

Trusted Support Service and OTP method for Moving Big Data to Cloud

^[1]Muhsina.B.S, ^[2]PreethikaRajashekhar, ^[3]Supriya.V, ^[4]ThusleenaKabeer, ^[5]Navin K.S
LBS Institute of Technology for Women, Thiruvananthapuram, Kerala, India muhsinabs@gmail.com,
^[1]preethz2408@gmail.com, ^[2]suppuvl@gmail.com, ^[3]thas235@gmail.com, ^[4]navinsivendran@gmail.com

Abstract: Cloud computing is a mechanism in which company hires the data centre and hosts the confidential information for computing the various services in low cost infrastructure, on demand and dynamic provisioning. An important open issue here is 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 or secure. This work studies timely, cost-minimizing upload of massive, dynamically-generated, geo-dispersed data into the cloud, for processing using a Map Reduce-like framework. But it outstands with security concerns. Security requirements for cloud computing environment should have trusted computing platform. The system proposes cloud computing system is combined with Trusted Support Service (TSS). In this design, better user authentication can be obtained by using One Time Password (OTP) which will sent to user via email and SMS. The system also provides the Trusted Support Service (TSS) by using Java/C editor. Paper also demonstrates the performance comparison of this system. For cost-minimizing data migration problem, we propose two online algorithms: an online lazy migration (OLM) algorithm and a randomized fixed horizon control (RFHC) algorithm, for optimizing at any given time the choice of the data centre for data aggregation and processing, as well as the routes for transmitting data there. Here we are demonstrating this concept through a banking application

Keywords: Cloud Computing, Big Data, and Online algorithms

I. INTRODUCTION

Cloud computing is a powerful technology to perform massive-scale and complex computing. It eliminates the need to maintain expensive computing hardware, dedicated space, and software. 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], [2], [3]. More and more data-intensive Internet applications, e.g., Facebook, Twitter, and big data analytics applications, such as the Human Genome Project [4], are relying on the clouds for processing and analyzing their petabyte-scale datasets, with a computing framework such as MapReduce and Hadoop. Massive growth in the scale of data or big data generated through cloud computing has been observed. Addressing big data is a challenging and time-demanding task that requires a large computational infrastructure to ensure successful data processing and analysis

While most efforts have been devoted to designing better computing models for big data analytics, an important issue has largely been left out in this respect. The current practice to move the massive amounts of data into a cloud, is to copy the data into large hard drives for physically transportation to the data center or even to move entire

machines. 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. 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 challenge escalates when we target at dynamically and continuously produced data from different geographical locations. With dynamic 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 can select the best data center to aggregate all data onto, for processing with a MapReduce-like framework, which is efficient to process data within *one* data center but not across data centers, due to the enormous overhead of inter-data center data moving in the stage of shuffle and reduce. Two efficient 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 $1 + (1/l+1)(\kappa/\lambda)$ that approaches 1 as the lookahead window l grows. Here κ and λ are system dependent parameters of similar magnitudes.

For better user authentication we propose OTP in our system. This authentication system will ensure authorized access to information building trust in cloud users. The proposed system application will be Integrated Development Environment (IDE) for Java/C source code compilation, debugging and execution facilities. Only registered users can use this cloud service. To ensure information security, system will implement email OTP authentication technique. For every login, an OTP will be sent to registered email id/mobile no. The cloud user has to fetch the OTP to email account/mobile and then only he can access cloud services like Java/C source code compilation, debugging and execution.

In the rest of the paper, we discuss related work in Sec. II, describe the system and problem models in Sec. III, and present the online solutions in Sec. IV, Sec V describes the security methods that can be provided and Sec. 6 concludes the paper.

II. RELATED WORK

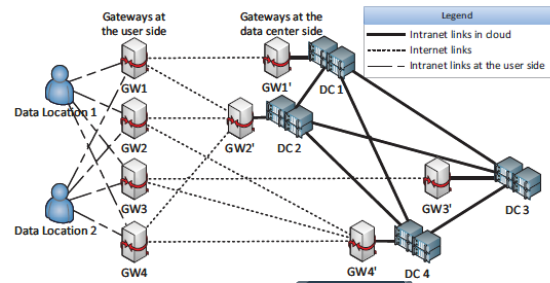
A series of recent work studies application migration to the cloud. Hajjat et al. [5] develop an optimization model for migrating enterprise IT applications onto a hybrid cloud. Cheng et al. [6] and Wu et al. [7] advocate deploying social media applications into clouds, for leveraging the rich resources and pay-as-you-go pricing. These projects focus mainly 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 first question of how to move large volumes of application data into the cloud is not explored. Few existing work discussed such transfer of big data to the cloud. Cho et al. [8] 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 pays attention to multiple data centers.

III. THE DATA MIGRATION PROBLEM

A. System Model

We can consider a cloud consisting of K geo-distributed data centers in a set of regions. A cloud user continuously produces large volumes of data at a set D of multiple

geographic locations. The user connects to the data centers from different data generation locations via virtual private networks (VPNs), with G



VPN gateways at the user side and K VPN gateways each collocated with a data center.

B. Cost-minimizing Data Migration: Problem Formulation

Assume the system executes in a time-slotted fashion with slot length τ . $F_d(t)$ bytes of data are produced at location d in slot t , for upload to the cloud. l_{dg} is the latency between data location $d \in D$ and user side gateway $g \in G$, p_{gi} is the delay along VPN link (g,i) , and m_{ik} is the latency between data centers i and k . These delays, which can be obtained by a simple command such as ping, are dictated by the respective geographic distances. A cloud user needs to decide (i) via which VPN connections to upload its data to the cloud, and (ii) to which data center to aggregate data, 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 jointly minimized. The total cost C to be minimized has four components: bandwidth cost, aggregate storage and computing cost, migration cost, and routing cost.

Decision variables. Two types of optimization variables are formulated:

- (1) Data routing variable $x_{d,g,i,k}(t), \forall d \in D, \forall g \in G, \forall i \in K, \forall k \in K$, 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 $d \rightarrow g \rightarrow i \rightarrow k$ is employed, and $x_{d,g,i,k} = 0$ otherwise.

Let $\vec{x}^t = (x_{d,g,i,k}(t)) \forall d \in D, \forall g \in G, \forall i \in K, \forall k \in K$, the set of feasible data routing variables are:

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

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

- (2) Binary variable $y_k(t), \forall k \in K$, indicates whether data center k is target of data aggregation in time slot t ($y_k(t) = 1$) or not ($y_k(t) = 0$). Following the practical requirement of the current MapReduce framework, we require that at any given

time, exactly one data center is chosen. Let $y^{\rightarrow}(t) = (y_k(t)) \forall k \in K$, the set of possible data aggregation variables are:

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

Costs. The costs incurred in time slot t , for uploading the data into the cloud and for processing the data at the selected data center, include the following components.

(1) *The overall bandwidth cost* for uploading data via the VPN connections, where $d \in D, k \in K \sum_{d \in D} x_{d,g,i,k}(t)$ is the amount uploaded via (g,i) , and f_{gi} is the charge for uploading one byte of data via (g,i) , derived from bandwidth prices set by the cloud provider:

$$CBW(x^{\rightarrow}(t)) \triangleq \sum_{g \in G, i \in K} (f_{gi} \sum_{d \in D, k \in K} F_d(t) x_{d,g,i,k}(t)). \quad (3)$$

(2) *Storage and computation costs* are important factors to consider in choosing the data aggregation point. The data produced in t and also from the past which is in the form of raw data or intermediate processing results may involve in processing and analysing. Without loss of generality, let the amount of current and history data to process in t be $F(t) = \sum_{v=1}^t (\alpha v \sum_{d \in D} F_d(v))$, where $\sum_{d \in D} F_d(v)$ is the total amount of data produced in time slot v from different data generation locations, and weight $\alpha v \in [0, 1]$ is smaller for older times v and $\alpha t = 1$ for the current time t . Specific applications determines the value of αv and it can be obtained through statistical data. Assume all the other historical data, except those in $F(t)$, are removed from the data centers where they were processed, since all needed information has been stored in the retained data. Let $\Psi_k(F(t))$ be a non-decreasing cost function for storage and computation in data center k in t . The aggregate storage and computing cost incurred in the cloud in t is:

$$CDC(y^{\rightarrow}(t)) \triangleq \sum_{k \in K} y_k(t) \Psi_k(F(t)). \quad (4)$$

The best data center for data aggregation can differ in t than in $t-1$, due to temporal and spatial variations in data generation. Historical data needed for processing together with new data in t , at the amount of $\sum_{v=1}^t (\alpha v \sum_{d \in D} F_d(v))$, should be moved from the earlier data center to the current, and a *migration cost* is incurred. Let $\phi_{ik}(z)$ be the non-decreasing migration cost to move z bytes of data from data center i to data center k . The migration cost between $t-1$ and t is:

$$C_{iMG}(y^{\rightarrow}(t), y^{\rightarrow}(t-1)) \triangleq \sum_{i \in K} \sum_{k \in K} ([y_i(t-1) - y_i(t)] +$$

$$[y_k(t) - y_k(t-1)] + \phi_{ik}(\sum_{v=1}^t (\alpha v \sum_{d \in D} F_d(v)))) \quad (5)$$

(4) The latency incurred for data upload is an important performance measure, to be minimized in the data routing and aggregation process. Targeting both monetary cost minimization and delay minimization, a routing cost for delays along the selected routing paths is formulated, and

combine it with other costs to be the optimization objective. The overall routing cost in the system in t is:

$$CRT(x^{\rightarrow}(t)) \triangleq d, g, i, k L x_{d,g,i,k}(t) F_d(t) (l_{dg} + p_{gi} + \eta_{ik}). \quad (6)$$

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 $d \rightarrow g \rightarrow i \rightarrow k$. The weighted formula suggests that transferring a large volume of data via a high latency path causes high cost. 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. A cloud user specifies L as a constant a priori. Latency $l_{dg} + p_{gi} + \eta_{ik}$ is fixed in each slot but can change over time. In summary, the overall cost incurred in t in the system is:

$$C(x^{\rightarrow}(t), y^{\rightarrow}(t)) = CBW(x^{\rightarrow}(t)) + CDC(y^{\rightarrow}(t)) + C_{iMG}(y^{\rightarrow}(t), y^{\rightarrow}(t-1)) + CRT(x^{\rightarrow}(t)). \quad (7)$$

IV. TWO ONLINE ALGORITHMS

A. The Online Lazy Migration (OLM) Algorithm

We divide the overall cost $C(x^{\rightarrow}(t), y^{\rightarrow}(t))$ incurred in t into two parts: (i) migration cost $C_{iMG}(y^{\rightarrow}(t), y^{\rightarrow}(t-1))$ defined earlier, related to decisions in $t-1$; (ii) non-migration cost that relies only on current information at t :

$$C_{t-MG}(x^{\rightarrow}(t), y^{\rightarrow}(t)) = CBW(x^{\rightarrow}(t)) + CDC(y^{\rightarrow}(t)) + CRT(x^{\rightarrow}(t)). \quad (8)$$

We design a lazy migration algorithm (Alg. 1), whose basic idea is to postpone data center switching even if the one-shot optimum indicates so, until the cumulative non-migration cost (in $C_{t-MG}(x^{\rightarrow}(t), y^{\rightarrow}(t))$) has significantly exceeded the potential data migration cost.

Algorithm 1: The Online Lazy Migration (OLM)

Algorithm

- 1: $t = 1$;
- 2: $\ell = 1$; // Time slot when the last change of aggregation data center happens
- 3: Compute data routing decision $x^{\rightarrow}(1)$ and aggregation decision $y^{\rightarrow}(1)$ by minimizing $C(x^{\rightarrow}(1), y^{\rightarrow}(1))$
- 4: Compute $C_{iMG}(y^{\rightarrow}(1), y^{\rightarrow}(0))$ and $C_{t-MG}(x^{\rightarrow}(1), y^{\rightarrow}(1))$;
- 5: while $t \leq T$ do
- 6: if $C_{t-MG}(x^{\rightarrow}(t), y^{\rightarrow}(t)) - C_{t-MG}(x^{\rightarrow}(\ell), y^{\rightarrow}(\ell)) \leq \beta \sum_{v=\ell}^t C_{iMG}(y^{\rightarrow}(v), y^{\rightarrow}(v))$ then
- 7: Derive $x^{\rightarrow}(t)$ and $y^{\rightarrow}(t)$ by minimizing $C_{t-MG}(x^{\rightarrow}(t), y^{\rightarrow}(t))$ and constraint $C_{iMG}(y^{\rightarrow}(t), y^{\rightarrow}(t-1)) \leq \beta (C_{t-MG}(x^{\rightarrow}(t), y^{\rightarrow}(t)) - C_{t-MG}(x^{\rightarrow}(\ell), y^{\rightarrow}(\ell)))$;
- 8: if $y^{\rightarrow}(t) \neq y^{\rightarrow}(t-1)$ then
- 9: Use the new aggregation data center indicated by $y^{\rightarrow}(t)$;
- 10: $\ell = t$;
- 11: if $\ell < t$ then // not to use new aggregation data center
- 12: $y^{\rightarrow}(t) = y^{\rightarrow}(t-1)$, compute data routing decision $x^{\rightarrow}(t)$ by solving (10) if not derived;
- 13: $t = t + 1$

At the beginning ($t=1$), we solve the one-shot optimization and upload data via the derived optimal routes $\vec{x}^*(1)$ to the optimal aggregation data center indicated by $\vec{y}^*(1)$. Let ℓ be the time of the data center switch. In each following time slot t , we compute the overall non-migration cost in $[\ell, t-1]$, $\sum_{t-1} v = t C_v - MG(\vec{x}^*(v), \vec{y}^*(v))$. The algorithm checks whether this cost is at least β_2 times the migration cost $CMG(\vec{y}^*(t), \vec{y}^*(t-1))$. 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 $C-MG t(\vec{x}^*(t), \vec{y}^*(t))$ and an additional constraint, that the potential migration cost, $C-MG t(\vec{y}^*(t), \vec{y}^*(t-1))$, is no larger than β_1 times the nonmigration cost $C-MG t(\vec{x}^*(t), \vec{y}^*(t))$ at time t (to make sure that the migration cost is not too excessive). If a change of migration data center is indicated ($\vec{y}^*(t) = \vec{y}^*(t-1)$), the algorithm accepts the new aggregation decision, and migrates data accordingly. In all other cases, the aggregation data center remains unchanged from $t-1$, while optimal data routing paths are computed given this aggregation decision, for upload of new data generated in t .

Alg.1 avoids aggressive switches of the aggregation data center, to prevent moving a large amount of data back and forth too often. Excessive “laziness” is also avoided. Parameters $\beta_2 > 0$ and $\beta_1 > 0$ control the “laziness” and “aggressiveness” of the algorithm: a large β_2 prolongs the inter-switch interval, while a large β_1 invites more frequent switches.

B. The Randomized Fixed Horizon Control (RFHC) Algorithm

In practical applications, near-term future data generation patterns can often be estimated from history. We assume that the information in the lookahead window can be predicted precisely without error. 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, assume information on data generation for the next l time slots, i.e., $Fd(t), Fd(t+1), \dots, Fd(t+l), \forall d \in D$, are known. We solve the following cost minimization over time frame $[t, t+l]$, given the data aggregation decision of $\vec{y}^*(t-1)$, to derive data routing decisions $\vec{x}^*(v)$ and aggregation decisions $\vec{y}^*(v), \forall v = t, \dots, t+l$, using Alg. 1 minimize $\sum C(\vec{x}^*(v), \vec{y}^*(v)) t + l v = t$. (9)

We design a Randomized Fixed Horizon Control (RFHC) algorithm (Alg. 2). At the beginning, the algorithm 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 $l+1$ FHC algorithms: at $t = 1$, it solves (16) to decide the optimal data routing and aggregation strategies in the period of $t = 1$ to $p-1$ ($p \neq 1$);

then at $t = p, p+l+1, p+2(l+1), \dots$, it solves (16) for optimal strategies in the following $l+1$ time slots, respectively.

Algorithm 2: The Randomized Fixed Horizon Control (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  $\vec{x}^*(1) \dots \vec{x}^*(p-1)$  and  $\vec{y}^*(1) \dots \vec{y}^*(p-1)$  by solving (16) 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  $\vec{x}^*(t), \dots, \vec{x}^*(t+l)$  and  $\vec{y}^*(t), \dots, \vec{y}^*(t+l)$  by solving (16) over the time frame  $[t, t+l]$ ;
9:  $t = t+1$ ;

```

An adversary, with no information on p , finds it hard to contrive specific inputs to degrade the performance of RFHC.

V. SECURITY

In our proposed system, security can be ensured by two methods.

(A). One Time Password (OTP)
Security requirements for cloud computing environment should have trusted computing platform. Our proposed system will demonstrate One Time Password through email to ensure information security. The system will generate this OTP every time the user is trying to login. The One time password will be mailed /sent SMS to them for every login.

1. Implementing trusted computing platform
Dynamic password is generated either in time based or event base mechanisms. One-time password authentication system (OTP) provides authentication for system access (login).

Algorithm 3: OTP generation

Input: User's detailed information including Gmail account & mobile number

Output: One time password

1. Generate random values between 0-123(excluding 0,123)
2. Check the values in range of 47-57(0-9), 65-91(A to Z), 97-122(a to z)
3. If the number is in the range then convert the number in to its ASCII value Else convert the number to its nearest number then convert the number in to its ASCII value
4. Append the generated characters in the password String(OTP)

(B). Privacy Preservation

Privacy preservation is an important issue in the release of data for mining. We focus on a study on the k-anonymity property [10]. The k-anonymity model assumes a *quasi-identifier*, which is a set of attributes that may serve as an identifier in the data set. It is assumed that the dataset is a table and that each tuple corresponds to an individual. Let Q be the quasi-identifier. An equivalence class of a table with respect to Q is a collection of all tuples in the table containing identical values for Q. The size of an equivalence class indicates the strength of identification protection of individuals in the equivalent class. If the number of tuples in an equivalence class is greater, it will be more difficult to re-identify individual. A data set D is k-anonymous with respect to Q if the size of every equivalence class with respect to Q is k or more. As a result, it is less likely that any tuple in the released table can be linked to an individual and thus personal privacy is preserved.

We propose a simple and effective model to protect both identifications and sensitive associations in a disclosed data set. The model extends the k-anonymity model to the (α, k) anonymity model to limit the confidence of the implications from the quasi-identifier to a sensitive value (attribute) to within α in order to protect the sensitive information from being inferred by strong implications.

DEFINITION: (α, k) -ANONYMIZATION. A view of a table is said to be an (α, k) -anonymization of the table if the view modifies the table such that the view satisfies both k-anonymity and α -deassociation properties with respect to the quasi-identifier.

The approaches mentioned in this section solve the privacy issues while moving big data to cloud. (α, k) anonymity model protects sensitive attribute when the attribute contains many values and no single value dominates the attribute. At the same time user authentication is provided by the OTP method.

CONCLUSION

This paper designs efficient algorithms for timely, cost minimizing migration of geo-dispersed big data to the cloud, for processing using a MapReduce-like framework. Two online algorithms are designed to practically migrate data in an online fashion. Data security and privacy is one of the biggest challenges in cloud computing. Cloud data must be protected not only against external attackers but also corrupt insiders. Our proposed system uses OTP and Privacy Preservation approach which aims to make cloud data self intelligent.

REFERENCES

[1]. "Moving Big Data to The Cloud: An Online Cost-Minimizing Approach" Linquan Zhang, Chuan Wu,

Zongpeng Li, Chuanxiong Guo, Minghua Chen, and Francis C.M. Lau

[2]. M. Armbrust, A. Fox, R. Griffith, 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.

[3]. S. Pandey, L. Wu, S. Guru, and R. Buyya, "A Particle Swarm Optimization (PSO)-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environment," in *Proc. IEEE AINA*, 2010

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

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

[6]. X. Cheng and J. Liu, "Load-Balanced Migration of Social Media to Content Clouds," in *Proc. ACM NOSSDAV*, June 2011.

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

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

[9]. Trusted Platform for support services in Cloud Computing environment Sneha Kolhe, Sudhir Dhage*2

[10]. Raymond Chi-Wing Wong, Jiuyong Li, Ada Wai-Chee Fu and Ke Wang, " (α, k) -Anonymity: An enhanced k-anonymity model for privacy preserving data publishing,"