23% cost reduction for Workload B each of which remains close to the offline optimal solutions.

V. RELATED WORK

Greening data centers is becoming an increasingly important topic in operating cloud-scale data centers for two main reasons: (1) the global energy crisis and environmental concerns (e.g. global warming) [9] and (2) increasing energy consumption in data centers [18]. We now discuss the related work.

Geographical Load Balancing. The research community has recently identified the potential of reducing the operating

cost in data centers by geographical load balancing based on the spatio-temporal variation in electricity prices. Qureshi et al. [3] studied the problem of reducing the electricity cost in a wholesale market environment. They try to lower the electricity bill by utilizing the varying electricity prices in different locations of distributed data centers. They describe greedy heuristics and evaluate them on historical electricity prices and network traffic data. But they did not consider migration of workload and SLA requirements. In this paper, we utilize SLA information for load balancing and compare our algorithms with their proposed greedy algorithm. Rao et al. [4] consider load-balancing of delay sensitive applications with the objective of minimizing the current energy cost subject to delay constraints in a multi-electricity-market environment. They used sophisticated queuing theory to restrict the average latency rather than utilizing the flexibilities from the SLAs and dynamic migration of workload. Buchbinder et al. [5] presented online algorithms for migrating jobs between data centers, which handle the fundamental tradeoff between energy and bandwidth costs. For constant workload they could give bounded competitive ratio but for varying workload they presented a heuristic algorithm to reduce the computational complexity without making any probabilistic assumption about the future workload and future electricity prices. In contrast, we use the deadline requirements and use probabilistic assumptions to make scheduling decisions. There has also been some work on utilizing renewable energy for energy efficiency in data centers. Liu et al. [9] presented formulation for geographical load balancing without deadline and investigated how renewable energy can be used to lower the electricity price of brown energy. In contrast we consider migration of workload between data centers to utilize energy price variation via dynamic deferral. Stewart et al. [8] try to maximize the use of renewable energy in data centers. However, they assume that data centers have their own energy sources (solar plants, wind mills, etc.). Unlike using renewable energy, we consider a different case where the cloud service providers buy energy from whole sale markets, which is a more common case for many data centers.

Scheduling with deadline. Many applications in real world require delay bound or deadline constraint e.g. see Lee et al. [19]. When combining with energy conservation, deadline is usually a critical adjusting tool between performance loss and energy consumption. Energy efficient deadline scheduling was first studied by Yao et al. [20]. They proposed algorithms, which aim to minimize energy consumption for independent jobs with deadline constraints on a single variable-speed processor. In the context of data center, most work on energy management merely talk about minimizing the average delay but not give any bound on delay until recently. Mukherjee et al. [21] proposed online algorithms considering deadline constraints to minimize the computation, cooling and migration energy for machines. Goiri et al. [6], [22] utilize availability of green energy and deadline information to schedule jobs in a data center. However, these works focus on job assignment inside one data center without electricity price variation.

VI. Conclusion

In this paper we have proposed online algorithms for geographical load balancing in data centers while guaranteeing the deadlines. The algorithms utilize the latency requirements of workloads as well as exploit the electricity price variation for cost savings and guarantee bounded cost and bounded latency under very general settings - arbitrary workload, general deadline and general energy cost models. Further the online algorithms are simple to implement and do not require significant computational overhead. To the best of our knowledge, this is the first formulation for load balancing with deadline utilizing the slackness in the execution of jobs for energy savings.

Our experiments highlight that significant cost and energy savings can be achieved via dynamic deferral of workload. However the performance of the online algorithms depend on the price variation and the quality of prediction. In this paper, we tried to limit our motivation towards the cloud considering data centers as computation units. Other factors such as capacity provisioning, heterogeneity, availability of renewable energy etc. could be taken into account during load balancing decisions. We would like to consider these issues with load balancing in future.

Acknowledgment

This work was sponsored in part by the Multiscale Systems Center (MuSyC) and NSF Variability Expedition.

References

[1] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, *A taxonomy and survey of energy-efficient data centers and cloud computing systems*, In Advances in Computers, Elsevier: Amsterdam, 2011.
[2] Z. Liu, M. Lin, A. Wierman, S. Low, and L. H. Andrew, *Greening Geographical Load Balancing,* In Proc. ACM SIGMETRICS, June 2011.
[3] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, *Cutting the electric bill for internet-scale systems,* In Proc. ACM SIGCOMM, August 2009.
[4] L. Rao, X. Liu, L. Xie, and W. Liu. *Minimizing electricity cost: Optimization of distributed internet data centers in a multi-electricity-market environment*, In Proc. IEEE INFOCOM, 2010.
[5] N. Buchbinder, N. Jain, and I. Menache, *Online job-migration for reducing the electricity bill in the cloud,* In Proc. IFIP Networking, 2011.
[6] I. Goiri et al. *GreenSlot: Scheduling Energy Consumption in Green Datacenters.* In Proc. Supercomputing, November 2011.
[7] S. K. Garg and R. Buyya. *SLA-based Resource Provisioning for Heterogeneous Workloads in a Virtualized Cloud Datacenter*, In Proc. of ICA3PP, October 2011.
[8] C. Stewart and K. Shen, *Some Joules Are More Precious Than Others: Managing Renewable Energy in the Datacenter,* In Proc. Power Aware Comput. and Sys., October 2009.
[9] Z. Liu, M. Lin, A. Wierman, S. Low, and L. H. Andrew, *Geographical load balancing with renewables*, In Proc. GreenMetrics, June 2011.
[10] M. Lin, A. Wierman, L. H. Andrew, and E. Thereska, *Dynamic right-sizing for power-proportional data centers*, In Proc. IEEE INFOCOM, April 2011.
[11] A. H. Mohsenian-Rad and A. Leon-Garcia, *Optimal residential load control with price prediction in real-time electricity pricing environments,* IEEE Trans. Smart Grid, 1(2), pp. 120-133, Sep. 2010.
[12] http://www.iso-ne.com.
[13] http://www.nyiso.com.
[14] http://www.ercot.com.
[15] http://www.electricityinfo.co.nz.
[16] Y. Chen, A. Ganapathi, R.Griffith, and R. Katz, *The Case for Evaluating MapReduce Performance Using Workload Suites,* In Proc. IEEE MASCOTS, 2011.
[17] K. Kc and K. Anyanwu, *Scheduling Hadoop Jobs to Meet Deadlines,* In Proc. IEEE CloudCom, 2010.
[18] *Server and Data Center Energy Efficiency,* Final Report to Congress, U.S. Environmental Protection Agency, 2007.
[19] C. B. Lee, A. Snavely, *Precise and realistic utility functions for user-centric performance analysis of schedulers*, In Proc. HPDC, 2007.
[20] F. Yao, A. Demers, and S. Shenker, *A scheduling model for reduced CPU energy*, In Proc. FOCS, pp. 374-382, 1995.
[21] T. Mukherjee, A. Banerjee, G. Varsamopoulos, and S. K. S. Gupta, *Spatio-Temporal Thermal-Aware Job Scheduling to Minimize Energy Consumption in Virtualized Heterogeneous Data Centers*, Computer Networks, 2009.
[22] I. Goiri, K. Le, T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini, *GreenHadoop: Leveraging Green Energy in Data-Processing Frameworks,* In Proc. EuroSys, April 2012.