Online Job-Migration for Reducing the Electricity Bill in the Cloud

Niv Buchbinder[1], Navendu Jain[2], and Ishai Menache[2]

[1] Open University, Israel
niv.buchbinder@gmail.com
[2] Microsoft Research
{navendu,t-ismena}@microsoft.com

Abstract. Energy costs are becoming the fastest-growing element in datacenter operation costs. One basic approach to reduce these costs is to exploit the spatiotemporal variation in electricity prices by moving computation to datacenters in which energy is available at a cheaper price. However, injudicious job migration between datacenters might increase the overall operation cost due to the bandwidth costs of transferring application state and data over the wide-area network. To address this challenge, we propose novel online algorithms for migrating batch jobs between datacenters, which handle the fundamental tradeoff between *energy* and *bandwidth* costs. A distinctive feature of our algorithms is that they consider not only the current availability and cost of (possibly multiple) energy sources, but also the *future* variability and uncertainty thereof. Using the framework of competitive-analysis, we establish worst-case performance bounds for our basic online algorithm. We then propose a practical, easy-to-implement version of the basic algorithm, and evaluate it through simulations on real electricity pricing and job workload data. The simulation results indicate that our algorithm outperforms plausible greedy algorithms that ignore future outcomes. Notably, the actual performance of our approach is significantly better than the theoretical guarantees, within 6% of the optimal offline solution.

Keywords: cloud computing, energy efficiency, job migration, datacenters.

1 Introduction

Energy costs are becoming a substantial factor in the operation costs of modern datacenters (DCs). It is expected that by 2014, the infrastructure and energy cost (I&E) will become 75% of the total operation cost, while information technology (IT) expenses, i.e., the equipment itself, will induce only 25% of the cost (compared with 20% I&E and 80% IT in the early 90's) [5]. Moreover, the operating cost per server will be double its capital cost (over its amortized lifespan of 3-5 years). According to an EPA report, servers and datacenters consumed 61 billion Kilowatt at a cost of $4.5 billion in 2007, with demand expected to double by 2011 [13]. These costs might further increase due to carbon taxes which are being imposed by several countries to reduce the environmental impact of electricity generation and consumption [11,16].

To reduce the energy costs, research efforts are being made in two main directions: (i) Combine traditional electricity-grid power with *renewable* energy sources such as

J. Domingo-Pascual et al. (Eds.): NETWORKING 2011, Part I, LNCS 6640, pp. 172–185, 2011.

solar, wave, tidal, and wind, motivated by long-term cost reductions and growing unreliability of the electric grid [15,18]; and (ii) Exploit the temporal and geographical variation of electricity prices [19,20]. As shown in Fig 1(a), energy prices can significantly vary over time and location (even within 15 minute windows). Therefore, a natural approach is to shift the computational load to datacenters where the current energy prices are the cheapest. In this paper, we combine these two directions, by proposing *efficient algorithms for dynamic placement (migration) of batch applications (or jobs) in datacenters, based on their energy cost and availability.* Specifically, we focus on executing batch applications such as MapReduce programs, web crawling and index generation for web search, and data analytics, which are typically delay tolerant and can handle the interruption during job migration between datacenters[1].

A key challenge for dynamic job placement is to account for the bandwidth cost of moving the application state and data between source and destination datacenters. The application data and memory footprint depends on the underlying job, and can range from a few KBs to hundreds of MBs [12] [2]. Bandwidth fees are typically charged proportionally to the number of bytes sent and received during the migration process [23]; consequently, bandwidth costs increase linearly with the amount of migrated data.

This paper proposes online algorithms for migrating batch applications within a cloud of multiple datacenters. The input for these algorithms includes the current energy prices and power availability at different datacenters. The algorithms accordingly determine which jobs should be migrated and to which hosting location, while taking into account the bandwidth costs and the current job allocation. Our model accommodates datacenters powered by multiple energy suppliers such as the grid and renewable energy sources. More generally, we assume that the total energy cost in each datacenter is a convex increasing function of utilization. Energy-cost reduction would thus depend on the ability to exploit the instantaneously cheap energy sources to their full availability. A distinctive feature of our approach is that we do *not* settle for an intuitive greedy approach to optimize the overall energy and bandwidth costs, yet take into account future deviations in energy price and power availability. We emphasize that our electricity-pricing model is distribution-free (i.e., we do not make probabilistic assumptions on future electricity prices), as prices can often vary unexpectedly [19].

Related work. The general framework of energy-efficient resource management in cloud datacenters has been an active research area over the last few years. Numerous papers consider the *intra* datacenter optimization, which includes techniques such as switching off idle servers [14] and VM migration and consolidation for load balancing and power management [6]. Motivated by the notable spatiotemporal variation in electricity prices, the networking research community has recently started focusing on *inter* datacenter optimization, which is also the context of our work. Qureshi et al. [19] propose greedy heuristics for redirecting application requests to different datacenters, and evaluate them using simulations on historical electricity prices and Akamai's traffic

[1] We note that there are known job migration techniques (e.g., pipelining and incremental snapshots [12]) to minimize the interruption delay.

[2] Note that persistent data such as crawled web pages and click-stream logs, are typically replicated at multiple datacenters and thus do not need to be migrated for individual jobs.

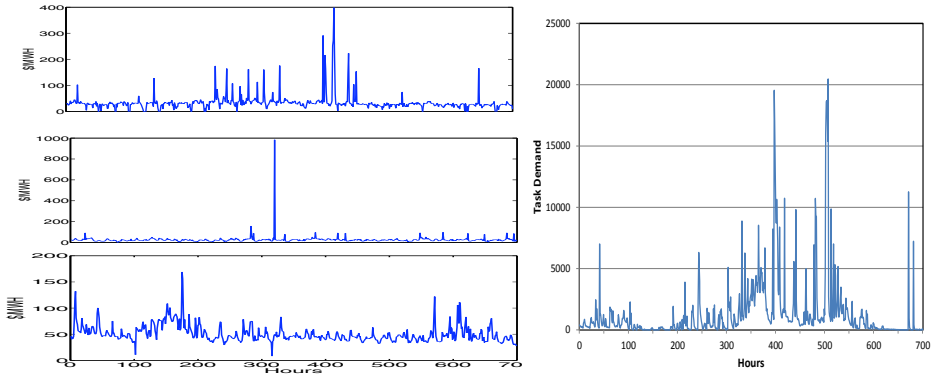

Fig. 1. (a, left) Electricity pricing trends across three US locations (CA (top), TX (middle), and MA (bottom)) over a month in 2009. The graphs indicate significant spatiotemporal variation in electricity prices. (b, right) Hourly task demand data from a cluster running MapReduce jobs. The demand data shows significant variation over time.

data. The authors show that judicious placement can save millions of dollars in energy costs. Rao et al. [20] consider load-balancing of delay sensitive applications with the goal of minimizing the energy cost subject to delay constraints. They use queuing theory and linear programming techniques and report cost savings through simulations with electricity pricing data. Finally, Le et al. [17] propose an optimization-based framework to distribute user requests across datacenters. Their goal is to process requests within a specified cap on energy from non-renewable sources, while meeting the application service-level agreements and minimizing energy costs.

Our paper significantly differs from prior work in three ways. First, we consider the migration of *batch* applications, rather than the redirection of *interactive* applications' requests. Second, our algorithms take into account not only the immediate savings of job migration, but also the future consequences thereof, thereby accounting for future uncertainty in the cost parameters. Third, we incorporate the bandwidth cost into the optimization problem, as it can be a significant cost factor for job migration.

Our solution method lies within the framework of online computation [7], which has been applied for solving problems with uncertainty in networks, finance, mechanism design and other fields. Specifically, we employ online primal-dual techniques [10], including recent ideas from [2,3]. We build upon two well-studied online problems, weighted paging/caching [1,21] and Metrical Task System (MTS) [4,8] (see our technical report [9] for details), and extend them to address two new challenges specific to our problem: (1) optimizing energy costs of executing batch jobs under time-varying electricity prices and job demand, and (2) balancing energy savings with bandwidth costs to minimize the total operation cost. Since our problem is a generalization of the MTS problem [8], we get a lower bound of $\Omega(\log n)$ on the competitiveness of any online algorithm where n is the number of datacenters. A tighter lower bound is subject for future examination.

Contribution and Paper Organization. In this paper, we show, both analytically and empirically, that existing cloud computing infrastructure can perform dynamic application placement based on energy cost and availability and bandwidth costs, for significant economic gains. Specifically,

• We design a basic online algorithm (Section 3) which is $O(\log H_0)$-competitive (compared to the optimal offline), where H_0 is the total number of servers in the cloud.
• To reduce the extensive computation required by the basic algorithm in each iteration, we construct a computationally efficient version of the basic algorithm, which can be easily implemented in practice (Section 4).
• We evaluate our algorithms (Section 5) using real electricity pricing data obtained from 30 US locations over three years, and job-demand data from a large cluster in a datacenter, and show significant improvements over greedy-optimization algorithms. Overall, the cost savings of our online algorithm are typically within 6% of the optimal solution.

The technical proofs of this work are omitted due to lack of space and can be found in an accompanying technical report [9].

2 The Model

In this section we formally define the job migration problem. In Section 2.1 we describe the elements of the problem. In Section 2.2 we formalize the online optimization problem, the solution of which is the main objective of this paper.

2.1 Preliminaries

We consider a large computing facility (henceforth referred to as a "cloud", for simplicity), consisting of n datacenters (DCs). Each DC $i \in \{1,\ldots,n\}$ contains a set $\mathcal{H}_i = \{1,\ldots,H_i\}$ of H_i servers which can be activated simultaneously. We denote by $H_0 = \sum_{i=1}^{n} H_i$ the total number of servers in the cloud, and often refer to H_0 as the *capacity* of the cloud. All servers in the cloud are assumed to have equal resources (in terms of CPU, memory, etc.).

A basic assumption of our model is that *energy (or electricity) costs* may vary in time, yet remain fixed within time intervals of a fixed length of τ (say 15 minutes, or one hour). Accordingly, we shall consider a discrete time system, where each time-step (or *time*) $t \in \{1,\ldots,T\}$ represents a different time interval of length τ. We denote by B_t the total job-load at time t measured in server units, namely the total number of servers which need to be active at time t. For simplicity, we assume that each job requires a single server, thus B_t is also the total number of jobs at time t. We note, however, that the latter assumption can be relaxed by considering a single job per processor or per VM, similar to the virtualization setup. In Section 3 we consider a simplified scenario where $B_t = B$, namely the job-load in the cloud is fixed. We will address the case where B_t changes in time, which corresponds to job arrivals and departures to/from the cloud in Section 4. At each time t, the control decision is to assign jobs to DCs, given the current electricity and bandwidth costs, which are described below.

Energy costs. We assume that the energy cost at every DC is a function of the number of servers that are being utilized. Let $y_{i,t}$ be the number of servers that are utilized

in DC i at time t. The (total) energy cost is given by $\tilde{C}_{i_t}(y_{i,t})$ (measured in dollars per hour). Let $C_{i_t} : \mathbb{R}_+ \to \mathbb{R}_+$ be the interpolation of $\tilde{C}_{i_t}(\cdot)$; for convenience, we shall consider this function in the sequel. We assume throughout that $C_{i_t}(y_{i,t})$ is a *non-negative, (weakly) convex increasing function*. Note that the function itself can change with t, which allows for time variation in energy prices. We emphasize that we do not make any probabilistic assumptions on $C_{i_t}(\cdot)$, i.e., this function can change *arbitrarily* over time. The simplest example that complies with the convexity assumption is the linear cost model, in which the total energy cost is linearly proportional to the number of servers utilized. Assuming that servers not utilized are switched off and consume zero power, we have $C_{i_t}(y_{i,t}) = c_{i,t}y_{i,t}$, where $c_{i,t}$ is the cost of using a single server at time t. Further examples can be found in [9].

Bandwidth Costs. The migration of jobs between DCs induces bandwidth costs. In practice, bandwidth fees are paid both for outgoing traffic and incoming traffic. Accordingly, we assume the following cost model. For every DC i, we denote by $d_{i,out}$ the bandwidth cost of transferring a job *out* of DC i, and by $d_{i,in}$ the bandwidth cost of migrating a job into this DC[3]. Thus, the overall cost of migrating a job from DC i to DC j is given by $d_{i,out} + d_{j,in}$. We note that there are also bandwidth costs associated with the arrival of jobs into the cloud (e.g., from a central dispatcher) and leaving the cloud (in case of a job departure). However, these costs are constant and do not depend on the migration control, and are thus ignored in our formulation.

Our goal is to minimize the total operation cost in the cloud, which is the sum of the energy costs at the DC and the bandwidth costs of migrating jobs between DCs. It is assumed that the job migration time is negligible with regard to the time interval τ in which the energy costs are fixed. This means that migrated jobs at time t incur the energy prices at their new DC, starting from time t. We refer to the above minimization problem as the *migration problem*. If all future energy costs were known in advance, then the problem could have optimally be solved as a standard convex program. However, our basic assumption is that electricity prices change in an unpredicted manner. We therefore tackle the migration problem as an *online* optimization problem.

2.2 Problem Formulation

In this subsection we formalize the migration problem. We shall consider a slightly different version of the original problem, which is algorithmically easier to handle. Nonetheless, the solution of the modified version leads to provable performance guarantees for the original problem. Recall that we consider the case of fixed job-load, namely $B_t = B$ for every t. We argue that it is possible to define a modified model in which we charge a migration cost $d_i \triangleq d_{i,in} + d_{i,out}$ whenever the algorithm migrates a job out of DC i, and do not charge for migrating jobs into the DC. The following lemma states that this bandwidth-cost model is equivalent to the original bandwidth-cost model up to an additive constant.

Lemma 1. *The original and modified bandwidth-cost model, in which we charge d_i for migrating jobs out of DC i, are equivalent up to additive constant factors. In particular,*

[3] ISPs charge bandwidth to DC operators based on the daily 95^{th} percentile of incoming/outgoing bandwidth over 5 min periods.

a c-competitive algorithm for the former results in a c-competitive algorithm for the latter.

We note that the same result holds for the model which charges d_i for migrating jobs *into* the DC, and nothing for migrating jobs out of the DC.

Using Lemma 1, we first formulate a convex program whose solution provides a lower bound on the optimal solution for the (original) migration problem. Let $z_{i,t}$ be the number of jobs that are migrated from DC i at time t. This variable is multiplied by d_i for the respective bandwidth cost. The optimization problem is defined as

$$\min \quad \sum_{i=1}^{n}\sum_{t=1}^{T} d_i z_{i,j} + \sum_{i=1}^{n}\sum_{t=1}^{T} C_{i_t}(y_{i,t}),$$

subject to the following constraints: $\sum_{i=1}^{n} y_{i,t} \geq B$ for every t (i.e., meet the job demand); $z_{i,t} \geq y_{i,t-1} - y_{i,t}$ for every i,t (i.e., $z_{i,t}$ reflects the number of jobs leaving DC i at time t); $y_{i,t} \leq H_i$ for every i,t (i.e., meet the capacity constraint); $z_{i,t}, y_{i,t} \in \{0,1,\ldots,H_i\}$ for every i,t. We shall consider a relaxation of the above optimization problem, where the last constraint is replaced with $z_{i,t}, y_{i,t} \geq 0$ for every i,t. A solution to the relaxed problem is by definition a lower bound on the value of the original optimization problem.

Instead of considering the above described convex program (with the relaxation), we construct below an equivalent LP, which will be easier to work with. For every time t, the cost of using the j-th server at DC i is given by $c_{i,j,t}$, where $c_{i,j,t} \in [0,\infty]$ is increasing in j ($1 \leq j \leq H_i$). We assume that $c_{i,j,t}$ may change arbitrarily over time (while keeping monotonicity in j as described above). Note that we allow $c_{i,j,t}$ to be unbounded, to account for the case where the total energy availability in the DC is not sufficient to run all of its servers. Let $0 \leq y_{i,j,t} \leq 1$ be the utilization level of the j-th server of DC i (we allow for "partial" utilization, yet note that at most a single server will be partially utilized in our solution). Accordingly, the energy cost of using this server at time t is given by $c_{i,j,t}y_{i,j,t}$. Define $z_{i,j,t}$ to be the workload that is migrated from the jth server of DC i to a different location. This variable is multiplied by d_i for the respective bandwidth cost. The optimization problem is defined as follows.

$$(P): \min \quad \sum_{i=1}^{n}\sum_{j=1}^{H_i}\sum_{t=1}^{T} d_i z_{i,j,t} + \sum_{i=1}^{n}\sum_{j=1}^{H_i}\sum_{t=1}^{T} c_{i,j,t}\cdot y_{i,j,t} \qquad (1)$$

$$\text{subject to} \quad \sum_{i=1}^{n}\sum_{j=1}^{H_i} y_{i,j,t} \geq B \quad \text{for every } t,$$

$$z_{i,j,t} \geq (y_{i,j,t-1} - y_{i,j,t}), \quad 0 \leq y_{i,j,t} \leq 1, \quad z_{i,j,t} \geq 0, \quad \text{for every } i,j,t.$$

Observe that the first term in the objective function corresponds to the total bandwidth cost, whereas the second term corresponds to the total energy cost.

One may notice that the above problem formulation might pay the bandwidth cost d_i for migrating data *within* each DC i, which is clearly not consistent with our model assumptions. Nevertheless, since the energy costs $c_{i,j,t}$ in each DC i are non-decreasing in j for every t, an optimal solution will never prefer servers with higher indexes over lower-index servers. Consequently, jobs will not be migrated within a DC. Thus, solving (P) is essentially equivalent to solving the original convex problem. This observation, together with Lemma 1, allows us to consider the optimization problem (P). More formally,

Lemma 2. *The value of (P) is a lower bound on the value of the optimal solution to the migration problem.*

We refer to the optimization problem (1) as our *primal problem* (or just(P)). The dual of (P) is given by

$$\begin{aligned}(D): \max \quad & \textstyle\sum_{t=1}^{T} Ba_t - \sum_{i=1}^{n}\sum_{j=1}^{H_i}\sum_{t=1}^{T} s_{i,j,t}, \\ \text{subject to} \quad & -c_{i,j,t} + a_t + b_{i,j,t} - b_{i,j,t+1} - s_{i,j,t} \leq 0, \\ & 0 \leq b_{i,j,t} \leq d_i, \quad \text{and} \quad s_{i,j,t} \geq 0 \quad \text{for every } i,j,t.\end{aligned}$$

By Lemma 2 and the weak duality theorem (see, e.g., [22]), an algorithm that is c-competitive with respect to a *feasible* solution of (D), would be c-competitive with respect to the offline optimal solution.

3 An Online Job-Migration Algorithm

In this section we design and analyze an online algorithm for the migration problem (P). The algorithm we design is based on a *primal-dual* approach, meaning that the new primal and dual variables are *simultaneously* updated at every time t. The general idea behind the algorithm is to maintain a feasible dual solution to (D), and to upper-bound the operation cost at time t (consisting of bandwidth cost and the energy cost) as a function of the value of (D) at time t. Since a feasible solution to (D) is a lower bound on the optimal solution, this procedure would immediately lead to a bound on the competitive ratio of the online algorithm.

The online algorithm outputs a *fractional* solution to the variables $y_{i,j,t}$. To obtain the total number of servers that should be activated in DC i, we simply calculate the sum $\sum_{j=1}^{H_i} y_{i,j,t}$ at every time t. Since this sum is generally a fractional number, one can round it to the nearest integer. Because the number of servers in each DC (H_i) is fairly large, the effect of the rounding is negligible and thus ignored in our analysis.

The online algorithm receives as input the energy cost vector $\{c_{i,j,t}\}_{i,j}$ at every time t defining the energy cost at each server (i,j) at time interval t. As a first step, we use a well known reduction that allows us to consider only *elementary cost vectors* [7, Sec. 9.3.1]. Elementary cost vectors are vectors of the form $(0,\ldots,0,c_{i_t,j_t},0,\ldots,0)$, namely vectors with only a single non-zero entry. This reduction allows us to split any general cost vector into a finite number of elementary cost vectors without changing the value of the optimal solution. Furthermore, we may translate any online migration decisions done for these elementary task vectors to online decisions on the original cost vectors without increasing our cost. Thus, we may consider only elementary cost vectors from now on. Abusing our notations, we use the original time index t to describe these elementary cost vectors. We use c_t instead of c_{i_t,j_t} to describe the (single) non-zero cost at server (i_t,j_t) at time t. Thus, the input for our algorithm consists of i_t, j_t, c_t.

The algorithm we present updates at every time t the (new) dual variables a_t, s_t and $b_{i,j,t+1}$. The values of these variables are determined incrementally, via a continuous update rule (closed-form one-shot updates seems hard to obtain). We summarize in Table 1 the procedure for the update of the dual variables at any time t. We refer to the procedure at time t as the t-th *iteration*. We note that the continuous update rule is

Table 1. The procedure for updating the dual variables of the online problem

Input: Datacenter i_t, server j_t, cost c_t (at time t).
Initialization: Set $a_t = 0$. For all i, j: set $b_{i,j,t+1} = b_{i,j,t}$ and $s_{i,j,t} = 0$.
Loop:

- Keep increasing a_t, while **termination condition** is not satisfied.
- Update the other dual variables as follows:
  - For every $i, j \neq i_t, j_t$ such that $y_{i,j,t} < 1$, increase $b_{i,j,t+1}$ with rate $\frac{db_{i,j,t+1}}{da_t} = 1$.
  - For every $i, j \neq i_t, j_t$ such that $y_{i,j,t} = 1$, increase $s_{i,j,t}$ with rate $\frac{ds_{i,j,t}}{da_t} = 1$.
  - For $i, j = i_t, j_t$, decrease $b_{i_t,j_t,t+1}$ so that $\sum_{i,j \neq i_t,j_t} \frac{dy_{i,j,t}}{da_t} = -\frac{dy_{i_t,j_t,t}}{da_t}$.

Termination condition: Exit when either $y_{i_t,j_t,t} = 0$, or $-c_t + a_t + b_{i_t,j_t,t} - b_{i_t,j_t,t+1} = 0$.

mathematically more convenient to handle. Nonetheless, it is possible to implement the online algorithm through discrete-time iterations, by searching for the scalar value a_t up to any required precision level.

Each primal variable $y_{i,j,t}$ is continuously updated as well, alongside with the continuous update of the respective dual variable $b_{i_t,j_t,t+1}$. The following relation between $b_{i_t,j_t,t+1}$ and $y_{i,j,t}$ is preserved throughout the iteration:

$$y_{i,j,t} := \frac{1}{H_0}\left(\exp\left(\ln(1+H_0)\frac{b_{i,j,t+1}}{d_i}\right) - 1\right). \tag{2}$$

The variable $y_{i_t,j_t,t}$ is updated so that the amount of work that is migrated to other servers is equal to the amount of work leaving j_t (i.e, total work conservation).

The next theorem presents a performance bound for the above described algorithm.

Theorem 1. *The online algorithm is $O(\log(H_0))$-competitive.*

The proof proceeds in a number of steps. We first show that our algorithm preserves a feasible primal solution and a feasible dual solution. We then relate the change in the dual variable $b_{i,j,t+1}$ to the change in the primal variable $y_{i,j,t}$, and the change of the latter to the change in the value of the feasible dual solution. This allows us to upper bound the bandwidth cost at every time t as a function of the change in the value of the dual. Finally, we find a precise relation between the bandwidth cost and the energy cost, which allows us to also bound the latter as a function of the change in the dual value. A detailed proof can be found in [9].

4 An Efficient Online Algorithm

Theorem 1 indicates that the online algorithm is expected to be robust against *any* possible deviation in energy prices. However, its complexity might be too high for an efficient implementation with an adequate level of precision. First, the reduction of [7, Sec. 9.3.1] generally splits each cost vector at any given time t into $O(H_0^2)$ vectors, thereby requiring a large number of iterations per each actual time step t. Second, the

Table 2. The iteration of the EOA. Time indexes are omitted.

Input: (i) Cost vector $\mathbf{c} = (c_1, c_2, \ldots, c_n)$; (ii) $y = (y_1, y_2, \ldots, y_n)$ (the current load vector on the datacenters).
Job Migration Loop: Outer loop: $k = 1$ to n, Inner loop: $i = k + 1$ to n
– Move load from DC i to DC k according to the following migration rule.

$$\min \left\{ y_i, H_k - y_k, s_1 \cdot \frac{c_{i,k} y_i}{d_{i,k}} \cdot (y_k + s_2) \right\}, \tag{3}$$

where $s_1, s_2 > 0$.

need for proper discretization of the continuous update rule of the dual variables (see Table 1) might make each iteration computationally expensive. Therefore we design in this section an "easy-to-implement" online algorithm, which inherits the main ideas of our original algorithm, yet decreases significantly the running complexity per time step.

We employ at the heart of the algorithm several ideas and mathematical relations from the original online algorithm. The algorithm can accordingly be viewed as a computationally-efficient variant of the original online algorithm, thus would be henceforth referred to as the *Efficient Online Algorithm* (EOA). While we do not provide theoretical guarantees for the performance of the EOA, we shall demonstrate through real-data simulations its superiority over plausible greedy heuristics (see Section 5). The EOA provides a complete algorithmic solution which also includes the allocation of newly arrived jobs; the way job arrivals are handled is described towards the end of this section.

For simplicity, we describe the EOA for *linear* electricity prices (i.e., the electricity price for every DC i and time t is $c_{i,t}$, regardless of the number of servers that are utilized). In [9], we describe the adjustments needed to support non-linear prices. The input for the algorithm at every time t is thus an n dimensional vector of prices, $\mathbf{c}_t = (c_{1,t}, c_{2,t}, \ldots, c_{n,t})$. For ease of illustration, we reorder the DCs in each iteration according to the present electricity prices, so that $c_1 \leq c_2 \leq \cdots \leq c_n$. For exposition purposes we omit the time index t from all variables (e.g., we write $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ instead of $\mathbf{c}_t = (c_{1,t}, c_{2,t}, \ldots, c_{n,t})$). We denote by $d_{i,k} = d_{i,out} + d_{k,in}$ the bandwidth cost per unit for transferring load from DC i to DC k. We further denote by $c_{i,k}$ the difference in electricity prices between the two DCs, namely $c_{i,k} = c_i - c_k$, and refer to $c_{i,k}$ as the *differential energy cost* between i and k. The iteration of the EOA is described in Table 2.

Before elaborating on the logic behind the EOA, we note that the running complexity of each iteration is $O(n^2)$. Observe that the inner loop makes sure that data is migrated from the currently expensive DCs to cheaper ones (in terms of the corresponding electricity prices). We comment that the fairly simple migration rule (3) could be implemented in a distributed manner, where DCs exchange information on their current load and electricity price; we however do not focus on a distributed implementation in the current paper.

We now proceed to discuss the rational behind the EOA, and in particular the relation between (3) (which will be henceforth referred to as the *migration rule*) and the

original online algorithm described earlier. We first motivate the use of the term $c_{i,k}y_i$ in the migration rule. Note first that this term corresponds to the energy cost that could potentially be saved by migrating all jobs in DC i to DC k. The reason that we use differential energy costs $c_{i,k} = c_i - c_k$ is *directly* related to the reduction from general cost vectors to elementary cost vectors (see Section 3). The reduction creates elementary cost vectors by iterating over the differential energy costs $c_i - c_{i-1}$, $i \in [2, n]$.

Further examination of the migration rule (3) reveals that the amount of work that is migrated from DC i to DC k is proportional to y_k (the load at the target DC). The motivation for incorporating y_k in the migration rule follows by differentiating the primal-dual relation (2) with respect to $b_{i,j,t+1}$, which leads to the following relation

$$\frac{dy_{i,j,t}}{db_{i,j,t+1}} = \frac{\ln(1+H_0)}{d_i} \cdot \left(y_{i,j,t} + \frac{1}{H_0} \right) \tag{4}$$

The above equation implies that the change in the load in DC k should be proportional to the current load. As discussed earlier, this feature essentially encourages the migration of jobs into DCs that were consistently "cheap" in the past, and consequently loaded in the present. Another idea that follows from (4) and is incorporated in the migration rule is that the migrated workload is *inversely* proportional to the bandwidth cost. The last idea which is borrowed from (4) is to include an additive term s_2, which reduces the effect of y_k. This additive term enables the migration of jobs to DCs, even if they are currently empty. Intuitively, the value of s_2 manifests the natural tradeoff between making decisions according to current energy prices (high s_2) or relaying more on usage history (low s_2). The other parameter in the migration rule, s_1, sets the "aggressiveness" of the algorithm. Increasing s_1 makes the algorithm greedier in exploiting the currently cheaper electricity prices (even at the expense of high bandwidth costs). Both s_1 and s_2 can be tuned throughout the execution of the algorithm.

To complete the description of the EOA, we next specify how newly arrived traffic is distributed. We make here the simplifying assumption that jobs arrive to a single global dispatcher, and then allocated to the different DCs. The case of multiple dispatchers is outside the scope of this paper, nonetheless algorithms for this case can be deduced from the rule we describe below. A newly arrived job is assigned through the following probabilistic rule: *At every time t, assign the job to DC i with probability proportional to:* $\frac{1}{d_i}(y_i + s_2)$, *where* $d_i = d_{i,in} + d_{i,out}$. The reasoning behind this rule is the same as elaborated above for the migration rule (3).

5 Simulations

In this section we evaluate the performance of the EOA on real datasets, while comparing it to two plausible greedy algorithms.

Alternative Job-Migration Algorithms. As a first simple benchmark for our algorithm, we use a greedy algorithm referred to as *Move to Cheap* (MTC) that always keeps the load on DCs having the lowest electricity prices (without exceeding their capacity). While its energy costs are the lowest possible by definition, the bandwidth costs turn out to be significant and its overall performance is eventually poor. The second greedy heuristic that we consider is *MimicOPT*. This heuristic tries to emulate the

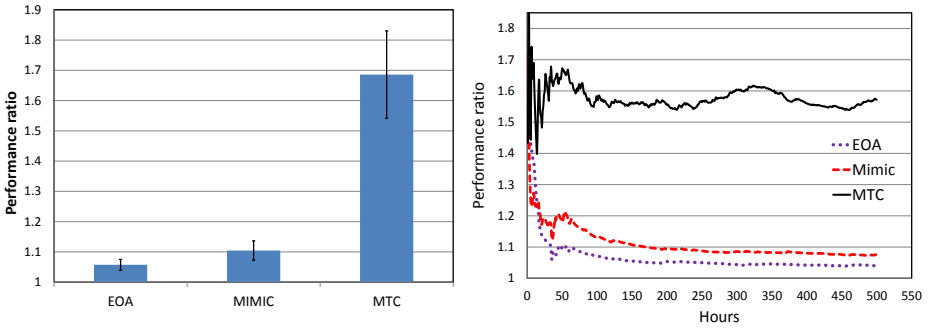

Fig. 2. Performance comparison of the MTC, MimicOPT, and EOA algorithms with respect to OPT: (a, left) Average performance ratios along with standard-deviations, and (b, right) The evolution of the competitive ratio of the algorithms with respect to OPT for a sample run

behavior of the optimal solution i.e., at any given time, MimicOPT solves an LP with the energy costs until the current time t, and migrates jobs so as to mimic the job allocation of the current optimal solution up to time t. As MimicOPT does not have the data on future power costs and does not consider future outcomes, it is obviously not optimal. However, this heuristic is reasonable in non-adversarial scenarios and, indeed, performs fairly well. To make the running time of MimicOPT feasible, we restrict the LP to use a window of the last 80 hours (a further increase in the window size does not seem to significantly change the results). In general, we expect MimicOPT to perform well on inputs in which electricity prices are highly correlated in time. We compare below the performance of the MTC, MimicOPT and EOA algorithms. Our reference is naturally the *offline* solution to the optimization problem, denoted OPT. Obviously, OPT obtains the lowest possible cost.

Electricity Prices and Job-Demand Data. We use in our experiments the historical hourly electricity price data ($/MWh) from publicly available government agencies for 30 US locations, covering January 2006 through March 2009. Due to space constraints, we omit the organization details of the electricity system in the US, see, e.g., [19] for details. In addition to the hourly electricity market, there are also 15-minute real-time markets which exhibit relatively higher volatility with high-frequency variation. Since these markets account for only a small fraction of the total energy transactions (below 10%), we do not consider them in our simulations.

A snapshot of the dynamics of electricity prices in different locations (Figure 1(a)) demonstrates that (i) prices are relatively stable in the long-term, but exhibit high day-to-day volatility, (ii) there are short-term spikes, and (iii) hourly prices are not correlated across different locations.

For our simulations, we use task-demand data obtained from a 10K node cluster running MapReduce jobs for a large online services provider. Fig. 1(b) shows the hourly task demand data exhibiting a significant variation in the number of tasks corresponding to job submissions and different MapReduce phases across running jobs.

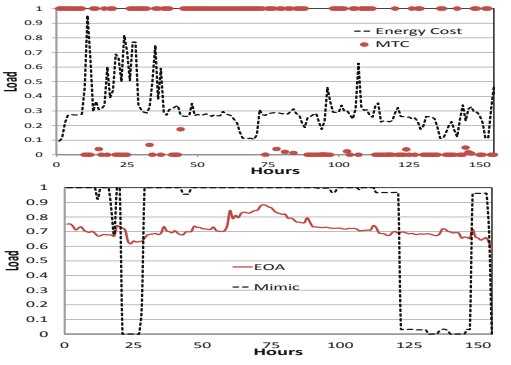

Fig. 3. Job load and normalized electricity price over time at Boston, MA. The top graph shows the electricity price and the load that MTC puts on this DC. The bottom graph shows the load for EOA and MimicOPT.

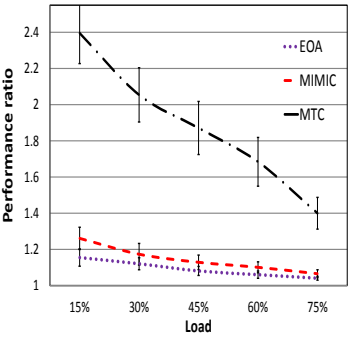

Fig. 4. The relative performance of EOA, MimicOPT, and MTC with respect to OPT as a function of the DC utilization

Performance Evaluation. We first compare the performance ratio of the total electricity and bandwidth costs incurred by different algorithms over pricing data from 10 locations. We assume that each location has 25K servers, each server consumes 200 Watts, bandwidth price is $0.15 per GB [23], and the task memory footprint is 100 MB (corresponding to intermediate results in MapReduce phases). Based on our discussion with datacenter operators, we compute the total job demand by fixing background demand of 60% (15K servers) per location and a dynamic component from MapReduce traces which runs on the remaining 40% capacity.

(i) Evaluating Total Cost. Fig. 2 compares the total cost (energy + bandwidth) for EOA, MimicOPT and MTC algorithms. Fig. 2(a) shows a ten-run average of the competitive ratio of each algorithm in 20 different executions of 500 hours each. We observe that the EOA performs better than MimicOPT and MTC, with a relatively small standard deviation. EOA performs within 5.7% of the optimum while MimicOPT and MTC perform within 10.4% and 68.5% of the optimum, respectively. Fig. 2(b) shows the competitive ratio of the three algorithms as a function of the number of hours in a 500 hour run. We observe that all algorithms exhibit high variation initially due to a high migration cost, whereas OPT migrates much less and stabilizes quickly, because it knows all future prices in advance.

Fig. 3 shows the workload evolution for the (Boston, MA) location. The top graph shows the normalized electricity costs of the location (exhibiting up to 10x variation), along with the normalized load that MTC puts on the Boston location. Note that MTC always uses the cheapest DCs. Thus, a load of one indicates that Boston is one of the 60% cheaper locations, while a load of zero indicates that it is among the 40% more expensive locations. The bottom figure shows the fraction of servers which are kept active in this location by MimicOPT and EOA. As expected, we observe that the number of active servers for MimicOPT varies significantly with electricity pricing (the DC is either full or empty), indicating that the optimal job assignment varies with time across locations. The EOA algorithm exhibits relatively stable job assignment, as it takes into

account the relative pricing of this location, and carefully balances the energy costs with the bandwidth costs of migrating jobs between locations.

(ii) Evaluating Performance as a function of Utilization. Fig. 4 shows the performance ratio as we vary the background utilization of the DCs from 15% to 75%. Each point is the average competitive ratio of ten runs of 500 hours each. We observe that the performance ratio improves (decreases) as the load increases due to the reduced flexibility in the job assignment and EOA performs closest to OPT compared to other algorithms.

Overall, we observe that the job assignment changes significantly with the variation in electricity prices, as all plausible algorithms increase the number of servers at locations with lower prices. Overall, the EOA algorithm provides the best performance ratio and a relatively stable job assignment. While MimicOPT performs close to EOA, the performance gap between the two algorithms becomes larger when subtracting the minimal energy cost that has to be paid by any algorithm. Specifically, the performance-ratio becomes around 1.55 and 1.8 for EOA and MimicOPT, respectively, indicating that the actual cost savings by our algorithm are significant.

6 Concluding Remarks

This paper introduces novel online algorithms for migrating batch applications, with the objective of reducing the total operation cost in a cloud of multiple datacenters. We provide a competitive-ratio bound for the performance of a basic online algorithm, indicating its robustness against any *future* deviation in energy prices. An easy-to-implement version of the basic algorithm is designed, which can be seamlessly integrated as part of existing control mechanisms for datacenter operation. Our simulations on real electricity pricing data and job demand data demonstrate that the actual performance of our algorithm could be very close to the minimal operation cost (obtained through offline optimization). Importantly, our online algorithm outperforms several reasonable greedy heuristics, thereby emphasizing the need for a rigorous online optimization approach.

The algorithms we suggest solve the fundamental energy-cost vs. bandwidth-cost tradeoff associated with migrating batch jobs. Yet, we have certainly not covered herein all possible models and aspects that may need to be considered in the design of future job-migration algorithms. For example, one simplifying assumption we make is that every job can be placed in every DC. In some cases, however, some practical constraints might prevent certain jobs from being run at certain locations. Such constraints could be due to government regulations (e.g., EU data can only be hosted on servers in Europe), delay restrictions (e.g., if a particular DC is too far from the location of the source), security considerations, and other factors. On the other hand, some constraints might require certain jobs to be placed adjacent to others (e.g., to accommodate low inter-component communication delay for interactive applications). Further restrictions might be imposed on the migration paths, e.g., due to the delay overhead of the migration process. Overall, the general framework of online job-migration opens up new algorithmic challenges, as proper consideration of the dynamically evolving electricity prices can contribute to significant cost reductions in cloud environments.

References

1. Bansal, N., Buchbinder, N., Naor, J.: A primal-dual randomized algorithm for weighted paging. In: FOCS, pp. 507–517 (2007)
2. Bansal, N., Buchbinder, N., Naor, J.: Metrical task systems and the k-server problem on HSTs. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6198, pp. 287–298. Springer, Heidelberg (2010)
3. Bansal, N., Buchbinder, N., Naor, J.: Towards the randomized k-server conjecture: a primal-dual approach. In: SODA (2010)
4. Bartal, Y., Blum, A., Burch, C., Tomkins, A.: A polylog(n)-competitive algorithm for metrical task systems. In: STOC, pp. 711–719 (1997)
5. Belady, C.: In the data center, power and cooling costs more than it equipment it supports. Electronics Cooling Magazine (February 2007)
6. Beloglazov, A., Buyya, R.: Energy efficient resource management in virtualized cloud data centers. In: 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pp. 826–831 (2010)
7. Borodin, A., El-Yaniv, R.: Online computation and competitive analysis. Cambridge University Press, Cambridge (1998)
8. Borodin, A., Linial, N., Saks, M.E.: An optimal on-line algorithm for metrical task system. Journal of the ACM 39(4), 745–763 (1992)
9. Buchbinder, N., Jain, N., Menache, I.: Online job-migration for reducing the electricity bill in the cloud. Tech. Rep. MSR-TR-2011-20, Microsoft Research (February 2011)
10. Buchbinder, N., Naor, J.: The design of competitive online algorithms via a primal-dual approach. Foundations and Trends in Theoretical Computer Science 3(2-3), 93–263 (2009)
11. Butler, D.: (2009), http://www.nature.com/news/2009/090911/full/news.2009.905.html
12. Clark, C., Fraser, K., Steven, H., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A.: Live migration of virtual machines. In: NSDI (2005)
13. Department of Energy: Carbon dioxide emissions from the generation of electric power in the united states (2000), http://www.eia.doe.gov/cneaf/electricity/page/co2_report/co2emiss.pdf
14. Gandhi, A., Gupta, V., Harchol-Balter, M., Kozuch, M.: Optimality analysis of energy-performance trade-off for server farm management. In: Performance (2010)
15. Kahn, C.: As power demand soars, grid holds up... so far (2010), http://www.news9.com/global/story.asp?s=12762338
16. Komanoff, C.: Carbon tax center (2010), http://www.carbontax.org
17. Le, K., Bilgir, O., Bianchini, R., Martonosi, M., Nguyen, T.D.: Managing the cost, energy consumption, and carbon footprint of internet services. In: Sigmetrics, pp. 357–358 (2010)
18. McGeehan, P.: Heat wave report: 102 degrees in central park (2010), http://cityroom.blogs.nytimes.com/2010/07/06/heatwave-report-the-days-first-power-loss/
19. Qureshi, A., Weber, R., Balakrishnan, H., Guttag, J.V., Maggs, B.V.: Cutting the electric bill for internet-scale systems. In: SIGCOMM, pp. 123–134 (2009)
20. Rao, L., Liu, X., Xie, L., Liu, W.: Minimizing Electricity Cost: Optimization of Distributed Internet Data Centers in a Multi-Electricity-Market Environment. In: INFOCOM (2010)
21. Sleator, D.D., Tarjan, R.E.: Amortized efficiency of list update and paging rules. Communications of the ACM 28(2), 202–208 (1985)
22. Vazirani, V.V.: Approximation Algorithms. Springer, Heidelberg (2001)
23. Windows Azure Platform, http://www.azure.com