Journal of Parallel and Distributed Computing 152 (2021) 98–107

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

A novel cooperative resource provisioning strategy for Multi-Cloud load balancing

Bo Zhang [a], Zeng Zeng [b,*], Xiupeng Shi [b], Jianxi Yang [c], Bharadwaj Veeravalli [d], Keqin Li [e]

[a] School of Electronics and Control Engineering, Chang'an University, China
[b] Institute for Infocomm Research, A*STAR, Singapore
[c] AI Research Center, Chongqing Jiaotong University, China
[d] ECE Department, National University of Singapore, Singapore
[e] Computer Science Department, State University of New York at New Paltz, United States of America

ARTICLE INFO

Article history:
Received 12 June 2019
Received in revised form 5 November 2020
Accepted 1 February 2021
Available online 8 March 2021

MSC:
00-01
99-00

Keywords:
Cloud computing
Cost model
Continuous writing application
Queueing theory
Resource provisioning

ABSTRACT

The paradigm of cloud computing has heralded a new avenue of computing, offering benefits of increased data accessibility with low cost. Continuous Writing Applications (CWA) (e.g., augmented online services for Health Care) have specific requirements on data storage, computation and bandwidth, thus are cost-sensitive with limited budgets and time. Herein, we propose an architecture of multi-cloud service provider (CSP) or "Multi-Cloud" to provide services to CWA, and design a novel resource scheduling algorithm to minimize the system cost. The system models of classic CWAs to tackle the resource requirements of users on MCP are exploited. The study can help to understand the characteristics of different resources and conclude Multi-Cloud being the most attractive to many CWA implementations. Interconnections of multiple CSPs and their load paths (i.e., data passing through possible interconnections) are introduced. We then formulate the problem and present optimal user scheduling based on Minimum First Derivative Length (MFDL) of system load paths. Theoretical analysis demonstrated that the solutions with minimized costs can be achieved by the proposed algorithm, termed "Optimal user Scheduling" for Multi-Cloud (OSMC). Through rigorous simulations regarding different influencing factors, the proposed strategy has proven to be scalable, flexible, and efficient in many practical scenarios.

© 2021 Elsevier Inc. All rights reserved.

1. Introduction

Subscription based or pay-per-use service business model known as Cloud Computing has gained significant importance over the past years. Cloud computing has numerous advantages, including provisioning of computing capacities, wide and heterogeneous network access, resource pooling and rapid elasticity with measured services [17,22,24]. Most applications running on local PCs have been migrated to Cloud Service Providers (CSPs), such as Amazon EC2 [28], Windows Azure [26], IBMs Blue Cloud [1], due to many advantages, e.g., providing flexible costs and improving data and application availability. In [27], it was suggested that cloud computing services can allow fast access to applications as well as reduce infrastructure costs, even for small and medium companies.

The primary service offered by CSPs is Cloud Data Storage. Instead of storing data on local machines, users can store data on CSPs that allow easy data retrieval from any geographical region via the Internet. Various IT giants have provided limited free services and storage spaces for data uploading such as pictures and videos. Besides, the ability to share the data with other users is allowed through Cloud Data Storage, such as Amazon S3, Microsoft SkyDrive, and Apple's iCloud. CSPs also provide more service options for users who are willing to pay for extra spaces and more corresponding function utilities. Nowadays, with the emergence of Wireless Sensor Networks (WSNs), mobile networks, Internet of Thing (IoT) [5], more and more data and information "ensed" from the environment can be stored and processed on CSPs, and then shared by the authorized entities. Fig. 1 illustrates four main categories of services provided by CSPs, that are detailed in the following.

(*a*) Surveillance System: Due to convenient storage and retrieval functions, more and more deployed surveillance systems have adopted CSPs to store the round the clock video recordings, instead of the traditional way using local storage devices. Additional services provided by CSPs include video surveillance search, customized pre-alarms, as well as flexible and convenient methods for viewing, analyzing and extracting information from

* Corresponding author.
E-mail address: zengz@i2r.a-star.edu.sg (Z. Zeng).

https://doi.org/10.1016/j.jpdc.2021.02.003
0743-7315/© 2021 Elsevier Inc. All rights reserved.

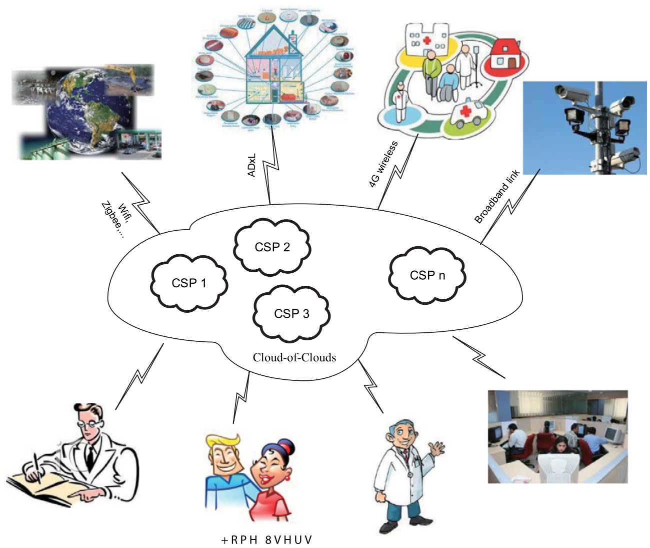


Fig. 1. Four classic services CSPs may offer.

real-time videos or recorded footage. Some commercial companies, such as Camba [2], have entered this market with the hope to attract more subscribers with very low prices.

(*b*) Health Care or e-Health: A human being generates 1 TB of information in a day [12]. This data is already in use in fields such as marketing or social networks. However, this data has not been exploited to its full potential the area of e-health. CSPs are ideal platforms to keep and analyze this data, for implementing online e-learning, data mining by machine learning, or other state-of-the-art technologies.

(*c*) Smart Homes: A smart home is a concept of the pervasive computing. It will progressively gain significance for the people living in high technology areas. Large amounts of data and complex control required for implementing smart homes exceed the ability of local computers. Cloud computing can help reduce local workload, through which home users can obtain the real time information from Internet directly. Smart home based on the CSPs is more efficient, convenient, flexible and cost effective [15].

(*d*) Environment Monitor: With the advancement of WSNs, more and more environmental data are "sensed", ranging from water pollution, air pollution to bridge health, high-speed railway health, in which CSPs play an important role to save and process such massive data. TBs of data are generated every day. Cloud computing provides a platform to store and compute high volumes of data. This in turn leads to generation of invaluable information for disaster predictions. Cloud enables an efficient way to access and utilize the data for users around the world, especially government officers, researchers, and scientists, among others.

"Continuous Write Applications" (CWAs) are CSP applications used to compute large volumes of data. A main concern by single CSP is to provide continuous service availability. For example, the license agreement of Amazon states that the provided cloud services may be unavailable occasionally [28]. This may lead to that the user's cloud services may terminate at any time. In addition, Amazon is not responsible for cloud service availability. To tackle the above limitation, in this paper, "Multi-Cloud" is proposed to assure service availability that can satisfy particular requirements of CWAs. We also reduce the operating cost of the CSPs that deliver the CWAs services. There are some service demands for each of the CWAs that involve resources requirements in terms of storage capacity, bandwidth, and CPU cycles. The CSPs completely controls all resources, including both local resources and that provided by the other CSPs, then provided as services of the CWA to the users under the term of a solo CSP. It is assumed that the resource utility costs may vary for the same service for users based on geographical distribution. To assurance data availability, we assume that same data from a single user will be stored by at least two CSPs. In this way, each CSP can build its particular virtual Multi-Cloud or "multiple Clouds" and organize the resources flexibly to meet the needs of increasing users, as well as ensure a higher Quality of Service (QoS).

A worth-exploring problem in the field of Multi-Cloud can offer services to massive CWA users around the world: "how can the CSPs organize the resources of Multi-Cloud, to reach the lowest utility cost for each customer?" Herein, a multi-tuple mathematical model is formulated to describe the key concerns, including users' resource requirements, utility costs of the CSPs, as well as inter-cloud communications. Additionally, based on the model, we propose a novel algorithm of cooperative multi-cloud load-balancing, which can meet all the resource requirements of users while achieving the minimum cost per user.

The rest of the paper is organized as follows. The relevant research work is discussed in Section 2. Section 3 describes the CWA model and formulates the optimization problem. In Section 4, our strategy for optimal load-path search in Multi-Cloud is proposed. In Section 5, we introduce our proposed algorithm in detail, termed Optimal user Scheduling for Multi-Cloud (OSMC). Sections 6 and 7 cover the experimental results and conclusions.

2. Related work

Many online applications can be classified as CWAs, such as surveillance systems, e-Health, and Internet-of-Things [5], etc. These applications continuously generate massive data that require to be transported, stored and analyzed in real time. Each application requires three basic factors, namely communication bandwidth for data transportation, large data storage capacity, and computational power to analyze the collected data. Many traditional CWAs are restricted by small scale systems. In order to set up a system that can satisfy all the requirements independently, expensive devices, such as cameras, sensors, wires, central control devices, and pre-compiled programs need to be planted to provide improved storage space and computational capacities [15]. However, the scalability of such systems is extremely poor.

With the development of cloud computing [21] and big data processing [10,11], the basic requirements of CWAs can be satisfied by CSPs. Many novel business models targeting at CWA markets have emerged recently, such as Camba for surveillance systems [12]. With the support of Cloud computing, CWAs can evolve from closed, independent systems to open, scalable platforms. The success of CWAs on cloud computing mainly depends on two factors namely operational cost and reliability. It is widely recognized that the resource utilization cost and scheduling in networked systems entails strategic significance, and many seminars and working group meetings have explored the subject for a long period [9,13,29].

Recently, more and more research work explore the specific solutions provided by the major Cloud/Edge provider and it is difficult to fully exploit different Clouds/Edges concurrently. EasyCloud is proposed in [4], which is an easy and effective toolkit and user interface able to not only interact with multiple and different Cloud/Edge platforms at the same time but also to provide a rule-based engine where the user can specify what to do in real-time when the workload of the services running on the Clouds/Edges becomes under-utilized (e.g., switch-off the service to save money) or over-utilized (e.g., switch-on new computational resources to overcome the increased workload).

Many research efforts have been conducted on resource utilization cost models in the literature, such as learning curve [14], Divisible Load Theory (DLT) [8], and queueing models [7]. Cost of use and supply issues are still an arduous task in cloud computing. CSPs provides a service template menu which consists of various resources, such as CPU, storage, bandwidth, and memory. Difficulty in resource provisioning increases since various requirements may emerge from various users on each of these dimensions [14]. Although Cloud computing brings in many benefits such as low cost and data accessibility, however, due to concerns such as the risks of service unavailability and potential malicious attacks, dealing with a "single cloud" providers is expected to be no longer preferred by users. Therefore, the focus has recently shifted to "multi-clouds", or in other words, "inter-clouds", or "cloud-of-cloud" [3,16]. Herein, we study the collaboration of Multi-Cloud for data backup with aims to enhance system reliability.

3. System model, notation and problem definition

This study focuses on CSPs that provide Infrastructure-as-a-Service (IaaS). IaaS is a service model providing an infrastructure of the computer for supporting enterprise operations. Typical IaaS services include Microsoft Azure, Amazon EC2, Rackspace, and GoGrid. One CSP can be assigned as the *Chief Cloud* of the multiple Clouds. Furthermore, every CWA requires at least two CSPs for data storage to improve data availability. It is assumed that the Chief Cloud schedules the resources of other CSPs. Let $\mathcal{C}_0$ denote the set of CSPs. CSP_0 is introduced to build the Multi-Cloud accordingly, which can be the *Chief Cloud*. We have $\mathcal{C}_0 = \{CSP_0, CSP_1, \dots, CSP_n\}$. Each CSP_k, $k = 0, 1, 2, \dots, n$, has certain resources. A tuple is used to denote the resources $\mathcal{R}_k = [B_k, S_k, C_k]$, where B_k denotes the bandwidth, S_k denotes the storage space, and C_k denotes the computational capability of CSP_k. $\bar{\mathcal{R}}_k = [\bar{B}_k, \bar{S}_k, \bar{C}_k]$ denotes the upper-bound of the resources that a CSP_k can offer to CSP_0. A vector $\mathcal{P}_k = [P_{B_k}, P_{S_k}, P_{C_k}]$ is used to represent the utility cost functions of the CSP_k for the resources in terms of bandwidth, storage, computing, respectively.

It is assumed that the CSP_0 involves m users, where $m \gg n$. According to the service packages, the users can be divided into J categories. For example, CSP_0 can provide storage service packages for one day, one month, or one year, and charge different service fees. All users are assigned some priorities on the basis of category classifications. Higher priority of user infers that the user has paid more for the services, thus in return gaining access to more data storage in the system. Let $\mathcal{U} = \{u_i^j\}$ denote the set of users within the CSP_0, where u_i^j denotes the user i with a priority of $j = 1, \dots, J$ for user $i = 1, \dots, m$. Each user u_i^j has some basic resource requirements in terms of bandwidth, storage, and computation that can be denoted as $r_i^j = [b_i^j, s_i^j, c_i^j]$. All the user requirements must be satisfied by $\mathcal{C}_0$ in order to successfully allocate resources.

As shown in Fig. 2, the main focus is on the communications of the inter-cloud and intra-cloud. We assume that if two CSPs are connected by a direct communication link, then CSPs are the neighboring nodes of each other. We use $\mathcal{N}_k$ to denote the set of neighboring nodes of CSP_k, e.g., in Fig. 2, $\mathcal{N}_3 = \{CSP_2, CSP_4, CSP_n\}$. We use $L_{k,l}$ to denote the communication bandwidth between CSP_k and CSP_l, and $B_{k,l}$ to denote the communication cost on a link $L_{k,l}$. If two CSPs have no direct link, then the CSPs need two or more network links to set up the communication paths.

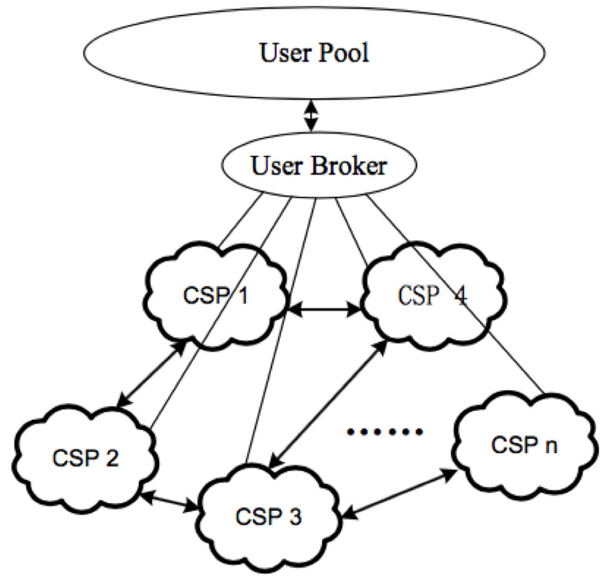


Fig. 2. Illustration of a user broker for Multi-Cloud.

3.1. Customer's utility cost model of CWA

Two CSPs are required for each customer i, CSP_{i-edge} and CSP_{i-back}. When the customer i subscribes a CWA package j, then CSP_0 decides which CSP_{i-edge} in C_0 should be connected directly to u_i^j. Simultaneously, another CSP is required to store the data sent from CSP_{i-edge} as a backup, which is denoted as CSP_{i-back}. It is assumed that the CSP_{i-edge} only select one CSP from $\mathcal{N}_{i-edge}$ as the CSP_{i-back}. The following equation is obtained for utility cost function of customer i,

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

subject to:

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

A list of notations with their descriptions is presented in Table 1.

Supposing r_{i-back} is used to denote the resource required by customer i on CSP_{i-back}, then we get $r_{i-back} = [b_i, s_i, 0]$, where the computational requirement is zero. Consequently, the cost function of (1) can be transferred 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 $r_i^j \preceq \bar{\mathcal{R}}_{i-edge}$, $r_{i-back}^j \preceq \bar{\mathcal{R}}_{i-back}$, and CSP_{i-edge} and CSP_{i-back} are the neighboring CSPs.

3.2. Problem formulation

In this section, the problem will be formulated in detail. We first obtain the system cost utility function of all of the users on $\mathcal{C}_0$ as:

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

subject to:

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

Table 1
List of notations.

Notation	Description
$fcost(i, B_{i-edge})$	Bandwidth cost of link from customer i to CSP_{i-edge}
$fcost(i, S_{i-edge})$	Storage cost of customer i's data on CSP_{i-edge}
$fcost(i, S_{i-back})$	Storage cost of customer i's CSP_{i-back}
$fcost(i, B_{i-back})$	Usage cost of bandwidth of CSP_{i-back}
$fcost(i, C_{i-edge})$	Computational cost of customer i's data processed on CSP_{i-edge}
P_i^T	Transpose of vector $[P_{B_k}, P_{S_k}, P_{C_k}]$ to denote the utility cost functions of the CSP_k

From (5), it is observed that the utility cost function of CSP, denoted as $\mathcal{P}$, can be considered as one of the critical factors that impact the resource scheduling for the Multi-Cloud. $\mathcal{P}$ can be divided into two types of functions, i.e., linear or non-linear functions [6]. According to *Divisible Load Theory* (DLT) [8], linear functions represent computation and communication utility cost functions. However, linear functions are simplified problem formulations. For real-time scenarios (e.g., system queueing [29], etc.), linear functions are required to quantify the utility costs [30]. Resource scheduling algorithms vary with different cost functions. The paper proposes a general method to minimize the total utilization cost of Multi-Cloud using various cost functions.

With reference to Fig. 2, the problem considered in this paper can be elaborated as follows. The Multi-Cloud that provides the CWA services, has a user pool with m users and each of the user has different resource requirements. Let CSP_0 be the *user broker* also known as the *Chief Cloud*. A *user broker* is the broker which collects the resource status from all the clouds, keeping a log of the resource cost of each of the cloud. The *user broker* has two functions:

- One is to schedule the users from the user pool.
- One is to make decision on the CSP_{i-edge} and CSP_{i-back} for each of the user.

For every user provided with CSP_{i-edge} and CSP_{i-back}, they are neighboring CSPs that are connected by direct communication links. The aforementioned scheduling must satisfy the two main constraints: (a) The resource requirements of all the users are to be satisfied by the Multi-Cloud; and (b) the sum of all of the users' resources that a CSP provides must not exceed the actual available resources.

4. Optimal load paths within Multi-Cloud

With reference to Section 3, when a CWA is registered, the Multi-Cloud must seamlessly deliver two connected CSPs to the user, which are a *Chief* CSP and a data backup CSP. With reference to 2, if CSP_2 is a user's *Chief* CSP, then two potential load paths are available, namely (CSP_2, CSP_1) and (CSP_2, CSP_3). Thus, the complexity of the potential load paths for each of the user is given by $O(nk)$, where n is the number of CSPs in the system and k is the average number of adjacent CSPs. Let P_0 denote the set of all potential paths within the Multi-Cloud. After the definitions, we shall analyze the cost of all the load paths in the following.

4.1. Cost of single CSP and load paths

Each CSP can be considered as either *Chief* CSP or data backup CSP. With reference to Eqs. (1) and (3), it can be noted that computational resources are not required for the backup CSP. It is assumed that a set of users, U_l, that require services from CSP_l can be further classified into two sub-sets U_{l-edge} and U_{l-back}, where U_{l-edge} denotes the set of users using CSP_l as the *Chief* CSP and U_{l-back} denotes the set of users using CSP_l as the data backup CSP. Further it can be noted that $U_{l-edge} \bigcap U_{l-back} = \emptyset$ and $U_{l-edge} \bigcup U_{l-back} = U_l$ hold. Referring to (15), the utilization cost function of a single CSP_l can be obtained as follows:

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

Supposing a new user, u_i, select CSP_l as the *Chief* CSP or data backup CSP, therefore the new set of users U_l is updated to $U_l' = U_{l-edge}' \bigcup U_{l-back}$ or $U_l = U_{l-edge} \bigcup U_{l-back}'$, respectively, where $U_{l-edge}' = \{U_{l-edge}, u_i\}$ and $U_{l-back}' = \{U_{l-back}, u_i\}$. According to (6), the new utilization cost of the CSP_l because of the new user, u_i, can be obtained, namely $F_l(U_l')$.

Suppose the load path of u_i is determined to be (CSP_2, CSP_1) by the user broker. CSP_2 and CSP_1 function as the *Chief* and a data backup CSP, respectively. We use F_{P_i} to represent the cost of u_i along a path P_i, where P_i is path $(CSP_{i-edge}, CSP_{i-back})$. Consequently, we obtain:

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

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

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

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

Thus, it can be found that each user has multiple load paths with various costs. In order to minimize the utilization cost of the entire system, the optimal cost path for all the users shall be determined one after another. For this reason, we introduce the definition of the First Derivative Length (FDL) of the load path as follows:

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

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:

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

$$+ fcost(i, S_{i-back}) + b_i^j \cdot \frac{\partial fcost(i, B_{i-back})}{\partial b_i^j}$$
$$+ fcost(i, B_{i-back}) + s_i^j \cdot \frac{\partial fcost(i, S_{i-back})}{\partial s_i^j}. \quad (10)$$

4.2. Optimal load path for each user

For every user u_i, among all the P_i paths, there must be at least one path, denoted as $\bar{P}_i$, with minimum FDL. This path is defined as the minimum first derivative length (MFDL). The following theorem is from [29].

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*** *each user i chooses the path with the MFDL from the set of all potential paths, P_0. Moreover, if fcost is assumed to be convex, then $\bar{P}_i$ is optimal if and only if the path with the MFDL in P_0 provides a service to u_i.*

From Theorem 1, it is observed that an optimal solution results only if users' requests travel along the MFDL paths among P_0. The optimal load path for the users can be determined one by one within the system. Theoretically, the sequence of the user scheduling has no bearing on the final results. In the DLT, the load can be arbitrarily divided and the entire system can be eventually balanced according to the optimal solutions. However, in this paper, the user service load cannot be divided arbitrarily to be served by parallel load paths. To further explain the aforementioned, a situation is considered where a few users require large amounts of data. If such users are scheduled at the end, a system imbalance will be observed. Thus, user scheduling is a necessity. Herein, it is assumed that users have been divided into several categories with different priorities the users can be scheduled according to required service volume. The scheduling sequences are considered as follows:

- Sequence A (SA): Select a user amongst the ones not yet scheduled randomly;
- Sequence B (SB): Sort and assign the users from low to high according to the priorities, and select the un-selected users one by one in the same priority randomly, and;
- Sequence C (SC): Requires two sorting sequences. First, the users are sorted according to their priorities. Second, sort the users with the same priorities on the basis of amount of required services. The user is then selected from the highest priority.

In the following section, an algorithm is proposed that can schedule the users to reach the minimum utilization cost of the entire system.

5. The proposed algorithm

Each user will be assigned to a load path by the user broker (CSP_0) as shown in Fig. 2). The basic idea is to determine the MFDL path for all the users within the system one by one according to a predefined sequence, such as SA, SB, and SC, to minimize the utilization cost of the entire system. In the sub-sections, an algorithm is proposed to reach a near optimal solution of (8). This is referred to as the "Optimal user Scheduling for Multi-Cloud" (OSMC).

5.1. Design of OSMC

The pseudo code of the proposed algorithm can be described as follows (refer to Table 2). In the initialization phase, a set of load paths is constructed, P_0. Each CSP within the system can build a load path with any neighboring CSP. Let the number of clouds in the system be n. Assuming the Multi-Cloud is fully-connected, each CSP has n-1 load paths. Therefore, there will be a total of $n \cdot (n - 1)$ load paths within the system. Table 3 presents the pseudo code of the Sub-algorithm of Load Path Finding (*SaLPF*). It constructs a set of load paths, P_0, within the system. After the construction of P_0, the FDL of each path according to (10) obtained. The minimum among P_0 is then determined. If two or more load paths have the same MFDL, then one is chosen randomly as the MFDL. The users should be sorted in $\mathcal{U}$ following the sequences of SA, SB, or SC (See Table 2).

Table 2
Algorithm 1: *OSMC*.

Step 1: ***Initialization***
  Construct the set of load paths, P_0, using *SaLPF*.
  For $k = 1$ to m, where m = $|\mathcal{C}_0|$.
    Set $B_k = 0, S_k = 0, C_k = 0$.
  End For.
  Calculate the FDL of each path in P_0 by using (10).
  Construct $\mathcal{U}$, sorting the users according to SA, SB, or SC
Step 2: ***Main Loop***
  For each priority j.
  For $i = 1$ to n, where $n = |\mathcal{U}|$.
    Sort the path in P_0 according to the FDL from lowest to highest.
    For $l = 1$ to $|P_0|$.
      Set CSP_{i-edge} and CSP_{i-back} along path P_l;
      If $r_i^j + \mathcal{R}_{i-edge} \preceq \bar{\mathcal{R}}_{i-edge}$ and $r_{i-back}^j + \mathcal{R}_{i-back} \preceq \bar{\mathcal{R}}_{i-back}$;
        Assign u_i to path P_l;
        Set $\mathcal{R}_{i-edge} = \mathcal{R}_{i-edge} + r_i^j$ and $\mathcal{R}_{i-back} = \mathcal{R}_{i-back} + r_{i-back}^j$;
        Calculate the FDL of all paths in P_0 that involve CSP_{i-edge} or CSP_{i-back} by using (10).
        Break Loop of l;
      Else
        Continue;
    End For
    If $l > P_0$
      Set a warning "Out of Recourses, User u_i cannot be scheduled!";
    End If
  End For
Step 3: ***End Algorithm***
  Output a list of users that cannot be scheduled.

Table 3
Algorithm 2: Sub-algorithm of Load Path Finding (*SaLPF*).

  Set $P_0 = \emptyset$
  For $k = 1$ to n
    For $l = 1$ to $|\mathcal{N}_i|$, where $\mathcal{N}_i$ is the set of neighboring CSPs of CSP_i.
      Construct the load path (CSP_i, CSP_l).
      Include (CSP_i, CSP_l) into P_0.
    End For
  End For
  Return P_0.

The target of the algorithm is to determine the optimal path for each user and guarantee that the resources of the load path satisfy the user's requirements. The main loop (refer to Table 2) focuses on three procedures as follows.

1. Sort the load paths in P_0 according to the FDL from the lowest to the highest;
2. Examine whether the load path can satisfy the requirements of the user. If the resources of the user cannot be satisfied along the load path, examine the load path with the next minimum FDL within the available load paths until the set of available load paths becomes empty; and
3. If the optimal load path for the user has been determined, update the FDL of all the load paths effected by the selected load path.

There may be some users whose resource requirements cannot be satisfied due to the limited resources of the Cloud-of-Clouds. In such a case, the algorithm will send a warning for system extension, instead of rescheduling all of the users to meet the requirements.

5.2. Case studies of load paths

In this section, the analysis of a single load path is illustrated as an example. The cost functions of bandwidth, storage, and computation for each CSP are initially constructed. Referring to (3), it is assumed that for CSP_k, $\mathcal{P}_k = [P_{B_k}(b_k), P_{S_k}(s_k), P_{C_k}(c_k)]$, where

- $P_{B_k}(b_k)$ denotes the bandwidth cost function, where b_k is the bandwidth used;
- $P_{S_k}(s_k)$ denotes storage cost function, where s_k is the total utilized storage space;
- $P_{C_k}(c_k)$ denotes computation cost function, where c_k is the computation capacity used in CSP_k.

Following the assumptions that the communication cost can be modeled as $M/M/1$ system [7,19,29] and the communication cost per unit time for CSP_k is α_k. Therefore, $P_{B_k}(b_k)$ can be formulated as

$$P_{B_k}(b_k) = \frac{\alpha_k}{\mu_k^b - b_k}, \tag{11}$$

where μ_k^b is the communication processing rate of CSP_k in *KBPS* (kilobit per second).

Further assumptions are made that the storage cost is independent on s_k, and we can obtain

$$P_{S_k}(s_k) = \beta_k, \tag{12}$$

where β_k is a constant in the unit cost per gigabytes.

As per the discussion in [9], although an $M/G/n$ queueing system are taken into consideration in cloud computing, the $M/M/n$ system is the only model that accommodates an analytical and closed-form expression of the probability density function to calculate the waiting time of a new service request. Therefore, the computational cost can be modeled as $M/M/n$ system. The computational cost per unit time for CSP_k is assumed to be γ_k. Following [9,29], we can obtain $P_{C_k}(c_k)$ as

$$P_{C_k}(c_k) = \gamma_k \cdot Tc_k = \frac{\gamma_k P_Q}{n_k\mu_k^c - c_k}, \tag{13}$$

$$where,\ P_Q = P\{Queueing\} = \frac{p_{a0}(n_k\rho_k)^{n_k}}{n_k!(1-\rho_k)}, \tag{14}$$

$$and\ \rho_k = \frac{c_k}{n_k\mu_k^c}, (\rho_k < 1).$$

In (14), n_k is the number of virtual machines that can provide the computation services to the users in CSP_k, μ_k^c is the computation processing rate of the virtual machines in MIPS (million instructions per second), and p_{a0} is the probability of no request waiting in the queue of the node, which is given as:

$$p_{a0} = (\sum_{j=0}^{n_k-1}\frac{(n_k\rho_k)^j}{j!} + \frac{(n_k\rho_k)^{n_k}}{n_k!(1-\rho_k)})^{-1}. \tag{15}$$

The unit of (13) is unit cost per MIPS.

We have two kinds of users within the system and we assume that the resource requirements of the users with priority one is denoted as $r_i^1 = [b_i^1, s_i^1, c_i^1]$. For example, if $r_i^1 = [200, 100, 20]$, then that means the users of priority one require 200 kbps bandwidth, 100 G storage space, and 20 MIPS computation capacities. Therefore, from (8), (12), and (13), $\mathcal{P}_k = [F_B(b_k), F_S(s_k), F_C(c_k)]$ can be constructed for each CSP within the system and we can obtain the closed-form of the load path by using (9) for users with priority equal to one.

Following assumptions are also made:

- Users can be classified into multiple categories with different resource requirements;
- The users of the same category have the same resource requirements, that means $r_x^j = r_y^j = r^j, x \neq y$ hold.

In (9), we can observe that r_i^j may vary with different j, and if we consider only a single user, then the calculation will increase rapidly. Therefore, in *OSMC* we calculate the FDL of each of the load path for users with the same priority j by using r^j. We also can further simplify the calculations by using $\bar{r} = [\bar{b}, \bar{s}, \bar{c}]$, instead of r_i^j in (9), where

$$\bar{r} = \frac{1}{m}[\sum_{i=1}^{m} b_i^j, \sum_{i=1}^{m} s_i^j, \sum_{i=1}^{m} c_i^j], \tag{16}$$

for all load paths in the system.

From the above discussion, it was observed *fcost* in (5) is the sum of communication cost (convex function), storage cost (linear function), and computational cost (convex function). Hence, *fcost* in this model is also a convex function and the condition of Theorem 1 can be guaranteed.

5.3. Rate of convergence

Our algorithm is based on Newton's method [25]. We denote $h(r^k) = D'(r^k)$ for iteration k. If an optimal solution is at point $\bar{r}$, then we have $h(\bar{r}) = 0$. Alternatively, we have:

$$\phi(r^k) = r^k - \alpha^k \frac{h(r^k)}{h'(r^k)}$$

By the mean value theorem, we obtain:

$$r^{k+1} - \bar{r} = \phi(r^k) - \phi(\bar{r}) = \phi'(\xi^k)(r^k - \bar{r}),$$

where ξ^k lies between r^k and $\bar{r}$. Then

$$|r^{k+1} - \bar{r}| = \frac{|h''(\xi^k)h(\xi^k)|}{[h'(\xi^k)]^2}|r^k - \bar{r}|$$

and

$$|h(\xi^k)| = |h(\xi^k) - h(\bar{r})| = |h'(\eta^k)||\xi^k - \bar{r}| \leq |h'(\eta^k)||r^k - \bar{r}|$$

where η^r lies between ξ^k and $\bar{r}$. Hence:

$$|r^{k+1} - \bar{r}| \leq \frac{|h''(\xi^k)h|}{[h'(\xi^k)]^2}|r^k - \bar{r}|^2$$

Let

$$\beta = sup\frac{|h''(\xi^k)h'(\eta^k)|}{[h'(\xi^k)]^2},$$

then

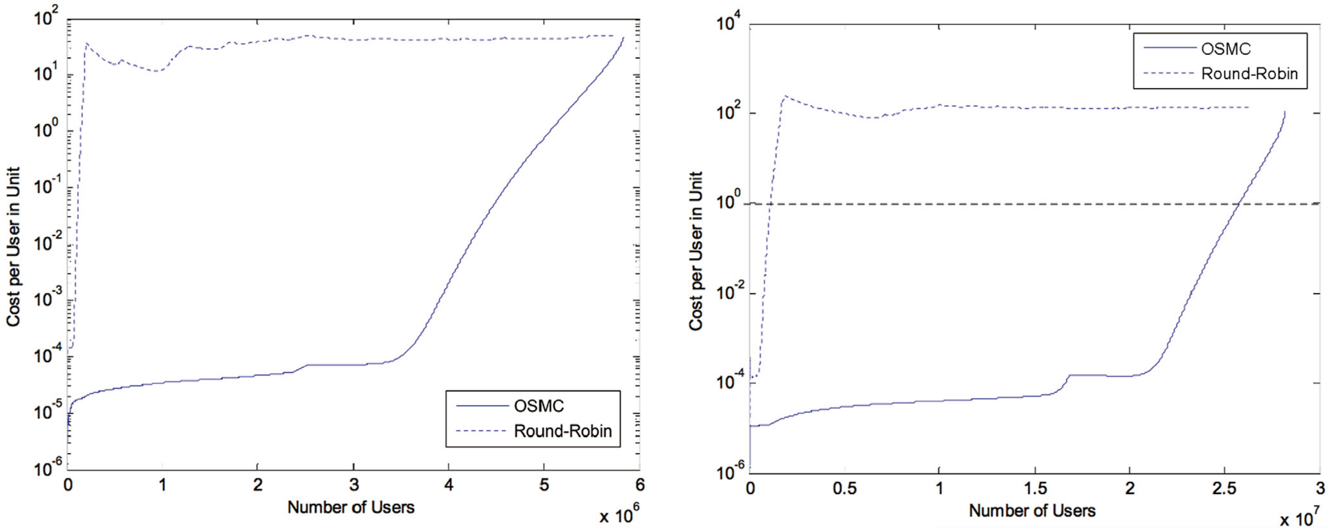
$$|r^{k+1} - \bar{r}| \leq \beta|r^k - \bar{r}|^2$$

Hence the convergence of our algorithm is 2, which exhibits a **super-linear** convergence.

6. Performance evaluation and discussions

6.1. Assumptions for simulations

In our simulations, we employ a discrete-event approach to model, simulate, and evaluate the system [18]. 10 CSPs are considered for serial simulations. For the simulations different settings of Multi-Cloud are taken into consideration. A number of other CSPs ranging from one to nine are randomly selected as neighboring CSPs for each CSP_k to construct the set of $\mathcal{N}_k$ within the system. The resources, $\mathcal{R}_k$, of CSP_k are generated within the following ranges randomly. The bandwidth of the CSPs is set to



(a) Low computation capacity. (b) Balanced system settings.

Fig. 3. Performance comparisons of OSMC and round-robin algorithms with SA in large–medium size systems.

Table 4
The parameters of CSPs in small scale system.

Parameters	CSP 1 to CSP 10									
Band. (GBPS)	960	240	610	490	900	770	460	200	830	450
Storage(PB)	124	160	186	148	360	820	188	184	840	180
Number of VM	2000	4000	10000	2000	2000	4000	2000	8000	4000	2000
Comp. (MIPS)	1000	8000	5000	10000	5000	5000	9000	6000	3000	7000
α_k	84	69	38	84	51	57	71	43	31	19
β_k	20	69	31	55	16	70	38	87	86	60
γ_k	50	90	83	65	82	67	35	29	35	54

the value randomly selected from [10, 1000], with the unit in gigabit per second (GBPS). The storage space of CSPs is set to the value randomly generated from [2, 200] PB. The number of virtual machines in CSPs is set to the value randomly selected from [1000, 100000], and the value of μ_k^c is set to the random value from [500, 5000] MIPS. Further, it is assumed that the performance of all the virtual machines within the same CSP remains the same. In the previous section, the costs of each of the unit resource in the CSPs have been defined as α_k for bandwidth, β_k for storage, and γ_k for computation, which are set to the value randomly chosen from [1, 100] with the unit of cost per 24 hours. For each CSP, the upper bounds of the resources are set to be 0.95 of the total CSP resources.

In the simulations, four types of users are assumed: (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 assumed that the resource requirements of these four categories of users were 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 of the simulation experiment, the number of users is increased till no more users could be scheduled. For each user, categories were determined randomly between one to four using uniform distribution.

To compare the performance with *OSMC*, the simulations use round-robin algorithm [7] as the benchmark algorithm. Round-robin algorithm is selected as it is simple and is efficient for job scheduling. In the round-robin algorithm, a list of all paths is selected one after another for each user, until no more users can be added to the system.

6.2. Experiments on small size of Multi-Cloud

In this set of experiments, we randomly generate a Multi-Cloud with 10 CSPs. The parameters assumed for all CSPs are as shown in Table 4. We generate a set of users with random classes and schedule the users by using the *OSMC* and round-robin (RR) algorithm, respectively. Initially, only the SA sequence is used. In the later sections the effect of the sequences will be discussed by carrying out further experiments.

For simulations purposes, the number of users are generated from one to infinite. The simulation is stopped until no load paths are available for the users. The upper bound utilizations of bandwidth, storage, and computation capability were set to 0.95. For every set of users within the experiments, the average utility costs per user are recorded with the increasing of user numbers, until the end of the procedures, and the results are illustrated in Fig. 3(b). It was observed that when the number of users raises up to 754,054, both *OSMC* and *RR* terminate the scheduling. It can be inferred that the current system settings can support only 754,054 users. The resource utilizations of all the CSPs are elaborated when the system cannot support adding more users, as reported in Table 5.

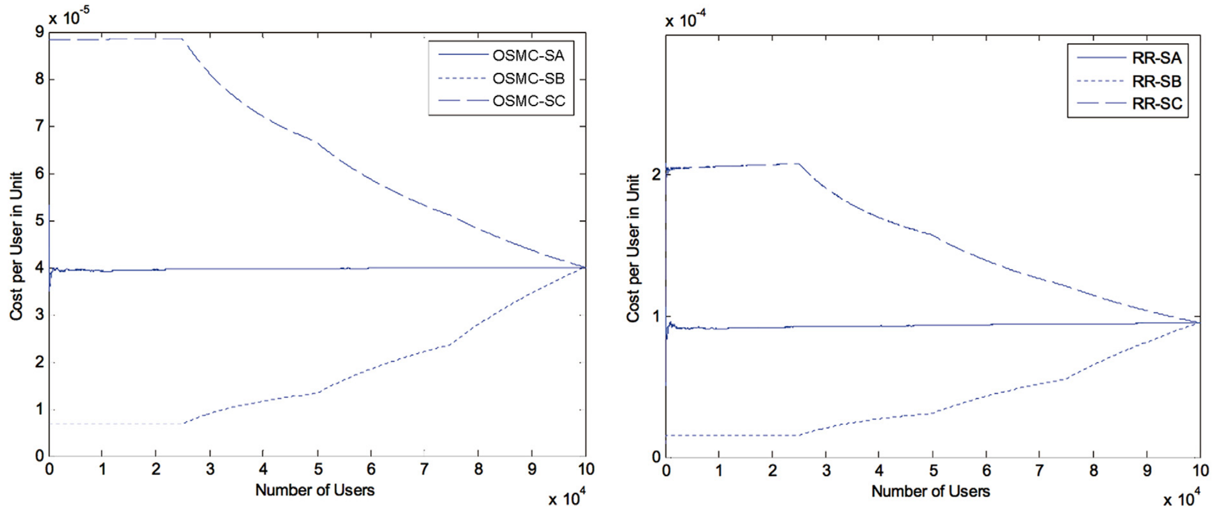
In Table 5, computation capability is observed to be the bottleneck of the entire system. We can find that in either *OSMC* or *RR* $\rho_k = c_k/(n_k \cdot \mu_k)|_{\forall CSP_k}$ has reached the upper limit of the system, namely 0.95. The utilizations of other resources (e.g., storage and bandwidth) are at very low levels, rarely exceeding 10%.

In Fig. 3(b), since the utilization of bandwidth and computation are very low when the number of users is at a very small

[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 5
The utilizations of resources in CSPs.

Types	Utilizations of CSP 1 to CSP 10									
OSMC (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
OSMC (band.)	0.001	0.004	0.007	0.011	0.012	0.013	0.039	0.041	0.448	0.075
RR (band.)	0.012	0.009	0.016	0.02	0.029	0.029	0.053	0.036	0.93	0.017
OSMC (storage)	0.003	0.018	0.031	0.013	0.016	0.021	0.029	0.066	0.024	0.932
RR (storage)	0.046	0.043	0.076	0.024	0.035	0.047	0.04	0.06	0.051	0.21



(a) OSMC.

(b) RR.

Fig. 4. Performance comparisons of algorithms with sequences SA, SB, and SC.

scale, the user cost will account for the storage costs. Thus, for each user, the *OSMC* searches for the best load paths, and ensure the lowest costs. The *RR* considers whether resources can support users, and schedules users one by ones. However, the resources utilizations along the paths are not considered. The *RR* will allocate more and more users to certain paths which will lead to overloading while, many other paths remain very low utilizations. With the user number increasing, the lowest resources in the system soon dominate the costs, which then rises rapidly. Herein, the CSP with the lowest computation power is the lowest resource in this setting. Hereafter the path with the lowest resources turn out to be saturated, the load paths with the second, third lowest resources become saturated successively. This clarifies that the costs of the *RR* grows steadily when the number of users reaches certain small point. Since the cost per user is considered, the cost may fluctuate when more users join in the system, as indicated in the figure. It is anticipated that the users with lowest resources along the paths will possess much more costs compared with the average. By contrast, users are considered equally in the proposed *OSMC* algorithm which promotes nearly the same costs for each user in the system. Merely after the number of users increases around 400,000 that the cost begins to rise slightly until the entire system becomes saturated. It may be noted that when some thresholds of resources are set, and single type of resources are the bottleneck in the system, both the *RR* and *OSMC* algorithms can take full usage of the entire system.

6.3. Experiments on medium to large size of Multi-Cloud

To observe the scalability of the *OSMC* algorithm, in this set of experiments, the *OSMC* is compared with *RR* on Multi-Cloud, the size of which is large to medium. 50 Multi-Cloud are randomly generated. The topology of the system is generated randomly and for each CSP, a range of one to nine neighboring CSPs may exist. Similar to Section 6.1, the computation of the system is at first set to be the system bottleneck and then, balance the recourses to examine the performance of the *OSMC* and *RR* algorithms.

Referring to Fig. 3(a), which are the results of large to medium size systems when the bottleneck is from computation capacity, it is observed that compared to the small size of Multi-Cloud, the system can support much more users. For small number of users, the cost per user of the *RR* shows a rapid increase and almost reaches the peak around 0.2×10^6, which is 0.6% of 5.8×10^6 (i.e., the number of total users). On the other side, in most cases, the cost per user of the *OSMC* can be maintained within a very low scale, and reaches one when the number of users surpasses 5×10^6; whereas for the *RR*, the cost per users again retains around 50.

The computation capacities of CSPs are balanced out with the same methods used in Section 6.1 and carry out the experiments again. The results are presented in Fig. 3(b), where the *OSMC* can support 28, 196, 493 users and the *RR* can merely support 26, 388, 862 users. It also indicates when the cost of users exceeds one for both the *OSMC* and *RR* algorithms.

From these experiments, it can be concluded that our proposed *OSMC* is efficient, flexible and extensible, and can be applied for large scale systems.

6.4. Effects of scheduling sequences on OSMC and RR algorithms

As mentioned earlier, in our simulations, four categories of users are generated that have distinct resource requirements or priorities. Given a set of users, they can be scheduled one by one by the algorithms, and follow certain pre-defined sequences, e.g., *SA*, *SB*, and *SC*, as presented in the previous section. Furthermore, we carry out several experiments based on small size

Multi-Cloud for the *OSMC* and *RR* algorithms, to examine the effects of scheduling sequences on the final results. We assume that we have 100,000 users and the number of each type users is set to 25,000. We report the final results of the *OSMC* and *RR* algorithms in Fig. 4.

In Fig. 4(a) *OSMC-SA*, *OSMC* schedules the users following the *SA* sequence, which performs steadily and the cost per user shows a slight fluctuation, which is around 4×10^{-5} units. For *OSMC-SB*, the users with the lowest priorities are scheduled firstly, as more and more users are scheduled, the cost per user increases. On the other side, for *OSMC-SC*, in which the users with the highest priorities are scheduled firstly, the cost per user decreases. The three curves converge to a single point, until all users have been scheduled. In Fig. 4(b), we also observe that when all of the users have been scheduled, the curves of *RR-SA*, *RR-SB*, and *RR-SC* converge to a point, which is around 1×10^{-4}.

From these experiments, we can find that there is no or very small effect of scheduling sequences on the final results. Because in the Multi-Cloud, millions of users are considered, it is impossible to re-schedule all of the users that have been resolved for some newcomers. It is very interesting to point out that because the user sequences have no effect on the final results, our proposed *OSMC* algorithm also has potential usage in dynamic scenarios.

7. Conclusions and future work

This study proposed an optimal user scheduling algorithm for CWA applications, named the *Optimal user Scheduling for Multi-Cloud* (OSMC). Various factors were considered, including the user requirements of bandwidth, storage, and computation, the resources of CSPs that can provide, CSPs for data backup, and the configurations of Multi-Cloud, cost models of CSPs. By using the $M/M/1$ queueing model for communication cost, $M/M/n$ queueing model for computation cost and a fix value model for storage cost, we formulate the problem of optimal user scheduling to minimize the cost per user in the Multi-Cloud environment. In our proposal solution, we develop a list of potential load paths and choose the optimal one with the MFDL for each of user within the system.

The performance of *OSMC* algorithm was compared with *RR* in our experiments. Simulations were carried out on Multi-Cloud with 10 CSPs, Further, we were able to prove that the *OSMC* is scalable, extensible, and convenient for implementation. In all areas considered, the *OSMC* outperforms the *RR*, including cost per user, maximum number of users supportable by the system, resource utilizations of CSPs. Our methodology can be applied to other utility cost models, such as learning curve, to satisfy various requirements of different Multi-Cloud. We also find that the order of the user scheduling has no effect on the final cost per user and the *OSMC* can be easily extended to dynamic situations without pre-defined set of users.

In cloud/edge computing systems, privacy-preserving [20], security [23], etc., are very important. In our future work, we shall address these challenges and consider the corresponding computational cost in our framework.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: The co-authors are from I2R Singapore, NUS Singapore, Chongqing Jiaotong University China, and State University of New York USA. The reviewers from the four Universities and Institutes shall have the conflict of interest and cannot review the draft. The domains shall be I2R.a-star.edu.sg, nus.edu.sg, cqjtu.edu.cn, and newpaltz.edu.

Acknowledgment

The research was supported by Singapore-China NRF-NSFC Grant with No. NRF2016NRF-NSFC001-111.

References

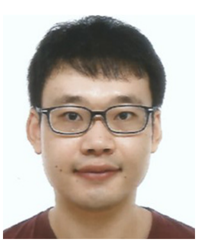
[1] http://www-03.ibm.com/press/us/en/pressrelease/22613.wss.
[2] http://www.camba.tv.
[3] M.A. AlZain, E. Pardede, B. Soh, J.A. Thom, Cloud computing security: from single to multi-clouds, in: 2012 45th Hawaii International Conference on System Sciences, IEEE, 2012, pp. 5490–5499.
[4] C. Anglano, M. Canonico, M. Guazzone, Easycloud: a rule based toolkit for multi-platform cloud/edge service management, in: 2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC), 2020, pp. 188–195, http://dx.doi.org/10.1109/FMEC49853.2020.9144821.
[5] L. Atzori, A. Iera, G. Morabito, The internet of things: A survey, Comput. Netw. 54 (15) (2010) 2787–2805.
[6] M. Avriel, Nonlinear Programming: Analysis and Methods, Courier Corporation, 2003.
[7] D.P. Bertsekas, R.G. Gallager, P. Humblet, Data Networks, 2, Prentice-Hall International New Jersey, 1992.
[8] V. Bharadwaj, D. Ghose, V. Mani, T.G. Robertazzi, Scheduling Divisible Loads in Parallel and Distributed Systems, 8, John Wiley & Sons, 1996.
[9] J. Cao, K. Hwang, K. Li, A.Y. Zomaya, Optimal multiserver configuration for profit maximization in cloud computing, IEEE Trans. Parallel Distrib. Syst. 24 (6) (2012) 1087–1096.
[10] C. Chen, K. Li, A. Ouyang, K. Li, Flinkcl: An opencl-based in-memory computing architecture on heterogeneous cpu-gpu clusters for big data, IEEE Trans. Comput. 67 (12) (2018) 1765–1779.
[11] C. Chen, K. Li, A. Ouyang, Z. Zeng, K. Li, Gflink: An in-memory computing architecture on heterogeneous CPU-GPU clusters for big data, IEEE Trans. Parallel Distrib. Syst. 29 (6) (2018) 1275–1288.
[12] M. Diaz, G. Juan, O. Lucas, A. Ryuga, Big data on the internet of things: An example for the e-health, in: 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IEEE, 2012, pp. 898–900.
[13] F. Franciosi, W. Knottenbelt, Data allocation strategies for the management of quality of service in virtualised storage systems, in: 2011 IEEE 27th Symposium on Mass Storage Systems and Technologies (MSST), IEEE, 2011, pp. 1–6.
[14] A. Gera, C.H. Xia, Learning curves and stochastic models for pricing and provisioning cloud computing services, Serv. Sci. 3 (1) (2011) 99–109.
[15] H. Gu, Y. Diao, W. Liu, X. Zhang, The design of smart home platform based on cloud computing, in: Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology, 8, IEEE, 2011, pp. 3919–3922.
[16] I. Houidi, M. Mechtri, W. Louati, D. Zeghlache, Cloud service delivery across multiple cloud platforms, in: 2011 IEEE International Conference on Services Computing, IEEE, 2011, pp. 741–742.
[17] S.U. Khan, L. Wang, A.Y. Zomaya, Scalable computing and communications: Theory and practice, Concurr. Comput.: Pract. Exper. 20 (13) (2008) 1573–1590.
[18] J.J. Kinney, Probability: An Introduction with Statistical Applications, John Wiley & Sons, 2014.
[19] J. Li, H. Kameda, Load balancing problems for multiclass jobs in distributed/parallel computer systems, IEEE Trans. Comput. 47 (3) (1998) 322–332.
[20] X. Liu, R.H. Deng, K.R. Choo, J. Weng, An efficient privacy-preserving outsourced calculation toolkit with multiple keys, IEEE Trans. Inf. Forensics Secur. 11 (11) (2016) 2401–2414, http://dx.doi.org/10.1109/TIFS.2016.2573770.
[21] C. Liu, K. Li, K. Li, Minimal cost server configuration for meeting time-varying resource demands in cloud centers, IEEE Trans. Parallel Distrib. Syst. 29 (11) (2018) 2503–2513.
[22] C. Liu, K. Li, C. Xu, K. Li, Strategy configurations of multiple users competition for cloud service reservation, IEEE Trans. Parallel Distrib. Syst. 27 (2) (2015) 508–520.
[23] N. Lwamo, L. Zhu, C. Xu, K. Sharif, X. Liu, C. Zhang, Suaa: A secure user authentication scheme with anonymity for the single and multi-server environments, Inform. Sci. 477 (2018) http://dx.doi.org/10.1016/j.ins.2018.10.037.

[24] P. Mell, T. Grance, Draft NIST working definition of cloud computing, 15, (32) 2009, p. 2, Referenced on June. 3rd.
[25] A. Mordecai, Nonlinear programming: Analysis and methods, 1976.
[26] T. Redkar, T. Guidici, T. Meister, Windows Azure Platform, Springer, 2009.
[27] S. Subashini, V. Kavitha, A survey on security issues in service delivery models of cloud computing, J. Netw. Comput. Appl. 34 (1) (2011) 1–11.
[28] G. Wang, T.E. Ng, The impact of virtualization on network performance of amazon ec2 data center, in: 2010 Proceedings IEEE INFOCOM, IEEE, 2010, pp. 1–9.
[29] Z. Zeng, B. Veeravalli, Design and performance evaluation of queue-and-rate-adjustment dynamic load balancing policies for distributed networks, IEEE Trans. Comput. 55 (11) (2006) 1410–1422.
[30] Z. Zeng, B. Veeravalli, Do more replicas of object data improve the performance of cloud data centers? in: Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing, IEEE Computer Society, 2012, pp. 39–46.

Bo Zhang received his bachelor's degree in Automatic Control from Chang'an University, China, in 2002 and master's degree in Public Administration from Dalian Maritime University, China, in 2010. He is currently working toward the Ph.D. degree at the School of Electronics and Control Engineering, Chang'an University, Xi'an, Shaanxi, China.

His research interests include Cloud Computing, transportation cyber–physical systems and their applications.

Zeng Zeng received the Ph.D. degree in electrical and computer engineering from the National University of Singapore, Singapore, in 2005, and the BS and MS degrees from the Huazhong University of Science and Technology, Wuhan, China, in 1997 and 2000, respectively. Currently, he works as Senior Scientist, the Head of Deep Learning for Semiconductor Program, I2R, A*STAR, Singapore. From 2011 to 2014, he worked as a senior research fellow with the National University of Singapore. From 2005 to 2011, he worked as an associate professor in Computer and Communication School, Hunan University, China. His research interests include distributed/parallel computing systems, data stream analysis, deep learning, multimedia storage systems, wireless sensor networks, and onboard fault diagnosis, etc.



Xiupeng Shi received the B.Eng. and M.Eng. degrees from Beijing Jiaotong University, China, in 2010 and 2012, respectively, and the Ph.D. degree from Nanyang Technological University (NTU), Singapore, in 2019.

He is currently a Scientist with Agency for Science, Technology and Research (A*STAR). From 2012 to 2015, he worked as a Business General Manager with China Railway Materials. His primary research interests include risk prediction, automated machine learning, smart mobility, and supply chain finance.

Email: shix@i2r.a-star.edu.sg.

Jianxi Yang received the Ph.D. degree in Bridge Engineering from Chongqing Jiaotong University, China, in 2011, and the BS and MS degrees from Chongqing Jiaotong University, China, in 2000 and 2007, respectively. Currently, he is a professor of Chongqing Jiaotong University. His current research interests include big data analysis of bridge structure monitoring.

Bharadwaj Veeravalli received the B.Sc. degree in physics, from Madurai Kamaraj University, India, in 1987, the Master's degree in Electrical Communication Engineering from the Indian Institute of Science, Bangalore, India in 1991, and the Ph.D. degree from the Department of Aerospace Engineering, Indian Institute of Science, Bangalore, India, in 1994. He received gold medals for his bachelor degree overall performance and for an outstanding Ph.D. thesis (IISc, Bangalore India) in the years 1987 and 1994, respectively. He is currently with the Department of Electrical and Computer Engineering, Communications and Information Engineering (CIE) division, at The National University of Singapore, Singapore, as a tenured Associate Professor. His main stream research interests include cloud/grid/cluster computing (big data processing, analytics and resource allocation), scheduling in parallel and distributed systems, Cybersecurity, and multimedia computing. He is one of the earliest researchers in the field of Divisible Load Theory (DLT). He is currently serving the editorial board of IEEE Transactions on Parallel & Distributed Systems as an associate editor. He is a senior member of the IEEE and the IEEE-CS.

Keqin Li is a SUNY Distinguished Professor of computer science with the State University of New York. He is also a Distinguished Professor with Hunan University, China. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber-physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU-GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent and soft computing. He has authored or coauthored more than 760 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He has chaired many international conferences. He is currently an associate editor of the ACM Computing Surveys and the CCF Transactions on High Performance Computing. He has served on the editorial boards of the IEEE Transactions on Parallel and Distributed Systems, the IEEE Transactions on Computers, the IEEE Transactions on Cloud Computing, the IEEE Transactions on Services Computing, and the IEEE Transactions on Sustainable Computing. He is an IEEE Fellow.