Cloud-of-Clouds Based Resource Provisioning Strategy for Continuous Write Applications

Zeng Zeng *, Bharadwaj Veeravalli †, Samee U. Khan ‡, and Sin G. Teo*
* Institute for Infocomm Research, Singapore, {zengz, teosg}@i2r.a-star.edu.sg
† Department of Electrical & Computer Engineering, National University of Singapore, elebv@nus.edu.sg
‡ Department of Electrical & Computer Engineering, North Dakota State University, samee.khan@ndsu.edu

Abstract—**Nowadays, more and more online services based on cloud computing have taken the places of some traditional applications (e.g., Health Care) that continuously generate large volume of data and require data storage and analysis in time. Such applications can be categorized as "Continuous Writing Applications" (CWA) that have particular requirements on bandwidth, storage, computation, and service reliability. In the meanwhile, they are very sensitive to the cost. In this paper, we present an architecture of multiple cloud service providers (CSPs) or "Cloud-of-Clouds" to provide services to the CWA and propose a novel resource scheduling algorithm to minimize the cost of entire systems. Difference from many research efforts that focus on a single resource, we take many factors into considerations that include user's requirements of bandwidth, storage and computation, the resources of CSPs that can provide, CSPs for data backup, the configurations of Cloud-of-Clouds, system models of CSPs, and many more. We first present the system models of classic CWA applications to capture the resource requirements of users on Cloud-of-Clouds. We then present the problem formulation and our optimal strategy of user scheduling based on *Minimum First Derivative Length* (MFDL) of load paths among the systems. Through theoretical analysis, we prove that our proposed algorithm *Optimal user Scheduling for Cloud-of-Clouds* (OSCC) can achieve the optimal solution.**

Index Terms—**Cloud computing, cloud-of-clouds, continuous writing application, cost model, queueing theory, resource provisioning.**

I. Introduction

We have been witnessing a paradigm shift of Information Technology (IT) towards a pay-per-use or subscription-based service model, known as *Cloud Computing*. This paradigm provides users with many advantages, e.g., provisioning of computing capacities, resource pooling, broad and heterogeneous network access, and rapid elasticity with measured services [4]. With the emergencies of more and more Cloud Service Providers (CSPs), e.g., Windows Azure [3] and Amazon EC2 [2], many existing computer applications running in servers or local PCs have been ported to services provided by the CSPs. Subashini *et al.*, argued that both small and medium-size enterprises use cloud computing services for many reasons, e.g., providing fast access to the computer applications and reducing the operational costs [6]. One of the frequently used services is *Cloud Data Storage*, that allows users to store their data directly on the CSPs, and retrieve the data easily via Internet. Many IT giants provide their users with some free storage spaces to perform some activities, e.g., uploading pictures and videos, and sharing files among friends using the *Cloud Data Storage* (e.g., Apple's iCloud, and Amazon S3). In addition, the CSPs also provide some service packages to the paid users who need more storage spaces and more corresponding function utilities. The paid service subscribers are main income of the CSPs.

The applications of CSPs are known as "Continuous Write Applications" (CWA), which users can perform uploading and accessing data in the CSPs.

In order to provide the best service for the CWA, a major concern of a single CSP is the service availability. Amazon mentions in the licensing agreement that it is possible that the service might be unavailable from time to time [2]. The user's cloud service may be terminated for any reason at any time. Moreover, Amazon is not held liable for any service failure. In this paper, we address "Multi-Clouds" or "Cloud-of-Clouds" to guarantee two things, i) service availability and ii) special requirements of the CWA. In the same time, we also minimize the operational cost of the CSPs that provide the CWA services. Each of the CWA has some service demands that require resources, such as bandwidth, storage space, and CPU cycles, which are to be provided by the CSPs. The CSPs have full control of all of the resources, including the local resources and that provided by the other CSPs, then provided as services of the CWA to the users under the name of a single CSP. We assume that the utility cost of the resources offered by different CSPs may be different and the users, who are geographically distributed, have different charges for the same services. To guarantee the data availability, we make at least two CSPs that store the same data from a single user. In such a way, each of the CSPs can construct its own virtual Cloud-of-Clouds and flexibly arrange the resources to satisfy the requirements of more users and guarantee a higher Quality of Service (QoS).

Now, we shall address an interesting problem emerging in the domain of Cloud-of-Clouds that can provide services to a large number of CWA users around the world: "*how can the CSPs arrange the resources of Cloud-of-Clouds, to achieve the minimum utility cost per customer?*" In this paper, we first formulate a multi-tuple mathematic model to specify the resources and utility costs of the CSPs, inter-cloud communications, and the user's resource requirements. Based on the model, a novel cooperative multi-cloud load balancing algorithm is proposed to achieve the minimum cost per user, while satisfying all of the resource requirements of the users.

We organize this paper as follows. The related work is discussed in Section II. Section III describes the model of the CWA and formulates the optimization problem. In Section IV, we propose our strategy to search for the optimal load path in the Cloud-of-Clouds. In Section V, we detail our proposed algorithm, named *Optimal user Scheduling for Cloud-of-Clouds* (OSCC). Section VI presents benchmarking results. Finally, we conclude our work in Section VII.

II. Related Work

The strategic significance of resource utilization cost and scheduling in networked systems is widely recognized and has long been the subject of multiple workshops and working group meetings [11], [14], [18]. Many resource utilization cost models have been proposed in the literature, such as Divisible Load Theory (DLT) [17], queueing models [13] and learning curve [19]. The utilization costs and provisioning problems, still remain as a challenging task in the cloud computing paradigm in various ways. The CSPs normally provide a menu of service templates that are combined of many resources, such as storage, memory, network bandwidth, and CPU. The resource provisioning problem has become more difficult which users have different requirements on each of the above dimensions [19]. Even Cloud computing provides many advantages (e.g., low cost and easy access to data), dealing with a "single cloud" provider is not a popular option by the users. The trend will in part be dictatorial due to the risks of service unavailability and the possibility of security attacks. A movement towards "multi-clouds", or in other words, "inter-clouds", or "cloud-of-clouds" has emerged recently [5], [10]. Because many commercial CSPs do not provide adequate means to securing the cloud from within the cloud infrastructure, many recently proposed architectures of the CSPs add relevant security and privacy features from the outside. In doing so, the CSPs mainly attempt not to affect the CSPs' interfaces and inner working styles. In this work, we consider collaboration of Cloud-of-Clouds for data backup to improve the system reliability.

III. System Model, Notation and Problem Definition

In this paper, we focus on CSPs that provide Infrastructure-as-a-Service (IaaS), e.g., Amazon EC2, Rackspace, and Microsoft Azure. To satisfy the special requirements of the CWAs that require at least two CSPs to store the data for a higher data availability, one CSP can become a *Chief Cloud* of multiple clouds and provide all of the services for the CWA with its own name. We assume that we have the *Chief Cloud* that can schedule the resources of other CSPs. We use $\mathcal{C}_0$ to denote the set of CSPs. CSP_0 is used to construct the Cloud-of-Clouds consequently, and it can be the *Chief Cloud*. We have $\mathcal{C}_0 = \{CSP_0, CSP_1, \ldots, CSP_n\}$. Each CSP_k $(k = 0, 1, 2, \ldots, n)$ has some resources and we use a tuple $\mathcal{R}_k = [B_k, S_k, C_k]$ to denote the required resources of CSP_k where B_k is the bandwidth, S_k is the storage space, and C_k is the computational capability of CSP_k. Here, we use $\bar{\mathcal{R}}_k = [\bar{B}_k, \bar{S}_k, \bar{C}_k]$ to denote the up-bound resources that CSP_k can provide to CSP_0. We use a vector $\mathcal{P}_k = [P_{B_k}, P_{S_k}, P_{C_k}]$ to denote the utility cost functions of the CSP_k for the resources of bandwidth, storage, and computation, respectively.

We assume that CSP_0 has m users, where $m \gg n$, and the users can be classified into J categories according to the service packages. For example, CSP_0 can provide either one days, one month, or one year of storage service packages with different service charges. Different kinds of users can be assigned with different priorities. The higher priority the category of users has, the more the users pay for the resources provided by the CSP_0 and then, can have more data stored in the system. We use $\mathcal{U} = \{u_i^j\}$ to denote the set of users within the CSP_0, where u_i^j is a user i, who has a priority of j $(j = 1, \ldots, J)$ $(i = 1, \ldots, m)$. Each user u_i^j has the basic resource requirements of bandwidth, storage, and computation that we denote as $r_i^j = [b_i^j, s_i^j, c_i^j]$. All of the requirements of the users must be satisfied by $\mathcal{C}_0$ for a successful resource allocation.

A. Customer's Utility Cost Model of CWA

When a customer i subscribes to a CWA package j with a service provided by CSP_0, the CSP_0 must decide which CSP, denoted as CSP_{i-edge}, in $\mathcal{C}_0$, u_i^j shall be connected directly. Another CSP must receive and store the data sent from the CSP_{i-edge} as backup, which is denoted as CSP_{i-back}. Here, we assume that the CSP_{i-edge} only chooses one CSP from $\mathcal{N}_{i-edge}$ as the CSP_{i-back} ($\mathcal{N}_{i-edge}$ is a set of neighboring nodes of CSP_{i-back}). The utility cost function of customer i is denoted as $fcost(i)$, which includes the bandwidth cost of link from customer i to CSP_{i-edge} is $fcost(i, B_{i-edge})$, and the bandwidth cost of link from CSP_{i-edge} to CSP_{i-back} is $fcost(i - back, B_{i-back})$. The computational cost of customer i's data processed on CSP_{i-edge} is denoted as $fcost(i, C_{i-edge})$, and the storage cost of customer i's data on CSP_{i-edge} and CSP_{i-back} are denoted as $fcost(i, S_{i-edge})$ and $fcost(i, S_{i-back})$, respectively. Here, we use $r_i^j \preceq \bar{\mathcal{R}}_k$ to define $b_i^j \leq \bar{B}_k, s_i^j \leq \bar{S}_k, c_i^j \leq \bar{C}_k$. Then, we can obtain:

$$\begin{aligned} fcost(i) = fcost(i, B_{i-edge}) + fcost(i, S_{i-edge}) + \\ fcost(i, C_{i-edge}) + fcost(i, B_{i-back}) + \\ fcost(i, S_{i-back}), \end{aligned} \tag{1}$$

subject to:

$$r_i^j \preceq \bar{\mathcal{R}}_{i-edge}, b_i^j \leq \bar{B}_{i-back}, s_i^j \leq \bar{S}_{i-back}. \tag{2}$$

If we use r_{i-back} to denote the resource required by customer i on CSP_{i-back}, then we have $r_{i-back} = [b_i, s_i, 0]$, where the computational requirement is zero. Consequently, the cost function of (1) can be converted to:

$$fcost(i) = r_i^j \cdot \mathcal{P}_{i-edge}^T + r_{i-back}^j \cdot \mathcal{P}_{i-back}^T, \tag{3}$$

with subject to $r_i^j \preceq \bar{\mathcal{R}}_{i-edge}$, $r_{i-back}^j \preceq \bar{\mathcal{R}}_{i-back}$, and CSP_{i-edge} and CSP_{i-back} are the neighboring CSPs.

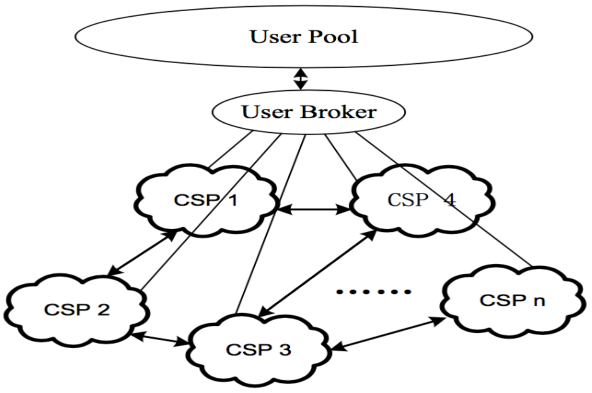


Fig. 1. Illustration of CSPs for Cloud-of-Clouds

B. Problem Formulation

In this section, we use the models to formulate our problem in detail. We first obtain the system cost utility function of all of the users i on $\mathcal{C}_0$ as:

$$\begin{aligned} F(\mathcal{C}_0,\mathcal{U}) &= \sum_{i=1}^{m} fcost(i) \\ &= \sum_{i=1}^{m}(r_i^j \cdot \mathcal{P}_{i-edge}^T + r_{i-back} \cdot \mathcal{P}_{i-back}^T), \quad (4)\end{aligned}$$

subject to:

$$\mathcal{R}_{k|\forall CSP_k \in \mathcal{C}_0} \preceq \bar{\mathcal{R}}_k. \quad (5)$$

From (5), we observe that the utility cost function of CSP, say $\mathcal{P}$, is one of the critical factors that can affect the entire resource scheduling of the Cloud-of-Clouds. Normally, $\mathcal{P}$ can be classified into two kinds of functions, namely linear or non-linear functions [16]. In the *Divisible Load Theory* (DLT) [17], linear functions are used to represent computational and communication utility cost functions. Linear functions are simple and easy for problem formulations. However, if we concerned about the real-time situations, such as system queueing [14], more complicated non-linear functions are formulated to quantify the "costs" [12]. Different cost functions can lead to various resource scheduling in the Cloud-of-Clouds. In this paper, we utilize different cost functions and propose some general algorithms to minimize the total utilization cost of the Cloud-of-Clouds.

IV. Optimal Load Paths within the Cloud-of-Clouds

As we mentioned in the previous section, when a user registers a CWA, the Cloud-of-Clouds must seamlessly provide the user two connected CSPs. One is the *Chief* CSP and another is used for data backup. For simplicity, we use Fig. 1 to illustrate the CSPs of the Cloud-of-Clouds. If CSP_2 is a user's *Chief* CSP, then the user has two potential load paths: (CSP_2, CSP_1) and (CSP_2, CSP_3). The complexity of the potential load paths for each of the user is $O(nk)$, where n is the number of CSPs within the system and k is the average number of neighboring CSPs. Here, we use P_0 to denote the set of all potential paths within the Cloud-of-Clouds. Below, we analyze the load path cost.

A single CSP can be considered as either the *Chief* CSP or a data backup CSP. Referring to (1) and (3), we can observe that there is no requirement on the computational resources for the backup CSP. We assume that a set of users U_l that require services from CSP_l can be further divided into two sub-sets U_{l-edge} and U_{l-back}, where U_{l-edge} is the set of users using CSP_l as the *Chief* CSP and U_{l-back} is the set of users using CSP_l as the data backup CSP. Moreover, we must understand $U_{l-edge} \bigcap U_{l-back} = \emptyset$ and $U_{l-edge} \bigcup U_{l-back} = U_l$. We obtain that the utilization cost function of a single CSP_l:

$$F_l(U_l) = \sum_{\forall i \in U_{l-edge}} (r_i^j \cdot \mathcal{P}_l^T) + \sum_{\forall i \in U_{l-back}} (r_i^j \cdot \mathcal{P}_l^T). \quad (6)$$

If a new user u_i chooses CSP_l as the *Chief* CSP or a data backup CSP, then the new set of users U_l change to $U_l' = U_{l-edge}' \bigcup U_{l-back}$ or $U_l = U_{l-edge} \bigcup U_{l-back}'$, where $U_{l-edge}' = \{U_{l-edge}, u_i\}$ and $U_{l-back}' = \{U_{l-back}, u_i\}$. From (6), we can obtain that the new utilization cost of the CSP_l due to a new user u_i is $F_l(U_l')$.

We use F_{P_i} to denote the cost of u_i along a path P_i, where P_i is the path $(CSP_{i-edge}, CSP_{i-back})$. Consequently, we obtain:

$$\begin{aligned} F_{P_i} &= F_{i-edge}(U_{i-edge}') - F_{i-edge}(U_{i-edge}) \\ &\quad + F_{i-back}(U_{i-back}') - F_{i-back}(U_{i-back}). \quad (7)\end{aligned}$$

It is worthy to mention that $r_i^j \cdot \mathcal{P}_{i-edge}^T + r_{i-back} \cdot \mathcal{P}_{i-back}^T = F_{P_i}$, $\forall i \in \mathcal{U}$. From (7), the objective function described in (5) can be rewritten as:

$$\begin{aligned} F(\mathcal{C}_0,\mathcal{U}) &= \sum_{i=1}^{m}(r_i^j \cdot \mathcal{P}_{i-edge}^T + r_{i-back} \cdot \mathcal{P}_{i-back}^T), \\ &= \sum_{i=1}^{m} F_{P_i}, \quad (8)\end{aligned}$$

where CSP_{i-edge} and CSP_{i-back} are the neighboring CSPs.

Now, we can observe that each user has several load paths with different path costs. To achieve the minimized utilization cost of the entire system, we must determine the optimal cost path for each user one by one. To do so, in the following, we present the definition of the First Derivative Length (FDL) of the load path:

$$\begin{aligned} F_{P_i}' &= (r_i^j \cdot \mathcal{P}_{i-edge}^T + r_{i-back} \cdot \mathcal{P}_{i-back}^T)' \\ &= \frac{\partial(r_i^j \cdot \mathcal{P}_{i-edge}^T)}{\partial r_i^j} + \frac{\partial(r_{i-back} \cdot \mathcal{P}_{i-back}^T)}{\partial r_{i-back}} \\ &= \mathcal{P}_{i-edge}^T + r_i^j \cdot \frac{\partial \mathcal{P}_{i-edge}^T}{\partial r_i^j} + \mathcal{P}_{i-back}^T + \\ &\quad r_{i-back} \cdot \frac{\partial \mathcal{P}_{i-back}^T}{\partial r_{i-back}}. \quad (9)\end{aligned}$$

Combining (1), (9), $r_i^j = [b_i^j, s_i^j, c_i^j], and r_{i-back}^j = [b_i^j, s_i^j, 0]$, we obtain:

$$\begin{aligned} F'_{P_i} &= fcost(i, B_{i-edge}) + fcost(i, S_{i-edge}) \\ &+ fcost(i, C_{i-edge}) + b_i^j \cdot \frac{\partial fcost(i, B_{i-edge})}{\partial b_i^j} \\ &+ s_i^j \cdot \frac{\partial fcost(i, S_{i-edge})}{\partial s_i^j} + c_i^j \cdot \frac{\partial fcost(i, C_{i-edge})}{\partial c_i^j} \\ &+ fcost(i, S_{i-back}) + b_i^j \cdot \frac{\partial fcost(i, B_{i-back})}{\partial b_i^j} \\ &+ fcost(i, B_{i-back}) + s_i^j \cdot \frac{\partial fcost(i, S_{i-back})}{\partial s_i^j}. \quad (10) \end{aligned}$$

A. Optimal Load Path for Each User

For each user u_i, among all the P_0 paths, there must be at least one path, denoted as $\bar{P}_i$, whose FDL is minimum. This path is defined as the minimum first derivative length (MFDL). Next, we use the theorem of our previous work [21] as follows.

Theorem 1. *The set of $\bar{P}_i$, where $\forall u_i \in \mathcal{U}$, is an optimal solution to (5)* ***if and only if*** *every user i selects the path with the MFDL from the set of all possible paths, P_0. Furthermore, if $fcost$ is convex, then $\bar{P}_i$ is also optimal if and only if the path with the MFDL in P_0 that provides a service to u_i.*

For space limitation, we put the proof in our technical report [20]. In this work, we assume that users have been categorized into several classes with different priorities. We schedule the users according to the amount of services required with the following scheduling sequences:

Sequence A (SA): Randomly select a user from all of the users who are not scheduled yet;

Sequence B (SB): Sort the users according to the priorities from low to high, and randomly select the un-selected users within the same priority one by one, and;

Sequence C (SC): Sort the users according to the priorities and then sort the users with the same priority according to the amount of required services. Then, select the users one by one from high to low.

V. The Proposed Algorithm

We propose an algorithm to achieve a near-optimal solution of (8) that is referred as the "Optimal user Scheduling for Cloud-of-Clouds" (OSCC). Our proposed algorithm has three phases. The first phase is the initialization, in which we construct the set of load paths P_0. As we discussed before, each of the CSP within the system can construct the load path with any neighboring CSP. If the Cloud-of-Clouds is fully-connected and the number of clouds within the system is n, then each of the CSP has $n - 1$ load paths and within the system there are a total of $n \cdot (n-1)$ load paths. Once P_0 has been constructed, we obtain the FDL of each path according to (10) and then, determine the minimum one among P_0. It is worthy to mention that if there are two or more load paths with the same MFDL, we randomly choose one as the MFDL among them. We sort the users in $\mathcal{U}$ according to the sequences SA, SB, or SC. For space limitation, we put the details of our algorithm in the technical report [20].

VI. Performance Evaluation and Discussions

In this section, we select a discrete-event approach that can model, simulate, and also evaluate the system [15]. We implemented the simulation in C++ to evaluate the proposed and existing algorithms. The existing round-robin algorithms [13] are selected as the benchmark algorithms to compare the performance with $OSCC$. In the round-robin algorithms, we construct a list of all paths by selecting one path for each user one by one, until no more user can be added into the system.

Our experiment settings are as follows. We set 10 CSPs. Each CSP can randomly choose at least one and at most nine other CSPs as its neighbors. The resources, $\mathcal{R}_k$, of CSP_k are randomly generated within the following ranges, i.e., the bandwidths of the CSPs are between $[10,\ 1000]$ gigabit per second (GBPS), the storage spaces of CSPs are between $[2, 200]$ PB, the number of virtual machines in CSPs are between $[1000, 100000]$, and the value of μ_k^c is between $[500, 5000]$ MIPS. The performances of all the virtual machines are same. α_k (for bandwidth), β_k (for storage), and γ_k (for computation) are randomly selected from $[1, 100]$ with the unit of cost per 24 hours. The upper bounds of the resources for each CSP are set to 0.95 of the total CSP resources.

In the experiment, we assume that there are four kinds of users: (1) u_i^1: Surveillance users; (2) u_i^2: Health Care users; (3) u_i^3: Smart home users; and (4) u_i^4: Environmental monitor users. We assume that the resource requirements of these four categories of users are defined as $r_i^1 = [200, 100, 100]$, $r_i^2 = [100, 50, 1000]$, $r_i^3 = [50, 20, 30]$, and $r_i^4 = [10, 10, 10]$ with the units of [KBPS, GB, MIPS], respectively[1]. In each experiment, we increase the number of users until no user can be scheduled any more. For each user, we randomly determine the category from one to four following the uniform distribution.

A. Experiments with Small Size of Cloud-of-Clouds

We first randomly generate the parameters of all CSPs. Subsequently, A set of users is generated with random classes. We then schedule the user using the $OSCC$ and round-robin (RR) algorithms, respectively. At first, we just use the SA sequence and later, we will discuss the effects of the sequences by carrying out some further experiments.

The upper bound utilizations of bandwidth, storage, and computation capability were set to 0.95. We stop a simulation when there are no load paths available for new users. For every set of users within the experiments, we recorded the average utility cost per user with the number of users increasing until the procedures stop, and the results are shown in Fig. 2(b). To

[1]It may be noted that the parameter values chosen are for the purpose of demonstration of the strategies and are not restricted to the actual parameter values in a practical setting.

TABLE I
THE UTILIZATIONS OF RESOURCES IN CSPS

Types	Utilizations of CSP 1 to CSP 10									
OSCC (ρ_k)	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
RR (ρ_k)	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
OSCC (band.)	0.001	0.039	0.041	0.013	0.075	0.007	0.012	0.448	0.004	0.011
RR (band.)	0.012	0.053	0.036	0.029	0.017	0.016	0.029	0.930	0.009	0.020
OSCC (storage)	0.003	0.029	0.066	0.021	0.932	0.031	0.016	0.024	0.018	0.013
RR (storage)	0.046	0.040	0.060	0.047	0.210	0.076	0.035	0.051	0.043	0.024

our surprise, we find that both $OSCC$ and RR stop scheduling when the number of users reaches $754,054$, which means that only $754,054$ users can be supported with the current system settings. Table I shows the resource utilizations of all the CSPs when no users can be added to the system.

Again, the bottleneck of the entire system is the computation capability as depicted in Table I. We find that in either $OSCC$ or RR $\rho_k = c_k/(n_k \cdot \mu_k)|_{\forall CSP_k}$ has reached 0.95, that is the upper bound of the system. The utilizations of other resources, such as storage and bandwidth, are very low and few of the utilizations have been exceeded 10%. Therefore, in such system settings, when the number of users reaches the maximum number which the system can support, computation costs dominate the total costs. Referring to Fig. 2(b), except the last point, the $OSCC$ performs much better than RR.

In Fig. 2(b), when the number of users is small, due to the facts that the utilizations of bandwidth and computation are very low and the costs of storage are fixed, the user costs are dominated by storage costs. The $OSCC$ can search the best load paths for each users to guarantee the minimum costs. The RR schedules the users one by one with the consideration of whether the resources can support the users or not. However, without the awareness of the utilizations of the resources along the paths, the RR assigns more and more users to some paths that can be overloaded soon and while, the utilizations of many other paths are still very low. As the number of users increases, the costs soon are dominated by the lowest resources (in such the setting, the lowest one is the CSP that has the lowest computation capability) in the system and then, rises rapidly. Soon after the path with lowest resources become saturated, the load paths with the second, third lowest resources become saturated one by one. This explains why when the number of users reaches some small point, the costs of the RR increases steadily. Because we consider the cost per user, when more users join in the system, the cost may fluctuate as indicated in the Fig. 2(b), and we can anticipate that the users along the paths with lowest resources will have much more costs than the average. On the contrary, our proposed $OSCC$ algorithm considers the users equally and the cost of each user can be nearly the same in the system. Only after the number of users reaches around $400,000$ that the cost starts rising slightly until the entire system becomes saturated.

B. Experiments on Medium to Large Size of Cloud-of-Clouds

To evaluate the scalability of the $OSCC$ algorithm, we compare the $OSCC$ with RR on Cloud-of-Clouds, the size of which is large to medium. We randomly generate a Cloud-of-Clouds with 50 CSPs. The topology of the system is randomly generated and each CSP may have one to nine neighboring CSPs. Similar to Section VI-A, we set the computation resources to be the bottleneck of the system at first and then, balance the recourses to evaluate the performance of the $OSCC$ and RR algorithms.

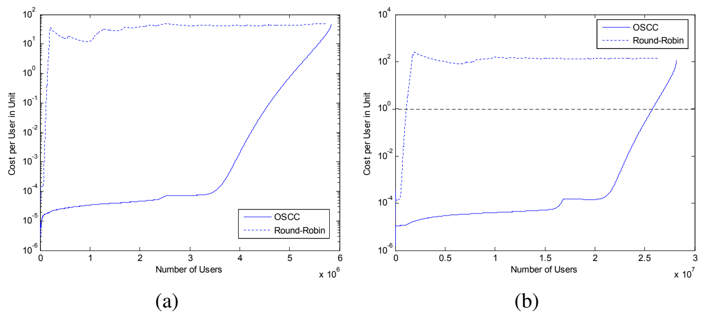


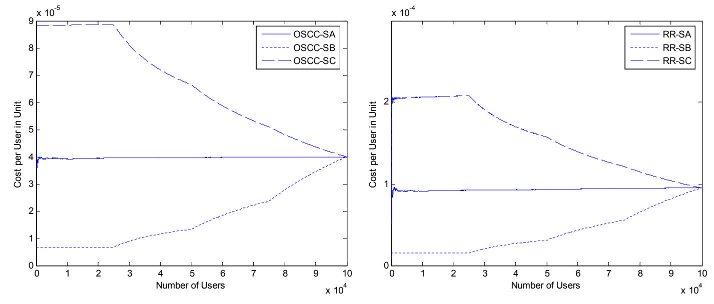
Fig. 2. Performance comparisons of OSCC algorithm and round-robin with SA, (a) Computation capacity is low; (b) System settings have been balanced.

Fig. 2(a) shows the result of large to medium size systems to support more users compared to the small size of Cloud-of-Clouds, when the computation capacity is the bottleneck. The cost per user of the RR increases rapidly when the number of users is very small and nearly hit the peak around 0.2×10^6, which is 0.6% of 5.8^6 that is the total number of the users. On the other hand, the cost per user of the $OSCC$ is kept within a very low level in most cases, and hits one when the number of users exceeds 5×10^6; whereas, the cost per users of the RR again is around 50.

We balance the computation capacities of CSPs as similar in Section VI-A Fig. 2(b) shows that $28,196,493$ users can be supported by the $OSCC$ and only $26,388,862$ users can be supported by the RR. The cost of users can exceed one for both the $OSCC$ and RR algorithms as depicted in Fig. 2(b). From the experiments, we show that our proposed $OSCC$ is efficient, flexible and extensible. It can be used especially for large scale systems.

C. Effects of Scheduling Sequences on the OSCC and RR Algorithms

As mentioned earlier, we evaluate the four categories of users where each has different resource requirements or priorities. Given a set of users, the algorithms schedule them



(a) (b)

Fig. 3. Performance comparisons of algorithms with sequences SA, SB, and SC. (a) OSCC; (b) RR.

one by one by following some pre-defined sequences, e.g., SA, SB, and SC, as presented in the previous section. To evaluate the performance of scheduling sequences on the final results, we run several experiments based on small size Cloud-of-Clouds for the $OSCC$ and RR algorithms. We experiment with $100,000$ users where each type of the users is set to $25,000$.

In Fig. 3(a), $OSCC - SA$, $OSCC$ scheduling the users by following SA sequence, performs steady and the cost per user changes slightly that is around 4×10^{-5} units. For $OSCC - SB$, which schedules the users with the lowest priorities first, the cost per user increases with more and more users scheduled. On the other hand, for $OSCC - SC$, which schedules the users with the highest priorities first, the cost per user decreases. Until all of the users have been scheduled, the three curves converge to a single point. In Fig. 3(b), the curves of $RR - SA$, $RR - SB$, and $RR - SC$ converge to a point around 1×10^{-4} when all users have been scheduled.

From the experiments, the scheduling sequences have no or very small effect on the results. In the Cloud-of-Clouds, we consider millions of users, it is unlikely to re-schedule all of the users that have been settled for some newly coming users. Therefore, our proposed $OSCC$ algorithm also may be used in dynamic scenarios.

VII. Conclusions

We propose an optimal user scheduling algorithm, *Optimal user Scheduling for Cloud-of-Clouds* (OSCC) for CWA applications. Our algorithm considers many factors that include user's requirements of the bandwidth, the storage, and the computation, the resources of CSPs that can provide, CSPs for data backup, and the configurations of Cloud-of-Clouds, the cost models of CSPs. In our solution, we construct a list of the potential load paths that selects the optimal one with the MFDL for each of the users within the system.

We compare the performance of the $OSCC$ and RR algorithms. Our experiment have showed that the $OSCC$ is scalable, extensible, and also easy for implementation. The $OSCC$ outperforms the RR in all fields under considerations, including the cost per user, the maximum number of users the system can support, and the resource utilizations of CSPs. Our solution can also be applied to other utility cost models, such as the learning curve, in order to satisfy the various requirements of different Cloud-of-Clouds. We can easily extend the $OSCC$ to dynamic situations without a pre-defined set of users.

Acknowledgement: The work is supported by 1st Singapore-China NRF-NSFC Grant (No. NRF2016NRF-NSFC001-111). Samee U. Khan's work is supported by (while serving at) the National Science Foundation.

References

[1] P. Mell and T. Grance, "Draft NIST Working Definition of Cloud Computing," Referenced on Jun. 3rd, 2009, Online at http://csrc.nist.gov/groups/SNS/cloud-computing/index.html, 2009.

[2] Guohui Wang, "The Impact of Virtualization on Network Performance of Amazon EC2 Data Center," IEEE INFOCOM Proceedings, Mar. 14-19, 2010.

[3] T. Redkar and T.Guidici, *Windows Azure Platform,* APRESS, 2011.

[4] S. U. Khan, A. Y. Zomaya, and L. Wang, "Scalable Computing and Communications: Theory and Practice," *Wiley-IEEE Computer Society Press,* New Jersey, USA, 2013, 880 p., 303 illus., ISBN 978-1-1181-6265-1.

[5] I. Houidi, M. Mechtri, W. Louati, and D. Zeghlache, "Cloud Service Delivery Across Multiple Cloud Platforms," *IEEE International Conference on Services Computing,* pp. 740-741, 2011.

[6] S. Subashini and V. Kavitha, "Survey on Security Issues in Service Delivery Models of Cloud Computing," *Journal of Network and Computer Applications,* vol. 34, no. 1, pp. 1-11, 2011.

[7] L. Atzori, A. Lera, and G. Morabito, "The Internet of Things: A Survey," *Computer Networks*, vol. 54, no. 15, pp. 2787-2850, Oct. 2010.

[8] http://www.camba.tv.

[9] M. Dia, G. Juan, O. Lucas, and A. Ryuga, "Big Data on the Internet of Things," *6th Internet Conference on Innovative Mobile and Internet Serivces in Ubiquitous Computing*, pp. 898-900, 2012.

[10] M. A. Zlzain, E. Pardede, B. Soh, and J. A. Thom, "Cloud Computing Security: From Single to Multi-Clouds", *Hawaii International Conference on System Sciences,* pp. 5490 - 5499, 2012.

[11] F. Franciosi and W. Knottenbelt, "Data Allocation Strategies for the Management of Quality of Service in Virtualised Storage Systems," *Mass Storage Systems and Technologies (MSST), 2011 IEEE 27th Symposium on,* pp. 1-6, 23-27 May 2011.

[12] Jie Li, and Hisao Kameda, "Load Balancing Problems for Multiclass Jobs in Distributed/Parallel Computer Systems," *IEEE Trans. Computers* vol. 47, no. 3, pp. 322-332, March 1998.

[13] D. Bertsekas and R. Gallager, *Data Networks*, Prentice-Hall, Inc., 1992.

[14] Z. Zeng and V. Bharadwaj, "Design and Performance Evaluation of Queue-and-Rate-Adjustment Dynamic Load Balancing Policies for Distributed Networks," *IEEE Trans. Computers,* vol. 55, no. 11, pp. 1410-1422, Nov. 2006.

[15] J. J. Kinney, *Probability: An Introduction with Statistical Applications,* New York: John Wiley & Sons, 1997.

[16] M. Avriel, *Nonlinear Programming Analysis and Methods*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1997.

[17] V. Bharadwaj, D. Ghose, V. Mani, and T. G. Robertazzi, *Scheduling Divisible Loads in Parallel and Distributed Systems,* IEEE Computer Society Press, Los Almitos, California, 1996.

[18] J. Cao, K. Hwang, K. Li, and A. Y. Zomaya, "Optimal Multiserver Configuration for Profit Maximization in Cloud Computing," *IEEE Trans. Parallel and Distributed Systems,* vol. 24, no. 6, pp. 1087-1096, Jun. 2013.

[19] A. Gera and C. H. Xia, "Learning Curves and Stochastic Models for Pricing and Provisioning Cloud Computing Services," *Service Science,* vol. 3, no. 1, pp. 99-109, 2011.

[20] Z. Zeng, V. Bharadwaj, S.C. Khan, and S.G. Teo "Technical Report for Cloud-of-Clouds Based Resource ProvisioningStrategy for Continuous Write Applications", https://www.dropbox.com/s/v1jlvmr0zypau8t/COMM2017.pdf?dl=0

[21] Z. Zeng and V. Bharadwaj "Optimal metadata replications and request balancing strategy on cloud data centers," *IEEE Trans. Parallel and Distributed Systems,* vol. 74, no. 10, pp. 2934-2940, 2014.