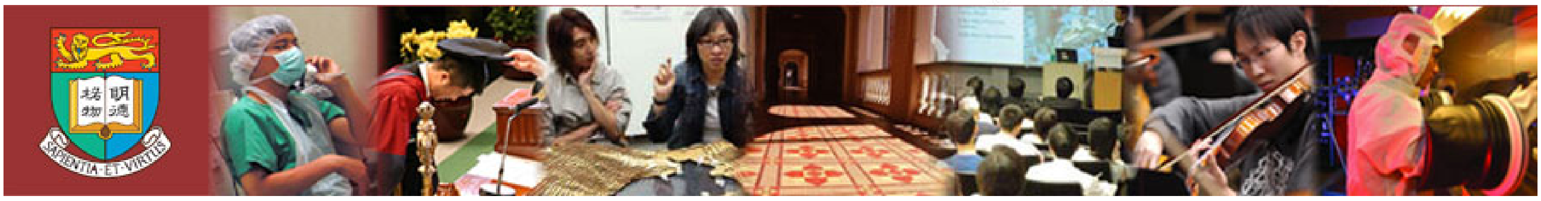

Title	Moving big data to the cloud
Author(s)	Zhang, L; Wu, C; Li, Z; Guo, C; Chen, M; Lau, FCM
Citation	The 32nd IEEE Conference on Computer Communications (IEEE INFOCOM 2013), Turin, Italy, 14-19 April 2013. In IEEE Infocom Proceedings, 2013, p. 405-409
Issued Date	2013
URL	http://hdl.handle.net/10722/186478
Rights	IEEE Infocom. Proceedings. Copyright © IEEE Computer Society.

Moving Big Data to The Cloud

Linquan Zhang*, Chuan Wu*, Zongpeng Li†, Chuanxiong Guo‡, Minghua Chen§ and Francis C.M. Lau*

*The University of Hong Kong, Hong Kong †University of Calgary, Canada
‡Microsoft Research Asia, China §The Chinese University of Hong Kong, Hong Kong

***Abstract*—Cloud computing, rapidly emerging as a new computation paradigm, provides agile and scalable resource access in a utility-like fashion, especially for the processing of big data. An important open issue here is how to efficiently move the data, from different geographical locations over time, into a cloud for effective processing. The *de facto* approach of hard drive shipping is not flexible, nor secure. This work studies timely, cost-minimizing upload of massive, dynamically-generated, geo-dispersed data into the cloud, for processing using a MapReduce-like framework. Targeting at a cloud encompassing disparate data centers, we model a cost-minimizing data migration problem, and propose two online algorithms, for optimizing at any given time the choice of the data center for data aggregation and processing, as well as the routes for transmitting data there. The first is an online lazy migration (OLM) algorithm achieving a competitive ratio of as low as 2.55, under typical system settings. The second is a randomized fixed horizon control (RFHC) algorithm achieving a competitive ratio of $1+\frac{1}{l+1}\frac{\kappa}{\lambda}$ with a lookahead window of l, where κ and λ are system parameters of similar magnitude.**

I. Introduction

The cloud computing paradigm enables rapid on-demand provisioning of server resources (CPU, storage, bandwidth) to users, with minimal management efforts. Recent cloud platforms, as exemplified by Amazon EC2 and S3, Microsoft Azure, Google App Engine, Rackspace, etc., organize a shared pool of servers from multiple data centers, and serve their users using virtualization technologies.

The elastic and on-demand nature of resource provisioning makes a cloud platform attractive for the execution of various applications, especially computation-intensive ones [1]. More and more data-intensive (big data) applications, *e.g.*, Facebook, Twitter, and big data analytics applications, such as the Human Genome Project [2], are relying on the clouds for processing and analyzing their petabyte-scale data sets, using a computing framework such as MapReduce and Hadoop [3].

An important issue however has largely been left out in this respect: How does one move the massive amounts of data into a cloud, in the very first place? The current practice is to copy the data into large hard drives for physical transportation to the data center [4], or even to move entire machines [5]. Such physical transportation incurs undesirable delay and possible service downtime, while outputs of the data analysis are often needed to be presented to users in the most timely fashion [5]. It is also less secure, given that the hard drives are prone to infection of malicious programs and damages from road accidents. A safer and more flexible data migration strategy is in need, to minimize any potential service downtime.

The research was supported in part by Hong Kong RGC General Research Fund (717812, 411209, 411010, and 411011), a China 973 Program (2012CB315904), an Area of Excellence Grant (AoE/E-02/08), and two gift grants from Microsoft and Cisco.

The challenge escalates when we consider that data are dynamically and continuously produced, from different geographical locations, *e.g.*, astronomical data from disparate observatories [6], user data from different Facebook front-end servers. For dynamically-generated data, an efficient *online algorithm* is desired, for timely guiding the transfer of data into the cloud over time; for geo-dispersed data sets, we wish to select the best data center to aggregate all data onto (*e.g.*, Amazon Elastic MapReduce launches all processing nodes in the same EC2 Availability Zone [7]), given that a MapReduce-like framework is most efficient when data to be processed are all in one place, and not across data centers due to the enormous overhead of inter-data center data moving in the stage of shuffle and reduce [8].

As the first dedicated effort in the cloud computing literature, this work studies timely, cost-minimizing migration of massive amounts of dynamically-generated, geo-dispersed data into the cloud, for processing using a MapReduce-like framework. Targeting a typical cloud platform that encompasses disparate data centers of different resource charges, we carefully model the cost-minimizing data migration problem, and propose efficient online algorithms, which optimize the routes of data into the cloud and the choice of the data center for data aggregation and processing, at any give time. Our detailed contributions are as follows:

▷ We analyze the detailed cost composition and identify the performance bottleneck for moving data into the cloud, and formulate an offline optimal data migration problem. The optimization computes optimal data routing and aggregation strategies at any given time, and minimizes the overall system cost and data transfer delay, over a long run of the system.

▷ Two online algorithms are proposed to practically guide data migration over time: an *online lazy migration (OLM)* algorithm and a *randomized fixed horizon control (RFHC)* algorithm. Theoretical analyses show that the OLM algorithm achieves a worst-case competitive ratio of 2.55, without the need of any future information and regardless of the system scale, under the typical settings in real-world scenarios. The RFHC algorithm achieves a competitive ratio of $1+\frac{1}{l+1}\frac{\kappa}{\lambda}$ that approaches 1 as the lookahead window l grows. Here κ and λ are system dependent parameters of similar magnitude.

▷ We conduct extensive experiments to evaluate the performance of our online algorithms, using real-world meteorological data generation traces. The online algorithms can achieve close-to-offline-optimum performance in most cases examined, revealing that the theoretical worst-case competitive ratios are pessimistic, and only correspond to rare scenarios in practice.

The remainder of this paper is organized as follows. We discuss related work in Sec. II, and present the system model and the offline optimal data migration problem in Sec. III. We design two online algorithms and analyze their competitive ratios in Sec. IV. Evaluation results are presented in Sec. V. Sec. VI concludes the paper.

II. Related work

The recent years have witnessed significant interest in migrating different applications onto the cloud platform. Hajjat *et al.* [9] develop an optimization model for migrating enterprise IT applications onto a hybrid cloud. Wu *et al.* [10] advocate deploying social media applications into clouds, for leveraging the rich resources and pay-as-you-go pricing. These projects focus on workflow migration and application performance optimization, by carefully deciding the modules to be moved to the cloud and the data caching/replication strategies in the cloud. The very rudimentary question of *how to move large volumes of application data into the cloud* however is not explored.

Few existing work discussed such transfer of large amounts of data to the cloud. Cho *et al.* [11] design Pandora, a cost-aware planning system for data transfer to the cloud provider, via both the Internet and courier services. Different from our study, they focus on static scenarios with a fixed amount of bulk data to transfer, rather than dynamically generated data; in addition, a single cloud site is considered, while our study considers multiple data centers.

A number of online algorithms have been proposed to address different cloud computing and data center issues. For online algorithms without future information, Lin *et al.* [12] investigate energy-aware dynamic server provisioning, by proposing a Lazy Capacity Provisioning algorithm with a 3-competitive ratio. Assuming lookahead into the future, Lu and Chen [13] study the dynamic provisioning problem in data centers, design future-aware algorithms based on the classic ski-rental online algorithm. Lin *et al.* [14] investigate load balancing among geographically-distributed data centers with a receding horizon control (RHC) algorithm, and show that the competitive ratio can be reduced substantially by leveraging the predicted future information.

III. The Data Migration Problem

A. System Model

Consider a cloud with K data centers distributed in a set of regions $\mathcal{K}$ ($K = |\mathcal{K}|$). A user (*e.g.*, a global astronomical telescopes application) continuously produces large volumes of data at a set $\mathcal{D}$ of geographic locations (*e.g.*, dispersed telescope sites). The user connects to the data centers from different data generation locations via VPNs, with G VPN gateways ($\mathcal{G}$) at the user side and K VPN gateways each collocated with a data center (Fig. 1). A private network of the user inter-connects all the data generation locations and the VPN gateways at the user side. Such a model reflects typical connection approaches between users and public clouds (*e.g.*, AWS Direct Connect [15]), where dedicated, private network connections are established between a user's premise and the cloud, for enhanced security and reliability, and guaranteed inter-connection bandwidth.

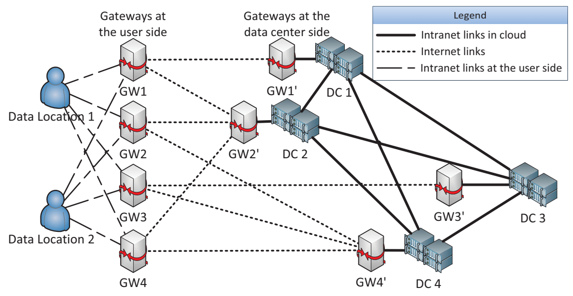

Fig. 1. An illustration of the cloud system.

While intra-cloud links and links in the private network are usually over provisioned, the bandwidth U_{gi} on a VPN link (g, i) from user side gateway g to data center i is limited, and constitutes the bottleneck in the system.

B. Cost-minimizing Data Migration: Problem Formulation

We consider a time-slotted system with slot length τ. $F_d(t)$ bytes of data are produced at location d in slot t. l_{dg} is the latency between data location $d \in \mathcal{D}$ and gateway $g \in \mathcal{G}$, p_{gi} is the delay along VPN link (g, i), and η_{ik} is the latency between data centers i and k. These delays are dictated by the respective geographic distances.

A cloud user faces the problem of deciding (i) via which VPN connections to upload the data into the cloud, and (ii) to which data center should they be aggregated, for processing by a MapReduce-like framework, such that the monetary charges incurred, as well as the latency for the data to reach the aggregation point, are minimized.

Decision variables. (1) Data routing variable $x_{d,g,i,k}(t)$ denotes the portion of data $F_d(t)$ produced at location d in t, to be uploaded through VPN connection (g, i) and then migrated to data center k for processing. $x_{d,g,i,k}(t) > 0$ indicates that the data routing path $d \to g \to i \to k$ is employed, and $x_{d,g,i,k} = 0$ otherwise. Let $\vec{x} = (x_{d,g,i,k}(t))_{\forall d,g,i,k}$, the set of feasible data routing variables are:

$$\mathcal{X} = \Big\{ \vec{x}(t) \mid \sum_{g\in\mathcal{G}, i\in\mathcal{K}, k\in\mathcal{K}} x_{d,g,i,k}(t) = 1 \text{ and } x_{d,g,i,k} \in [0,1], \\ \forall d \in \mathcal{D}, \forall g \in \mathcal{G}, \forall i \in \mathcal{K}, \forall k \in \mathcal{K} \Big\}. \quad (1)$$

Here $\sum_{g,i,k} x_{d,g,i,k}(t) = 1$ ensures that all data produced from location d are uploaded into the cloud in t.

(2) Binary variable $y_k(t)$ indicates whether data center k is target of data aggregation in time slot t ($y_k(t) = 1$) or not ($y_k(t) = 0$). At any given time, exactly one data center is chosen. Let $\vec{y}(t) = (y_k(t))_{\forall k\in\mathcal{K}}$, the set of possible data aggregation variables are:

$$\mathcal{Y} = \Big\{ \vec{y}(t) \mid \sum_{k\in\mathcal{K}} y_k(t) = 1 \text{ and } y_k(t) \in \{0,1\}, \forall k \in \mathcal{K} \Big\}. \quad (2)$$

Costs. The costs incurred in time slot t include the following components.

(1) The overall *bandwidth cost* for uploading data via the VPN connections, where $\sum_{d\in\mathcal{D},k\in\mathcal{K}} F_d(t)x_{d,g,i,k}(t)$ is the amount uploaded via VPN connection (g,i), and f_{gi} is the unit charge for uploading one byte of data via (g,i):

$$C_{BW}(\vec{x}(t)) \triangleq \sum_{g\in\mathcal{G},i\in\mathcal{K}} (f_{gi} \sum_{d\in\mathcal{D},k\in\mathcal{K}} F_d(t)x_{d,g,i,k}(t)). \quad (3)$$

(2) *Storage and computation costs* are important factors to consider in choosing the data aggregation point. In a large-scale online application, processing and analyzing in t may involve data produced not only in t, but also from the past, in the form of raw data or intermediate processing results [16]. Without loss of generality, let the amount of current and history data to process in t be $\mathcal{F}(t) = \sum_{\nu=1}^{t}(\alpha_\nu \sum_{d\in\mathcal{D}} F_d(\nu))$, where $\sum_{d\in\mathcal{D}} F_d(\nu)$ is the total amount of data produced in time slot ν from different data generation locations, and weight $\alpha_\nu \in [0,1]$ is smaller for older times ν and $\alpha_t = 1$ for the current time t. Assume all the other historical data, except those in $\mathcal{F}(t)$, are removed from the data centers where they were processed. Let $\Psi_k(\mathcal{F}(t))$ be a non-decreasing, convex cost function for storage and computation in data center k in t. The aggregate storage and computing cost in t is:

$$C_{DC}(\vec{y}(t)) \triangleq \sum_{k\in\mathcal{K}} y_k(t)\Psi_k(\mathcal{F}(t)). \quad (4)$$

(3) The best data center for data aggregation could differ in t than in $t-1$, due to temporal and spatial variations in data generation. Historical data needed for processing together with the new data in t, at the amount of $\sum_{\nu=1}^{t-1}(\alpha_\nu \sum_{d\in\mathcal{D}} F_d(\nu))$, should be moved from the former data center to the current. Let $\phi_{ik}(z)$ be the non-decreasing, convex migration cost to move z bytes of data from data center i to date center k, satisfying triangle inequality: $\phi_{ik}(z) + \phi_{kj}(z) \geq \phi_{ij}(z)$. The *migration cost* between time slot $t-1$ and time slot t is:

$$\begin{aligned} C^t_{MG}(\vec{y}(t), \vec{y}(t-1)) \triangleq \sum_{i\in\mathcal{K}}\sum_{k\in\mathcal{K}}([y_i(t-1) - y_i(t)]^+ \\ [y_k(t) - y_k(t-1)]^+ \phi_{ik}(\sum_{\nu=1}^{t-1}\alpha_\nu \sum_{d\in\mathcal{D}} F_d(\nu))). \end{aligned} \quad (5)$$

Here $[a-b]^+ = \max\{a-b, 0\}$.

(4) We use a *routing cost* to model delays along the selected routing paths:

$$C_{RT}(\vec{x}(t)) \triangleq \sum_{d,g,i,k} Lx_{d,g,i,k}(t)F_d(t)(l_{dg} + p_{gi} + \eta_{ik}), \quad (6)$$

where $x_{d,g,i,k}(t)F_d(t)(l_{dg} + p_{gi} + \eta_{ik})$ is the product of data volume and delay along the routing path $d \rightarrow g \rightarrow i \rightarrow k$. L is the routing cost weight converting $x_{d,g,i,k}(t)F_d(t)(l_{dg} + p_{gi} + \eta_{ik})$ into a monetary cost, reflecting how latency-sensitive the user is. In this work, L is a constant provided by the user *a priori*. The latency $l_{dg} + p_{gi} + \eta_{ik}$ is fixed in each time slot, but can change over time.

In summary, the overall cost incurred in t in the system is:

$$\begin{aligned} \mathbb{C}(\vec{x}(t), \vec{y}(t)) = & C_{BW}(\vec{x}(t)) + C_{DC}(\vec{y}(t)) + \\ & C^t_{MG}(\vec{y}(t), \vec{y}(t-1)) + C_{RT}(\vec{x}(t)). \end{aligned} \quad (7)$$

The offline optimization problem. The optimization problem of minimizing the overall cost of data upload and processing over a time interval $[1,T]$, can be formulated as:

$$\text{minimize} \quad \sum_{t=1}^{T} \mathbb{C}(\vec{x}(t), \vec{y}(t)) \quad (8)$$

subject to: $\forall t = 1,\ldots,T$,

$(8a)$ $\vec{x}(t) \in \mathcal{X}$,
$(8b)$ $\sum_{d\in\mathcal{K},k\in\mathcal{K}} F_d(t)x_{d,g,i,k}(t)/\tau \leq U_{gi}, \forall i \in \mathcal{K}, \forall g \in \mathcal{G}$,
$(8c)$ $x_{d,g,i,k}(t) \leq y_k(t), \forall d \in \mathcal{D}, \forall g \in \mathcal{G}, \forall i \in \mathcal{K}, \forall k \in \mathcal{K}$,
$(8d)$ $\vec{y}(t) \in \mathcal{Y}$,

Constraint $(8b)$ states that the total amount of data routed via (g,i) into the cloud in each time slot should not exceed the upload capacity of (g,i). $(8c)$ ensures that a routing path $d\rightarrow g\rightarrow i\rightarrow k$ is used ($x_{d,g,i,k}(t) > 0$), only if data center k is the point of data aggregation in t ($y_k(t) = 1$).

IV. Two Online Algorithms

We next design two online algorithms for guiding data routing and aggregation over time. The first algorithm relies only on the current and historical information, and the second further exploits predicted information from the future.

A. The OLM Algorithm

The offline optimization problem in (8) can be divided into T one-shot optimization problems at each time t:

$$\text{minimize } \mathbb{C}(\vec{x}(t), \vec{y}(t)) \quad \text{subject to: } (8a)(8b)(8c)(8d). \quad (9)$$

A naive online algorithm that solves (9) in each time slot can be far from optimal, migrating data back and forth prematurely. We design a more judicious online solution by exploring the inter-slot dependencies for data center selection.

We divide the overall cost $\mathbb{C}(\vec{x}(t), \vec{y}(t))$ into: (i) migration cost $C^t_{MG}(\vec{y}(t), \vec{y}(t-1))$ defined in (5), related to decisions in $t-1$; and (ii) non-migration cost that relies only on current information at t:

$$C^t_{-MG}(\vec{x}(t), \vec{y}(t)) = C_{BW}(\vec{x}(t)) + C_{DC}(\vec{y}(t)) + C_{RT}(\vec{x}(t)). \quad (10)$$

We design a lazy migration algorithm that postpones data center switching indicated by the one-shot optimum, until the cumulative non-migration cost (in $C^t_{-MG}(\vec{x}(t), \vec{y}(t))$) significantly exceeds the potential data migration cost.

As shown in Alg. 1, we solve the one-shot optimization in (9) at $t = 1$, and obtain the optimal data center indicted by $\vec{y}(1)$, the optimal routes $\vec{x}(1)$. Let $\hat{t}$ be the time of the data center switch. In each following time slot t, we compute the overall non-migration cost in $[\hat{t}, t-1]$, $\sum_{\nu=\hat{t}}^{t-1} C^{\nu}_{-MG}(\vec{x}(\nu), \vec{y}(\nu))$.The algorithm checks whether this cost is at least β_2 times the migration cost $C^{\hat{t}}_{MG}(\vec{y}(\hat{t}), \vec{y}(\hat{t}-1))$. If so, it solves the one-shot optimization to derive $\vec{x}(t)$ and $\vec{y}(t)$ without considering the migration cost, *i.e.*, by minimizing $C^t_{-MG}(\vec{x}(t), \vec{y}(t))$ subject to $(8a) - (8d)$ and an additional constraint, that the potential migration cost, $C^t_{MG}(\vec{y}(t), \vec{y}(t-1))$, is no larger than β_1 times the non-migration cost $C^t_{-MG}(\vec{x}(t), \vec{y}(t))$ at time t. If a change of migration data center is indicated ($\vec{y}(t) \neq \vec{y}(t-1)$), the

Algorithm 1 The Online Lazy Migration (OLM) Algorithm

```
1: $t = 1$;
2: $\hat{t} = 1$;     //Time slot when the last change of aggregation data center happens
3: Compute data routing decision $\vec{x}(1)$ and aggregation decision $\vec{y}(1)$ by minimizing $\mathbb{C}(\vec{x}(1), \vec{y}(1))$ subject to $(8a) - (8d)$;
4: Compute $C^1_{MG}(\vec{y}(1), \vec{y}(0))$ and $C^1_{-MG}(\vec{x}(1), \vec{y}(1))$;
5: while $t \leq T$ do
6:     if $C^{\hat{t}}_{MG}(\vec{y}(\hat{t}), \vec{y}(\hat{t}-1)) \leq \frac{1}{\beta_2} \sum_{\nu=\hat{t}}^{t-1} C^{\nu}_{-MG}(\vec{x}(\nu), \vec{y}(\nu))$ then
7:         Derive $\vec{x}(t)$ and $\vec{y}(t)$ by minimizing $C^t_{-MG}(\vec{x}(t), \vec{y}(t))$ in (10) subject to $(8a) - (8d)$ and constraint $C^t_{MG}(\vec{y}(t), \vec{y}(t-1)) \leq \beta_1 C^t_{-MG}(\vec{x}(t), \vec{y}(t))$;
8:         if $\vec{y}(t) \neq \vec{y}(t-1)$ then
9:             Use the new aggregation data center indicated by $\vec{y}(t)$;
10:            $\hat{t} = t$;
11:    if $\hat{t} < t$ then //not to use new aggregation data center
12:        $\vec{y}(t) = \vec{y}(t-1)$, compute data routing decision $\vec{x}(t)$ by solving (9) if not derived;
13:    $t = t + 1$;
```

algorithm accepts the new aggregation decision, and migrates data accordingly. Otherwise, the aggregation point remains unchanged, and only data routing paths are computed.

In Alg. 1, β_2 and β_1 reflect the "laziness" and "aggressiveness" of the algorithm: a larger β_2 prolongs the inter-switch interval of the aggregation data center, while a larger β_1 invites more frequent switches. We next analyze the competitive ratio of the OLM algorithm, *i.e.*, the ratio of the worst-case total cost incurred by the OLM algorithm in $[1, T]$, over that of the offline optimal algorithm.

Lemma 1. *The overall migration cost in* $[1, t]$ *is at most* $\max\{\beta_1, 1/\beta_2\}$ *times the overall non-migration cost in this period,* i.e., $\sum_{\nu=1}^{t} C^{\nu}_{MG}(\vec{y}(\nu), \vec{y}(\nu - 1)) \leq \max\{\beta_1, 1/\beta_2\} \sum_{\nu=1}^{t} C^{\nu}_{-MG}(\vec{x}(\nu), \vec{y}(\nu))$.

Lemma 2. *The overall non-migration cost in* $[1, t]$ *is at most* ϵ *times the total offline-optimal cost,* i.e., $\sum_{\nu=1}^{t} C^{\nu}_{-MG}(\vec{x}(\nu), \vec{y}(\nu)) \leq \epsilon \sum_{\nu=1}^{t} \mathbb{C}(\vec{x}^*(\nu), \vec{y}^*(\nu))$, *where*

$$\epsilon = \max_{\nu \in [1,T]} \frac{\max_{\vec{y}(\nu) \in \mathcal{Y}, \vec{x}(\nu) : (8a)-(8c)} C^{\nu}_{-MG}(\vec{x}(\nu), \vec{y}(\nu))}{\min_{\vec{y}(\nu) \in \mathcal{Y}, \vec{x}(\nu) : (8a)-(8c)} C^{\nu}_{-MG}(\vec{x}(\nu), \vec{y}(\nu))}$$

is the maximum ratio of the largest over the smallest possible non-migration cost incurred in a time slot, with different data upload and aggregation decisions.

Theorem 1. *The OLM Algorithm is* $\epsilon(1 + \max\{\beta_1, 1/\beta_2\})$*-competitive.*

Detailed proofs of the lemmas and the theorem are in our technical report [17]. The value of ϵ depends more on data generation patterns over time, and less on system scale. Under a typical value $\epsilon = 1.7$ from our experiments, setting $\beta_1 = 0.5$ and $\beta_2 = 2$ leads to a competitive ratio of 2.55.

B. The Randomized Fixed Horizon Control (RFHC) Algorithm

In practical applications, near-term future data generation patterns can often be estimated from history, *e.g.*, using a time series forecasting model [18]. We next design an algorithm that exploits such future information.

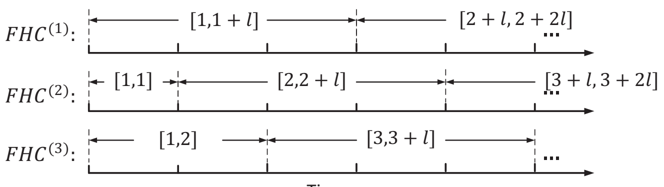

Fig. 2. An illustration of different FHC algorithms with $l = 2$.

We divide time into equal-size frames of $l + 1$ time slots each ($l \geq 0$). In the first time slot t of each frame, suppose we can predict all future information on data generation for the next l time slots: $F_d(t), F_d(t+1), ..., F_d(t+l), \forall d \in \mathcal{D}$. We solve the following cost minimization problem over $[t, t+l]$ — given $\vec{y}(t-1)$, to derive $\vec{x}(\nu)$ and $\vec{y}(\nu)$, $\forall \nu = t, \ldots, t+l$:

$$\text{minimize} \sum_{\nu=t}^{t+l} \mathbb{C}(\vec{x}(\nu), \vec{y}(\nu)), \tag{11}$$

subject to: constraints (8a)—(8d), for $\nu = t, \ldots, t+l$.

The method is essentially a fixed horizon control (FHC) algorithm, adapted from receding horizon control in the dynamic resource allocation literature [14]. Allowing the first time frame (of $l+1$ slots) to start from different initial times $p \in [1, l+1]$, we have $l+1$ versions of the FHC algorithm (Fig. 2). In particular, for $FHC^{(p)}$ starting from slot p, (11) is solved at $t = p$, $p+l+1$, $p+2(l+1), \ldots$, for routing and aggregation decisions in the following $l+1$ time slots. For each $FHC^{(p)}$, an adversary can tailor an input with a surge of data produced at the beginning of each time frame. A high migration cost is likely to occur at each frame start, since the surge was not considered by the decision making in the previous frame. A randomized algorithm defeats such adversaries by randomizing the starting times of the frames.

Alg. 2 shows our Randomized Fixed Horizon Control (RFHC) algorithm. It first uniformly randomly chooses $p \in [1, l+1]$ as the start of the first time frame of $l+1$ slots, *i.e.*, it randomly picks one specific algorithm $FHC^{(p)}$ from the $l+1$ finite horizon control algorithms: at $t = 1$, it solves (11) to decide the optimal data routing and aggregation strategies in the period of $t = 1$ to $p-1$ ($p \neq 1$); then at $t = p$, $p+l+1$, $p+2(l+1)$, …, it solves (11) for optimal strategies in the following $l+1$ time slots, respectively.

Algorithm 2 The RFHC Algorithm

```
1: $\vec{y}(0) = 0$;
2: $p = rand(1, l+1)$;     //A random integer within [1,l+1]
3: if $p \neq 1$ then
4:     Derive $\vec{x}(1) \cdots \vec{x}(p-1)$ and $\vec{y}(1) \cdots \vec{y}(p-1)$ by solving (11) over the time window $[1, p-1]$;
5: $t = p$;
6: while $t \leq T$ do
7:     if $(t-p) \bmod (l+1) = 0$ then
8:         Derive $\vec{x}(t), \cdots, \vec{x}(t+l)$ and $\vec{y}(t), \cdots, \vec{y}(t+l)$ by solving (11) over the time frame $[t, t+l]$;
9:     $t = t + 1$;
```

Lemma 3. *The overall cost incurred by* $FHC^{(p)}$ *is upper-bounded by the offline-optimal cost plus the migration costs*

TABLE I
PERFORMANCE COMPARISON AMONG THE ALGORITHMS: SPOT INSTANCE PRICING, $P = 0.25$, $L = 0.01$

	Simple	OLM	RFHC(0)	RFHC(1)	Offline
Overall cost (\$)	24709	16870	16676	16327	15944
Ratio	1.55	1.06	1.05	1.02	1

to move data from the aggregation data center computed by $FHC^{(p)}$ to the optimal one, at the end of the time frames. That is, letting $\vec{x}^{(p)}$ and $\vec{y}^{(p)}$ be the solution derived by the $FHC^{(p)}$ algorithm and $\Omega_{p,t} = \{\omega | \omega = p + k(l+1), k = 0, 1, \ldots, \lfloor \frac{t-p}{l+1} \rfloor\}$, we have for any $t \in [1, T]$,

$$\sum_{\nu=1}^{t} \mathbb{C}(\vec{x}^p(\nu), \vec{y}^p(\nu)) \leq \sum_{\nu=1}^{t} \mathbb{C}(\vec{x}^*(\nu), \vec{y}^*(\nu)) + \sum_{\omega \in \Omega_{p,t}} C_{MG}^{\omega}(\vec{y}^*(\omega-1), \vec{y}^{(p)}(\omega-1)).$$

Theorem 2. *The RFHC algorithm is $(1+\frac{1}{l+1}\frac{\kappa}{\lambda})$-competitive. Here l is the number of lookahead steps, $\kappa = \sup_{t\in[1,T],\vec{y}^1(t),\vec{y}^2(t)\in\mathcal{Y}} \frac{C_{MG}^t(\vec{y}^1(t),\vec{y}^2(t))}{\sum_{\nu=1}^{t-1}(\alpha_\nu \sum_{d\in\mathcal{D}} F_d(\nu))}$ is the maximum migration cost per unit data, and $\lambda = \inf_{t\in[1,T],\vec{x}(t),\vec{y}(t):(8a)-(8d)} \frac{\mathbb{C}(\vec{x}(t),\vec{y}(t))}{\sum_{\nu=1}^{t-1}(\alpha_\nu \sum_{d\in\mathcal{D}} F_d(\nu))}$ is the minimum total cost per unit data per time slot.*

The proofs of the lemma and theorem above can be found in our technical report [17]. Theorem 2 reveals that the more future steps predicted (the larger l is), the closer the RFHC algorithm can approach the offline optimum. Values of κ and λ are related to system input including prices and delays, and are less involved with the data generation patterns and the number of data centers. Under the practical settings used in our experiments, $\frac{\kappa}{\lambda} \approx 0.69$. In this case, even with $l = 1$, the competitive ratio is already as low as 1.34.

V. PERFORMANCE EVALUATION

Due to space limitations, detailed experiment set up and more empirical results are provided in our technical report [17]. We have implemented and compared our offline and online algorithms, as well as a Pandora-like simple algorithm that fixes the choice of the data center for aggregation to be the one in Hong Kong.

We investigate dynamical VM prices (time-varying data processing costs) following the Spot Instance prices from Amazon EC2 during Apr. 2012 to Jul. 2012. From Tab. I, we see that due to lack of future price information, the OLM algorithm performs slightly worse than the RFHC algorithm with lookahead window $l = 1$.

We also evaluate the RFHC algorithm with different lookahead window sizes, with dynamic prices. Tab. II shows that, with the increase of the lookahead window, the performance is approaching the offline optimum. Just a 1- or 2-step 'peek' into the future drives the performance very close to the offline optimum.

VI. CONCLUDING REMARKS

This paper designs efficient algorithms for timely, cost-minimizing migration of enormous amounts of dynamically-generated, geo-dispersed data into the cloud, for processing using a MapReduce-like framework. Two novel online algorithms are designed to practically guide data migration in an online fashion, based on solid theoretical analysis. The OLM algorithm achieves a worst-case competitive ratio of as low as 2.55 under typical real-world settings, without the need of any future information; the RFHC algorithm provides a decreasing competitive ratio with increasing size of the lookahead window. Our extensive experiments reveal the close-to-offline-optimum performance of both algorithms, by comparing them with a simple algorithm and the optimal offline algorithm, under real-world meteorological data generation patterns.

TABLE II
PERFORMANCE OF RFHC ALGORITHMS WITH DIFFERENT LOOKAHEAD WINDOW SIZES: SPOT INSTANCE PRICING, $P = 0.25$, $L = 0.01$

	RFHC(0)	RFHC(1)	RFHC(2)	RFHC(3)	Offline
Overall cost (\$)	16676	16327	16105	16138	15944
Ratio	1.05	1.02	1.01	1.01	1

REFERENCES

[1] M. Armbrust, A. Fox, R. Grifth, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. P. A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing," EECS, University of California, Berkeley, Tech. Rep., 2009.

[2] *Human Genome Project*, http://www.ornl.gov/hgmis/home.shtml.

[3] *Hadoop at Twitter*, http://www.slideshare.net/kevinweil/hadoop-at-twitter-hadoop-summit-201.

[4] *AWS Import/Export*, http://aws.amazon.com/importexport/.

[5] *Moving an Elephant: Large Scale Hadoop Data Migration at Facebook*, http://www.facebook.com/notes/paul-yang/moving-an-elephant-large-scale-hadoop-data-migration-at-facebook/10150246275318920.

[6] R. J. Brunner, S. G. Djorgovski, T. A. Prince, and A. S. Szalay, "Handbook of Massive Data Sets," J. Abello, P. M. Pardalos, and M. G. C. Resende, Eds. Norwell, MA, USA: Kluwer Academic Publishers, 2002, ch. Massive Datasets in Astronomy, pp. 931–979.

[7] *Amazon Elastic MapReduce*, http://aws.amazon.com/elasticmapreduce/.

[8] M. Cardosa, C. Wang, A. Nangia, A. Chandra, and J. Weissman, "Exploring mapreduce efficiency with highly-distributed data," in *Proc. of MapReduce 2011*, 2011.

[9] M. Hajjat, X. Sun, Y. E. Sung, D. Maltz, and S. Rao, "Cloudward Bound: Planning for Beneficial Migration of Enterprise Applications to the Cloud," in *Proc. of ACM SIGCOMM*, August 2010.

[10] Y. Wu, C. Wu, B. Li, L. Zhang, Z. Li, and F. Lau, "Scaling Social Media Applications into Geo-Distributed Clouds," in *Proc. of IEEE INFOCOM*, Mar. 2012.

[11] B. Cho and I. Gupta, "New Algorithms for Planning Bulk Transfer via Internet and Shipping Networks," in *Proc. of IEEE ICDCS*, 2010.

[12] M. Lin, A. Wierman, L. L. Andrew, and E. Thereska, "Dynamic Right-sizing for Power-proportional Data Centers," in *Proc. of IEEE INFOCOM*, April 2011.

[13] T. Lu and M. Chen, "Simple and Effective Dynamic Provisioning for Power-Proportional Data Centers," in *Proc. of IEEE CISS*, Mar. 2012.

[14] M. Lin, Z. Liu, A. Wierman, and L. Andrew, "Online Algorithms for Geographical Load Balancing," in *Proc. of IEEE IGCC*, 2012.

[15] *Amazone Web Services*, http://aws.amazon.com/.

[16] D. Logothetis, C. Olston, B. Reed, K. C. Webb, and K. Yocum, "Stateful Bulk Processing for Incremental Analytics," in *Proc. of ACM SoCC*, 2010.

[17] L. Zhang, C. Wu, Z. Li, C. Guo, M. Chen, and F. C. M. Lau, "Move My Data to the Cloud: an Online Cost-Minimizing Approach," `http://i.cs.hku.hk/~cwu/movedata.pdf`, Tech. Rep.

[18] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*, 4th ed. Wiley, 2008.