Collaboration- and Fairness-Aware Big Data Management in Distributed Clouds

Qiufen Xia, *Student Member, IEEE*, Zichuan Xu, *Student Member, IEEE*, Weifa Liang, *Senior Member, IEEE*, and Albert Y. Zomaya, *Fellow, IEEE*

Abstract—With the advancement of information and communication technology, data are being generated at an exponential rate via various instruments and collected at an unprecedented scale. Such large volume of data generated is referred to as big data, which now are revolutionizing all aspects of our life ranging from enterprises to individuals, from science communities to governments, as they exhibit great potentials to improve efficiency of enterprises and the quality of life. To obtain nontrivial patterns and derive valuable information from big data, a fundamental problem is how to properly place the collected data by different users to distributed clouds and to efficiently analyze the collected data to save user costs in data storage and processing, particularly the cost savings of users who share data. By doing so, it needs the close collaborations among the users, by sharing and utilizing the big data in distributed clouds due to the complexity and volume of big data. Since computing, storage and bandwidth resources in a distributed cloud usually are limited, and such resource provisioning typically is expensive, the collaborative users require to make use of the resources fairly. In this paper, we study a novel collaboration- and fairness-aware big data management problem in distributed cloud environments that aims to maximize the system throughout, while minimizing the operational cost of service providers to achieve the system throughput, subject to resource capacity and user fairness constraints. We first propose a novel optimization framework for the problem. We then devise a fast yet scalable approximation algorithm based on the built optimization framework. We also analyze the time complexity and approximation ratio of the proposed algorithm. We finally conduct experiments by simulations to evaluate the performance of the proposed algorithm. Experimental results demonstrate that the proposed algorithm is promising, and outperforms other heuristics.

Index Terms—Big data management, dynamic data placement, fair resource allocation, collaborative users, distributed clouds, data sharing

1 Introduction

Distributed clouds, consisting of multiple datacenters located at different geographical locations and interconnected by high-speed communication routes or links, are emerging as the next-generation cloud platforms, due to their rich cloud resources, resilience to disasters, and low access delay [11], [23], [24], [27]. Meanwhile, with the escalation of data-intensive applications from different organizations that produce petabytes of data, such applications are relying on the rich resources provided by distributed clouds to store and process their petabyte-scale data, i.e., big data management. For instance, the European radio telescope, Low-Frequency Array (LOFAR)) based on a vast array of omni-directional antennas located in Netherlands, Germany, the Great Britain, France and Sweden, produces up to five petabyte of raw data every four hours [16]. Another such an application, Large Hadron Collider in physics research, generates over 60 TB data per day [15]. To share the collected data and obtain valuable insights and scientific findings from such huge volume of data generated from different locations, data users need to upload and process their big data in a distributed cloud for cost savings. Thus, collaborative researchers at different geographic locations can share the data by accessing and analyzing it. A naive placement for such large volume of big data may incur huge costs on data transmission among not only the datacenters but also the collaborated researchers. In addition, the data generated at different locations must be fairly placed to the distributed cloud. Otherwise, biased data placements may severely degrade the reputation of the service provider, thereby reducing the potential revenue of the service provider.

The development of efficient solutions to the mentioned big data management problem is challenging, which lies in several aspects: (i) The *collaboration-aware* users/big data applications dynamically and continuously generate data from different geographical locations, high cost will be incurred when managing such data due to their geo-distributions and large volumes. (ii) Users typically have quality of service (QoS) requirements. Fair usage of cloud services is crucial, otherwise, biased allocation of cloud resources may result in that unsatisfied users no longer use the service, the service provider may fall into disrepute, and its revenue will be significantly reduced. (iii) Processing and analyzing the placed big data require massive computing resource. However, the computing resource in each datacenter typically is limited [4]. If the data placed in a datacenter cannot be processed as required, the overhead on migrating the placed data to other datacenters for processing will be high. (iv) Provisioning adequate computing and network resources for big data applications usually incurs a high operational cost, including the energy cost of powering

- *Q. Xia, Z. Xu, and W. Liang are with the Research School of Computer Science, Australian National University, Canberra, ACT 0200, Australia. E-mail: {qiufen.xia, edward.xu}@anu.edu.au, wliang@cs.anu.edu.au.*
- *A.Y. Zomaya is with the School of Information Technologies, University of Sydney, Sydney, NSW 2006, Australia. E-mail: albert.zomaya@sydney.edu.au.*

Manuscript received 31 May 2015; revised 5 Aug. 2015; accepted 22 Aug. 2015. Date of publication 25 Aug. 2015; date of current version 15 June 2016. Recommended for acceptance by S. Yu.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2015.2473174

servers in datacenters, the hardware cost on switches and routers between datacenters, and the communication cost for transmitting data along Internet links. To address these challenges, in this paper we study the *collaboration- and fairness-aware big data management problem* in a distributed cloud to fairly place continuously generated data to the datacenters of the distributed cloud, the placed data are then processed, and the generated intermediate results finally are utilized by other collaborative users. Our objective is to maximize the *system throughput*, while keeping the operational cost of the service provider minimized, subject to the constraints of resource capacity and fairness among users, where the system throughput is the ratio of the amount of generated data that are successfully placed and processed to the total amount of data generated by each user. In other words, the system throughput is identical for each user, i.e., the proposed algorithm can fairly place the same percentage of data for each user into the system.

Despite that several studies in literature focused on big data management [1], [5], [8], [14], [17], [26], we are not aware of any studies on the collaboration- and fairness-aware big data management problem yet. For example, the studies in [5], [8], [17], [26] focused on data management based on given static data, while the work in [1], [17] did not consider collaborations and the fairness issue among users. They neither take the intermediate results of processed data nor the computing capacity of datacenters into account.

The main contributions of this paper are as follows. We first formulate a novel collaboration- and fairness-aware big data management problem in a distributed cloud. We then propose an optimization framework for the problem, under which we devise an approximation algorithm with a guaranteed approximation ratio, by reducing the problem to the minimum cost multicommodity flow problem. We finally evaluate the performance of the proposed algorithm through experimental simulations. The simulation results show that the performance of the proposed algorithm is promising, which can reduce the operational cost of the distributed cloud significantly. To the best of our knowledge, this is the first time that the collaboration relationships among collaborative users, the fairness guarantee of the placed data, the capacity constraints of datacenters and communication links, and dynamic data generations are jointly considered in distributed cloud environments, and an efficient approximate solution to the problem is delivered.

The remainder of this paper is organized as follows. The system model and the problem definition are introduced in Section 2. The algorithm is proposed in Section 3, followed by evaluating the performance of the proposed algorithm in Section 4. The related work is discussed in Section 5, and the conclusion is given in Section 6.

2 Preliminaries

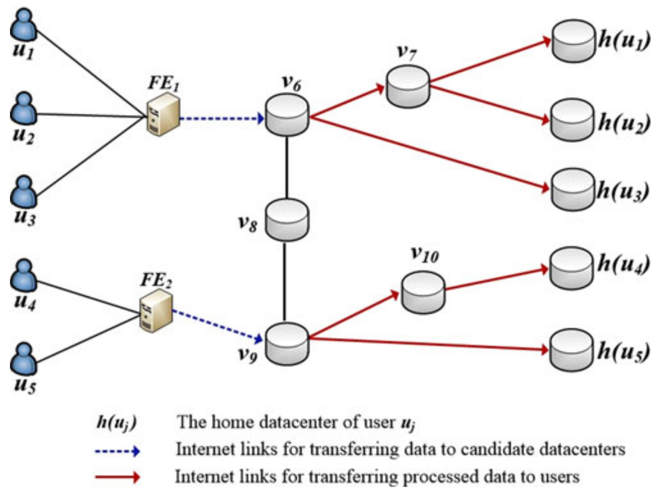
In this section we first introduce the system model, we then describe user collaboration groups and the cost model of data management. We finally define the problem precisely.

2.1 System Model

We consider a distributed cloud $G = (V \cup \mathcal{FE}, E)$, consisting of a number of datacenters located at different geographical locations and interconnected by Internet links, where V and $\mathcal{FE}$ are the sets of datacenters and front-end servers, and E is the set of communication links between datacenters and between datacenters and front-end servers [18]. Let v_i be a datacenter in V and e_{ij} a link in E between datacenters v_i and v_j. The storage and computing resources of each datacenter v_i are used to store and process data, and the bandwidth resource of each link is used for data transmission between datacenters. For the sake of convenience, we here only consider computing and network resources of G as storage resource provisioning is similar to that of computing resource. Denote by $B_c(v_i)$ the capacity of computing resource at datacenter $v_i \in V$, and $B_b(e_{ij})$ the bandwidth resource capacity of link $e_{ij} \in E$. Assuming that time is divided into equal time slots, the amount of available computing resource of datacenter v_i is represented by $A_c(v_i, t)$, and the amount of available bandwidth resource of link e_{ij} is represented by $A_b(e_{ij}, t)$ at time slot t. Let FE_m be a front-end server in $\mathcal{FE}$, where $1 \leq m \leq |\mathcal{FE}|$. Each front-end server FE_m serves as a portal node to the distributed cloud, and has a certain storage capacity to buffer data from its nearby users [9]. Without loss of generality, we assume that the number of front-end servers is proportional to the number of datacenters, i.e., $|\mathcal{FE}| = O(|V|)$.

Users, such as enterprises, organizations and institutions, nowadays are outsourcing their big data to the distributed cloud G. Let u_j be one of such users and $FE_m(u_j)$ the nearest front-end server of u_j, the generated data by u_j are buffered at $FE_m(u_j)$, and the placement of the data to the distributed cloud is scheduled at the beginning of each time slot t. For security concern and ease of management of its data, we assume that there is a set of candidate datacenters specified by each user u_j to place and process its generated data. Let $\mathcal{DC}(u_j)$ be the set of candidate datacenters specified by user u_j. Denote by $S(u_j, t)$ the *dataset* generated by user u_j at time slot t. Each dataset $S(u_j, t)$ can be further split into different fragments (or data blocks) and placed to several candidate datacenters of u_j, due to the limited resource availability at each datacenter [4]. Computing and bandwidth resources are needed to process dataset $S(u_j, t)$ and to transmit related data, let r_c and r_b be the amounts of computing and bandwidth resources allocated to one unit of data [20]. If the accumulated available resources of u_j's candidate datacenters are not enough for the whole dataset $S(u_j, t)$, a proportion of the dataset that cannot be placed at this time slot has to be stored at u_j's nearby front-end server $FE_m(u_j)$ temporarily and will be scheduled later. This procedure continues until all proportions of the dataset are successfully placed and processed. Once dataset $S(u_j, t)$ is placed to datacenters, it will be processed by the datacenters in which it is placed to generate *intermediate results*. We assume that the volume of the intermediate result of a dataset usually is proportional to the volume of the dataset, i.e., $\alpha \times |S(u_j, t)|$ [21], [28], where α is a constant that can be obtained through statistical analysis on the data with $0 < \alpha \leq 1$. The dataset $S(u_j, t)$ then can be removed from the datacenters immediately after its intermediate result has been obtained.

In addition to generating data, each user u_j also requires the intermediate results from its collaborators for further analysis and processing. For example, a metropolitan retail



u_1
u_2
u_3
u_4
u_5
FE_1
FE_2
v_6
v_7
v_8
v_9
v_{10}
$h(u_1)$
$h(u_2)$
$h(u_3)$
$h(u_4)$
$h(u_5)$
$h(u_j)$ The home datacenter of user u_j
Internet links for transferring data to candidate datacenters
Internet links for transferring processed data to users

Fig. 1. A big data management system.

chain consists of many collaborative retailers, each of them not only generates data about customer mobility but also needs intermediate results from its collaborators to better understand in-store traffic patterns and optimize product placements and staffing levels throughout the day, and measure the impact of advertising and special promotions [19]. We thus assume that each user u_j has a *home datacenter* to process and analyze these intermediate results, denote by $h(u_j)$ the home datacenter of u_j. Fig. 1 illustrates the system model, where users u_1, u_2 and u_3 buffer their datasets at the front-end server FE_1, users u_4 and u_5 buffer their datasets at FE_2, the candidate datacenter of u_1, u_2 and u_3 is v_6, the candidate datacenter of u_4 and u_5 is v_9, and the home datacenters of u_1, u_2, u_3, u_4 and u_5 are $h(u_1)$, $h(u_2)$, $h(u_3)$, $h(u_4)$ and $h(u_5)$, respectively. The datasets of u_1, u_2 and u_3 are transferred to their candidate datacenter v_6 to process, the intermediate results obtained from the processed datasets are then transferred to $h(u_1)$, $h(u_2)$ and $h(u_3)$. Similarly, the datasets of u_4 and u_5 are transferred to their candidate datacenter v_9 to process, the intermediate results obtained are then transferred to $h(u_4)$ and $h(u_5)$, respectively.

2.2 Collaborations in Big Data Management

In this paper we consider long-term close collaborations of a group of users. Two collaborative users in the same group need to share, process and analyze the intermediate results of each other in order to derive their own final results on a long term basis. We thus use a *collaboration group* to model a set of collaborative users that make use of the intermediate results of each other. Assume that there are K collaboration groups in the system. Denote by g_k, the kth collaboration group for each k with $1 \leq k \leq K$. For the sake of bandwidth resource savings, we assume that the intermediate result of each user u_j in group g_k will be multicast to all members in the group. In this paper, we consider data users like enterprises, organizations, and institutes, which tend to have more collaborations with their peers to build more wealth and improve daily lives of people. However, in reality, large collaboration groups with many group numbers are hard to form and manage, because many complicated commitments need to be negotiated among the members [16]. We thus assume that the number of members in each group is given as a priori and do not change over time. However, a user may belong to multiple groups, which is common in scientific collaborations where the members in one institution collaborate with many research groups in other institutions.

2.3 Cost Model

Managing datasets incurs the operational cost of a cloud service provider, where the operational cost includes the data storage cost, the data processing cost, and the communication cost of transferring datasets and intermediate results between datacenters. These costs are proportional to the volume of data stored, processed and transferred. Let $c_s(v_i)$ and $c_p(v_i)$ be the storage and processing costs at datacenter v_i for storing and processing one unit of data, and denote by $c_b(e_{ij})$ the cost of occupying one unit of bandwidth along link e_{ij} [2], [3], [28].

Recall that dataset $S(u_j,t)$ of u_j can be split into multiple segments that can be placed and processed in different candidate datacenters of u_j. Let $\lambda_{v_i}(u_j,t)$ be the proportion of dataset $S(u_j,t)$ that will be placed and processed by datacenter v_i at time slot t, then *the storage and processing cost* to store and process $\lambda_{v_i}(u_j,t)$ of dataset $S(u_j,t)$ in datacenter v_i thus is

$$C_1\big(S(u_j,t),v_i\big) = \lambda_{v_i}(u_j,t)\cdot|S(u_j,t)|\cdot\big(c_s(v_i)+c_p(v_i)\big). \quad (1)$$

Note that $|S(u_j,t)|$ represents the volume of dataset $S(u_j,t)$.

The communication cost by transferring a dataset and multicasting its intermediate results through one link $e \in E$ along which data is routed is

$$C_2\big(S(u_j,t),e\big) = \big(\lambda_e(t)\cdot|S(u_j,t)| + \lambda'_e(t)\cdot|S(u_j,t)|\cdot\alpha\big)\cdot r_b\cdot c_b(e), \quad (2)$$

where $\lambda_e(t)$ and $\lambda'_e(t)$ are the proportions of dataset $S(u_j,t)$ and its intermediate results that are routed through link e, and r_b is the amount of bandwidth allocated to one unit of data.

2.4 Problem Definition

Given a distributed cloud $G=(V\cup\mathcal{FE},E)$, a set $\mathcal{U}$ of users, each user $u_j \in \mathcal{U}$ has a set $\mathcal{DC}(u_j)$ of candidate datacenters, a home datacenter $h(u_j)$, and is in a collaboration group g_k. A dataset $S(u_j,t)$ is generated by each user u_j at time slot t, and the computing resource and the bandwidth resource assigned for processing and transferring a unit of data are r_c and r_b, respectively. There are K collaboration groups in the distributed cloud. The *collaboration- and fairness-aware big data management problem* for all groups is to place and process *the same proportion* $\lambda(t)$ of each dataset $S(u_j,t)$ generated by user u_j on its candidate datacenters and to multicast the intermediate results of the processed data to the home datacenters of all users in group g_k such that the value of $\lambda(t)$ is maximized with $0<\lambda(t)\leq 1$. That is, we aim to find the largest $\lambda(t)$ with $\lambda(t)=\lambda(u_1,t)=\lambda(u_2,t)=\cdots=\lambda(u_{|\mathcal{U}|},t)$ such that the volume of dataset $S(u_j,t)$ of each user $u_j\in\mathcal{U}$ that can be placed and processed in the distributed cloud G at time slot t is maximized, where $\lambda(u_j,t)$ is the proportion of dataset $S(u_j,t)$ that will be placed to datacenters

TABLE 1
Symbols

Symbols	Meaning
G	a distributed cloud
V	the set of datacenters in the distributed cloud G
E	the set of links among datacenters, among front-end servers and datacenters
$B_c(v_i)$	the computing resource capacity of datacenter $v_i \in V$
$B_b(e_{ij})$	the bandwidth resource capacity of link $e_{ij} \in E$
t	the current time slot t
$A_c(v_i, t)$	the amount of available computing resource of datacenter v_i at time slot t
$A_b(e_{ij}, t)$	the amount of available bandwidth resource of link e_{ij}
FE	a front-end server that serves as portal node
$\mathcal{FE}$	the set of front-end servers
$\lvert\mathcal{FE}\rvert$	the number of front-end servers in the system
u_j	a user in the system
$FE_m(u_j)$	the nearest front-end server of user u_j
$\mathcal{DC}(u_j)$	the set of candidate datacenters of user u_j
$h(u_j)$	the home datacenter of user u_j
$S(u_j, t)$	the dataset generated by user u_j at time slot t
$\lvert S(u_j, t)\rvert$	the size of dataset $S(u_j, t)$
U	the set of users in the system
α	the proportion between the volume of a source dataset and the volume of its processed intermediate results
p_{FE_m, v_i}	a shortest path between a front-end server FE_m and a candidate datacenter v_i in the distributed cloud G
$T(v_i, t)$	a multicast tree for transferring the intermediate results rooted at a candidate datacenter v_i in G
g_k	the kth collaboration group, $1 \le k \le K$
K	the number of groups in the system
r_c	the amount of computing resource allocated to one unit of data
r_b	the amount of bandwidth resource allocated to one unit of data
$c_s(v_i)$	the storage cost at datacenter v_i
$c_p(v_i)$	the processing cost at datacenter v_i
$c_b(e)$	the cost of occupying one unit of bandwidth along link e_{ij} per time sot
$\lambda_{v_i}(u_j, t)$	the proportion of dataset $S(u_j, t)$ that will be placed and processed by datacenter v_i
$\lambda(u_j, t)$	the total proportion of dataset $S(u_j, t)$ that will be placed and processed by the distributed cloud G
$\lambda(t)$	the system throughput at each time slot t
G_f	the constructed auxiliary flow graph
V_f	the vertex in G_f
E_f	the edges in G_f
s_0	the virtual source node in V_f
$FE^f_m(u_j)$	a virtual front-end node in V_f for each user u_j whose data are buffered at FE_m
v^f_i	a datacenter node in V_f that corresponds to a candidate datacenter v_i in the distributed cloud G
v^f_{i,g_k}	a virtual datacenter node in V_f
t_0	the virtual destination node in V_f
$f'(e)$	the flow through an edge e in G_f
ρ_{min}	the minimum overflow ratio

at time slot t and $\lambda(u_j, t) = \sum_{v_i \in \mathcal{DC}(u_j)} \lambda_{v_i}(u_j, t)$, the problem optimization objective thus is to

$$maximize \quad \lambda(t), \tag{3}$$

while keeping the operational cost $C(t)$ of the cloud service provider minimized, subject to both computing and bandwidth resource capacity constraints, where

$$C(t) = \sum_{u_j \in \mathcal{U}} \left(\sum_{v_i \in \mathcal{DC}(u_j)} C_1\big(S(u_j, t), v_i\big) + \sum_{e \in E} C_2\big(S(u_j, t), e\big) \right). \tag{4}$$

The maximum value of $\lambda(t)$ is referred to as the *system throughput*, which is the ratio of the amount of placed and processed data to the amount of the generated data by each user. The *fairness* here is referred to the same proportional of the dataset of each user will be placed and processed in the distributed cloud at each time slot.

Notice that the occupied storage, computing and bandwidth resources by the placed and processed data will be released when data storage, processing and transmission are finished. Without loss of generality, following the similar assumptions in [27], [28], we assume that all data processing and transmission can be finished within one time slot. This can be achieved by adjusting the length of each time slot. The symbols used in this paper are summarized in Table 1.

3 Approximation Algorithm

In this section, we first propose a novel optimization framework for the collaboration- and fairness-aware big data management problem. We then develop an efficient approximation algorithm with a guaranteed approximation ratio, based on the proposed optimization framework. We finally analyze the time complexity and approximation ratio of the proposed algorithm.

3.1 The Optimization Framework

Intuitively, a solution to the collaboration- and fairness-aware big data management problem consists of two phases: (i) upload the dataset of each user u_j to its candidate datacenters for processing; and (ii) multicast the intermediate results obtained from the processed datasets to the home datacenters of users in the collaboration group to which u_j belongs. A naive solution thus is to optimize these two phases separately, which will result in a sub-optimal solution. For example, given a user u_j in group g_k, assuming phase (i) places datasets of user u_j into a datacenter that is far away from the home datacenters of other users in group g_k, phase (ii) may incur a high communication cost for multicasting the intermediate results from the placed datasets to these home datacenters.

To provide a better solution to the problem, in the following we propose a novel optimization framework by jointly considering these two phases. The optimization framework essentially is to reduce the collaboration- and fairness-aware big data management problem in a distributed cloud $G = (V \cup \mathcal{FE}, E)$ to the minimum cost multicommodity flow problem in an auxiliary flow network $G_f = (V_f, E_f; u, c)$ with a cost function $c : E \mapsto \mathbb{R}^{\geq 0}$ and a capacity function $u : E \mapsto \mathbb{R}^{+}$, where the construction of G_f consists of two stages, which is described as follows.

We start by its construction in the first stage. Given each front-end server $FE_m \in \mathcal{FE}$ in G, there is a *virtual front-end node* $FE^f_m(u_j)$ in V_f for each user u_j whose data are buffered at FE_m. All virtual front-end nodes and a *virtual source node* s_0 are added to G_f. There is an edge in E_f from s_0 to each $FE^f_m(u_j)$ with the cost zero while the capacity of the edge is set as the total volume of datasets generated by all users in $\mathcal{U}$ at time slot t, i.e., $u(s_0, FE^f_m(u_j)) = \sum_{u_j \in \mathcal{U}} |S(u_j, t)|$. For each candidate datacenter v_i in G, there is a *datacenter node* v^f_i in G_f. For each datacenter node v^f_i and group g_k, if the corresponding datacenter v_i of v^f_i is a candidate datacenter of any user in g_k, there is a *virtual datacenter node* v^f_{i,g_k} for group g_k, e.g., datacenter v_0 is the candidate datacenter of users u_1 and u_2, u_1 and u_2 are in groups g_1 and g_2 respectively, then, two virtual datacenter nodes v^f_{0,g_1} and v^f_{0,g_2} are added to G_f. An edge from each virtual front-end server $FE^f_m(u_j)$ to each virtual datacenter node v^f_{i,g_k} is added to G_f if u_j is a member of group g_k and its corresponding datacenter v_i is one of the candidate datacenters of u_j. Each such an edge, e.g., $\langle FE^f_m(u_j), v^f_{i,g_k}\rangle$, represents the shortest routing path from FE_m to datacenter v_i in distributed cloud G, denoted by p_{FE_m,v_i}. Its cost thus is the accumulative communication cost of all links in the shortest path, i.e., $c(FE^f_m(u_j), v^f_{i,g_k}) = \sum_{e \in p_{FE_m,v_i}} c_t(e)$, and its capacity is the amount of data that can be routed through a bottleneck link in the shortest path that has the minimum available bandwidth, i.e., $u(FE^f_m(u_j), v^f_{i,g_k}) = \min_{e \in p_{FE_m,v_i}} \{\frac{A_b(e,t)}{r_b}\}$.

We then proceed the construction of G_f in its second stage, i.e., multicasting the intermediate results obtained from the processed source datasets, which is crucial to maximize the system throughput, as multicasting consumes the bandwidth resource of links in the distributed cloud G. There are two issues associated with multicasting in G_f: One is how to handle multiple multicasts of intermediate results sourced from each candidate datacenter within each group g_k; and the other is how to deal with the volume differences between the source data and their intermediate results.

For the first issue, we consider the case where the intermediate results of multiple users in the same group g_k at each candidate datacenter v_i will be transferred through a shared multicast tree $T(v_i, t)$, because they share identical (multicast source) root v_i and the same terminal set g_k. To this end, add an edge from each virtual datacenter node v^f_{i,g_k} to its corresponding datacenter node v^f_i to the edge set E_f, this edge represents a multicast tree that multicasts the intermediate results from source node v_i in G to other terminal nodes in g_k, where the source data are processed and the intermediate results are generated at v_i. The terminals in a multicast tree are the home datacenters of users in group g_k.

For the second issue, we assign the capacity of edge $\langle v^f_{i,g_k}, v^f_i\rangle$ in E_f, which is the minimum available capacity of links in the multicast tree $T(v_i, t)$. Specifically, the actual capacity of edge $\langle v^f_{i,g_k}, v^f_i\rangle$ is the amount of data that can be routed through the bottleneck link in tree $T(v_i, t)$. Here, we relax the capacity of edge $\langle v^f_{i,g_k}, v^f_i\rangle$ by $1/\alpha$ times of the capacity of its bottleneck link, since we route data through a path in the auxiliary graph G_f without considering the change of the data volume, while in reality, the volume of an intermediate result in the multicasting tree usually is the proportional of the volume of its dataset, i.e., $\alpha \times |S(u_j, t)|$ [21], [28], where α is a constant with $0 < \alpha \leq 1$. Therefore, the capacity of edge $\langle v^f_{i,g_k}, v^f_i\rangle$ is $u(v^f_{i,g_k}, v^f_i) = \frac{\min_{e \in T(v_i,t)}\{A_b(e,t)\}}{\alpha \cdot r_b}$, where r_b is the amount of bandwidth assigned to a unit of data. The cost of edge $\langle v^f_{i,g_k}, v^f_i\rangle$ is the accumulative cost of all edges in $T(v_i, t)$, i.e., $c(v^f_{i,g_k}, v^f_i) = \sum_{e \in T(v_i,t)} c_t(e)$.

A *virtual sink node* t_0 finally is added to G_f. For each datacenter node $v^f_i \in V_f$, there is an edge from v^f_i to t_0. The cost of each edge $\langle v^f_i, t_0\rangle$ is the cost of storing and processing a unit of data at datacenter v_i, that is $c(v^f_i, t_0) = c_s(v_i) + c_p(v_i)$. Its capacity is the total volume of data that can be processed by the available computing resource of $v_i \in V$ at time slot t, i.e., $u(v^f_i, t_0) = \frac{A_c(v_i,t)}{r_c}$, where r_c is the amount of computing resource allocated to one unit of data.

An example of the distributed cloud $G = (V \cup \mathcal{FE}, E)$ and the construction of the auxiliary flow network $G_f = (V_f, E_f; u, c)$ are shown in Fig. 2, where both users u_1 and u_2 are in group g_1, while users u_3 and u_4 are in group g_2. The home datacenters of u_1, u_2, u_3 and u_4 are $h(u_1)$, $h(u_2)$, $h(u_3)$ and $h(u_4)$ respectively. The candidate datacenters of user u_1 are v_3 and v_4, the candidate datacenters of user u_2 are v_4 and v_5. the candidate datacenters of user u_3 are v_5 and v_{10}, and the candidate datacenter of user u_4 is v_{13}. Recall that $FE^f_1(u_1)$ represents the front-end server where the dataset $S(u_1, t)$ of u_1 are buffered, the edge $\langle FE^f_1(u_1), v^f_{3,g_1}\rangle$ of G_f corresponds to placing a proportion of $S(u_1, t)$ from $FE_1(u_1)$ to the candidate datacenter v_3 of user u_1 in G, while

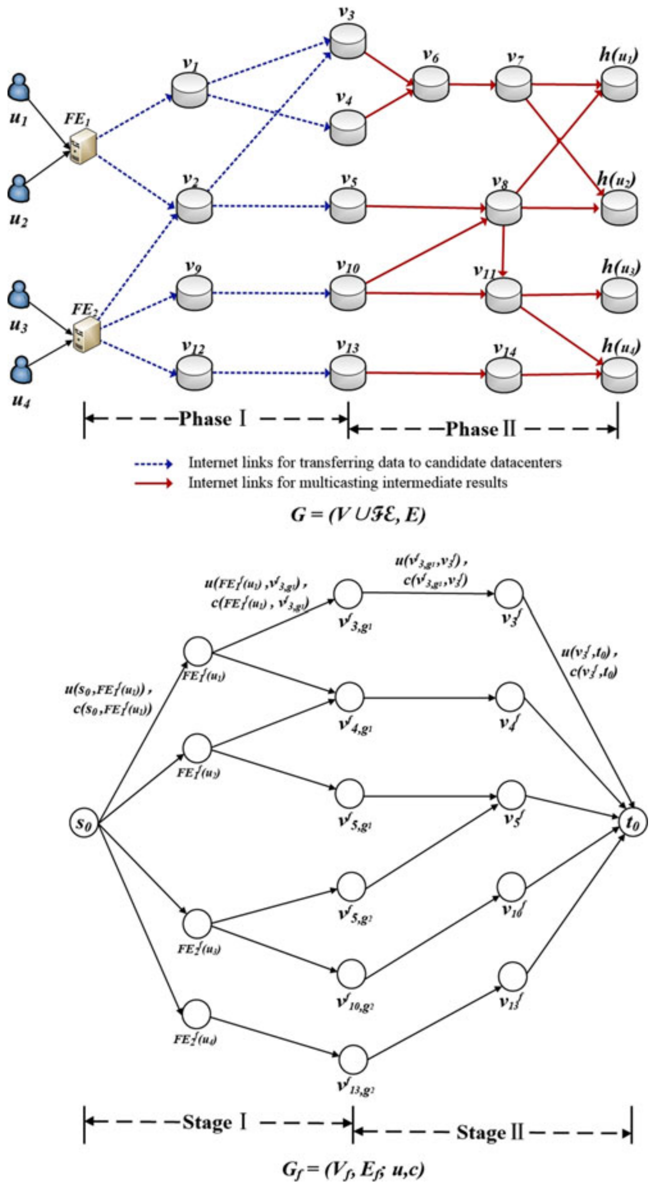

Phase I
Phase II
Internet links for transferring data to candidate datacenters
Internet links for multicasting intermediate results
G = (V ∪ 𝓕𝓔, E)
Stage I
Stage II
G_f = (V_f, E_f; u,c)

Fig. 2. An example of the construction of an auxiliary flow network $G_f = (V_f, E_f; u, c)$, where users u_1 and u_2 are in group g_1, users u_3 and u_4 are in group g_2, i.e., $g_1 = \{u_1, u_2\}$, $g_2 = \{u_3, u_4\}$, the sets of candidate datacenters of users u_1, u_2, u_3 and u_4 are $\mathcal{DC}(u_1) = \{v_3, v_4\}$, $\mathcal{DC}(u_2) = \{v_4, v_5\}$, $\mathcal{DC}(u_3) = \{v_5, v_{10}\}$, and $\mathcal{DC}(u_4) = \{v_{13}\}$, respectively. The home datacenters of u_1, u_2, u_3 and u_4 are $h(u_1)$, $h(u_2)$, $h(u_3)$ and $h(u_4)$, respectively.

$\langle FE_1^f(u_1), v_{4,g_1}^f \rangle$ of G_f corresponds to placing another proportion of $S(u_1, t)$ from $FE_1(u_1)$ to the candidate datacenter v_4 of user u_1 in G. Similarly, $\langle v_{3,g_1}^f, t_0 \rangle$ in G_f corresponds to processing data on v_3 and multicasting the intermediate results processed by v_3 to the home datacenters of users u_1 and u_2 in G. Similarly, $\langle v_{4,g_1}^f, t_0 \rangle$ in G_f corresponds to processing data on v_4 and multicasting the intermediate results processed by v_4 to home datacenters $h(u_1)$ and $h(u_2)$ of users u_1 and u_2 in G. The analysis of other edges is similar as that of $\langle FE_1^f(u_1), v_{3,g_1}^f \rangle$, $\langle v_{3,g_1}^f, t_0 \rangle$, $\langle FE_1^f(u_1), v_{4,g_1}^f \rangle$ and $\langle v_{4,g_1}^f, t_0 \rangle$.

Notice that the proposed optimization framework can be easily extended to the scenario where a user belongs to multiple groups, for which we only need to modify the flow network G_f to an *augmented auxiliary flow network* G'_f. The only difference between G_f and G'_f is that, for a user u_j in multiple groups that buffers its dataset at the front-end server $FE_m(u_j)$, there are multiple virtual candidate datacenter nodes and one virtual front-end node $FE_m^f(u_j)$. A set of edges are added, where there is only one edge connecting to $FE_m^f(u_j)$ from one of the multiple virtual candidate datacenter nodes. For example, let u_1 be the common user in groups g_1 and g_2, let $FE_1(u_1)$ be the front-end server of u_1 and v_1 the candidate datacenter of u_1. The constructed auxiliary flow network G_f is shown in Fig. 3a. The construction of the augmented auxiliary flow network G'_f is as follows. Remove edges $\langle FE_1^f(u_1), v_{1,g_2}^f \rangle$ and $\langle v_{1,g_1}^f, v_1^f \rangle$ from G_f, and add edge $\langle v_{1,g_1}^f, v_{1,g_2}^f \rangle$ into it, as shown in Fig. 3b. The capacity and cost of edge $\langle v_{1,g_1}^f, v_{1,g_2}^f \rangle$ are set to their corresponding ones of the removed edge $\langle v_{1,g_1}^f, v_1^f \rangle$.

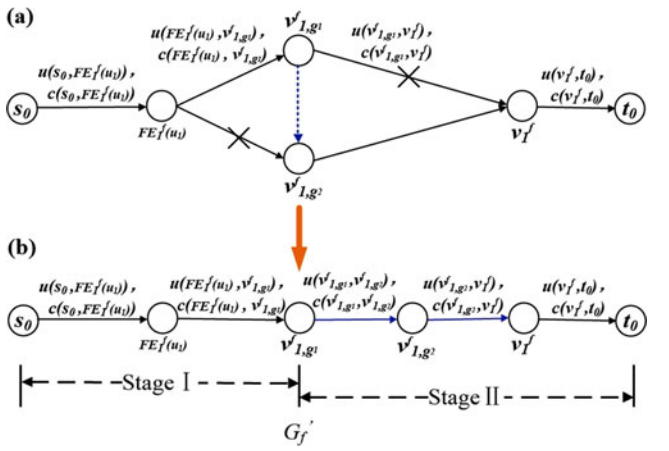


Fig. 3. An example of the augmented auxiliary graph G'_f, where user u_1 is in groups g_1 and g_2, the candidate datacenter of u_1 is v_1, the front-end server of u_1 is $FE_1(u_1)$.

3.2 Approximation Algorithm

In the following we reduce the collaboration- and fairness-aware big data management problem in G to a minimum cost multicommodity flow problem in G_f. The detailed reduction is given as follows.

Let $\mathcal{U}(FE_m)$ be the set of users whose datasets are buffered at the front-end server FE_m. The data of each user is treated as *a commodity* with demand $|S(u_j, t)|$ at source node $FE_m^f(u_j)$ in G_f, which will be routed to the common destination t_0. There are $\sum_{m=1}^{|\mathcal{FE}|} |\mathcal{U}(FE_m)|$ commodities in G_f to be routed to t_0. Suppose that f is a maximum flow with minimum cost from s_0 to t_0 that routes the commodities from different source nodes to their common destination t_0, and the constructed flow network G_f satisfies the flow conservation: for any vertex $v^f, v_0^f \in V_f \setminus \{s_0, t_0\}$, we have $\sum_{v_0^f \in V_f} f(v^f, v_0^f) = \sum_{v_0^f \in V_f} f(v_0^f, v^f)$. For each commodity, f implies multiple routing paths from s_0 to t_0 in G_f with each such a path p corresponding to processing a proportion of a user's dataset and multicasting the intermediate results of the dataset to other group members of the user. However, considering routing all commodities, the value of $|f|$ of flow f may not be a feasible solution to the collaboration- and fairness-aware big data management problem. This is because paths from s_0 to t_0 in G_f may correspond to routing paths in distributed cloud G that share physical links in G, and the physical link capacities may have been violated if the links are shared by many routing paths (we will provide a formal proof later).

Algorithm 1. *An Approximation Algorithm for the Collaboration- and Fairness-Aware Big Data Management Problem at Each Time Slot t*

```
Input: A distributed cloud $G = (V \cup \mathcal{FE}, E)$ with a set $V$ of datacenters, a set $\mathcal{FE}$ of front-end servers serving as portals to the datacenters, and a set $E$ of communication links; A number $K$ of collaboration groups of data users with each group $g_k$ consisting of data users to share data with each other; A set of users $\mathcal{U}$ who generated a set of datasets to be placed at each time slot $t$. The accuracy parameter $\epsilon$ with $0 < \epsilon \leq 1$.
Output: The proportion $\lambda(t)$ of datasets that are placed and processed, the minimum operational cost $C$, and the datacenters for storing and processing the data.
 1: Find all shortest paths from each front-end server $FE_m \in \mathcal{FE}$ to the candidate datacenters of all users;
 2: For each candidate datacenter $v_i$ and each user $u_j$ who is in group $g_k$ and specifies $v_i$ as a candidate datacenter, find a multicast tree whose root is $v_i$ and the terminal set is the home datacenters of users in $g_k$;
 3: Construct an auxiliary flow network $G_f$, in which there are $\sum_{m=1}^{|\mathcal{FE}|} |\mathcal{U}(FE_m)|$ commodities to be routed from their source node $FE^f_m(u_j)$ to the destination $t_0$;
 4: Let $f$ be the minimum cost flow for the minimum cost multicommodity flow problem in $G_f$ delivered by applying Garg and Könemann's algorithm [7];
 5: According to $f$, calculate the amount of data routed through each link $e \in E$ of distributed cloud $G$, where the amount of flow through edge $\langle FE^f_m(u_j), v^f_{i,g_k}\rangle$ equals to the amount of data routed through the shortest path from $FE_m$ to datacenter $v_i$ of $G$, and $\alpha$ proportion of the amount of flow through edge $\langle v^f_{i,g_k}, v^f_i\rangle$ equals to the amount of data routed through the multicast tree whose root is $v^f_i$ and terminal set is the home datacenters of users in $g_k$;
 6: Let $f'(e)$ be the amount of all data routed through each link $e \in E$ of $G$, find the overflow ratio $\rho(e)$ of the link, $\rho(e) = \frac{A_b(e,t)}{f'(e) \cdot r_b}$;
 7: $\rho_{min} \leftarrow \min\{\rho(e) \mid e \in E\}$; /* $\rho_{min}$ is the minimum overflow ratio */
 8: $|f'(e)| \leftarrow |f'(e)| \cdot \rho_{min}, \forall e \in E$; /* scale down the flow */
 9: Calculate the amount of routed data of $u_j$, and $\lambda(t)$ that is the ratio of the amount of routed data to the size of source dataset $S(u_j, t)$.
10: Calculate the cost by Eq. (4).
```

To obtain a feasible solution, flow f will be scaled down by an appropriate *scale factor* so that the resulting flow f becomes feasible. Specifically, we first map flow f to actual data routing in the original distributed cloud G. The amount of flow that goes through edge $\langle FE^f_m(u_j), v^f_{i,g_k}\rangle$ equals the amount of data routed through the shortest path in G from FE_m to datacenter v_i, since each edge $\langle FE^f_m(u_j), v^f_{i,g_k}\rangle$ in G_f represents the shortest path in G from FE_m to v_i. Similarly, The α proportion of the amount of flow that goes through edge $\langle v^f_{i,g_k}, v^f_i\rangle$ in G_f equals the amount of data routed through the multicast tree rooted at v^f_i with the terminal set consisting of the home datacenters of users in g_k. We then find the minimum *overflow ratio* of all edges in E of G, i.e., $\rho_{min} = \min_{e \in E}\{\frac{A_b(e,t)}{f'(e) \cdot r_b}\}$, where $f'(e)$ is the flow through edge e. Notice that in terms of "fair" big data management, we finally scale down the flow in all edges a ratio ρ_{min}. The proposed approximation algorithm is described in **Algorithm 1**.

3.3 Algorithm Analysis

We now first show the correctness of the proposed algorithm. We then analyze the performance and time complexity of Algorithm 1.

Lemma 1. *Given the auxiliary flow network $G_f = (V_f, E_f; u, c)$ derived from the distributed cloud $G = (V \cup \mathcal{FE}, E)$ and the capacity and cost settings of G_f, assume that the dataset of a user $u_j \in \mathcal{U}$ is buffered in the front-end server $FE_m(u_j)$, then, each path p in G_f from s_0 to t_0 derived by flow f corresponds to the processing of a proportion of dataset $S(u_j, t)$ and multicasting of the intermediate results of the processed data to the home datacenters of other users in the group of u_j.*

Proof. We first show that the amount of flow (data) entering to each datacenter node v^f_i in G_f will be processed by datacenter v_i in G. For the sake of clarity, we assume that u_j is in group g_k and one of its candidate datacenters is v_i. From the construction of G_f, it can be seen that there is a directed edge from the virtual front-end server node $FE^f_m(u_j)$ of dataset $S(u_j, t)$ to each virtual datacenter node v^f_{i,g_k}, and a directed edge from virtual datacenter node v^f_{i,g_k} to its corresponding datacenter node v^f_i. The value of the fractional flow on a path p from s_0 to v^f_i is the amount of data that are routed to datacenter v_i in G for processing. In addition, there is a directed edge from v^f_i to virtual sink t_0, which means that the proportional of dataset $S(u_j, t)$ represented by the fractional flow f has been processed by v_i when flow f goes through edge $\langle v^f_i, t_0\rangle$.

We then show that the α proportion of flow f entering into edge $\langle v^f_{i,g_k}, v^f_i\rangle$ of path p corresponds to the intermediate results, which will be multicast from datacenter v_i to all home datacenters of the users in g_k. Recall that the volume of the intermediate result is α proportional of the amount of data to be processed, where α is a constant with $0 < \alpha \leq 1$. To guarantee that all intermediate results can be routed along edge $\langle v^f_{i,g_k}, v^f_i\rangle$ that represents a multicast tree in the distributed cloud G without violating the edge capacity constraint, we relax the capacity of edge from $\langle v^f_{i,g_k}, v^f_i\rangle$ to $\frac{\min_{e \in T(v_i,t)}\{A_b(e,t)\}}{\alpha \cdot r_b}$, while the actual amount of data that will be sent through the multicast tree in G is $|f| \times \alpha$, which is the volume of intermediate results that are analyzed from the data in flow f (i.e., the data processed in datacenter v^f_i). □

Lemma 2. *Given the auxiliary flow network $G_f = (V_f, E_f; u, c)$ derived from the distributed cloud $G = (V \cup \mathcal{FE}, E)$, the* *capacity and cost settings of edges in G_f, and users in $\mathcal{U}$ whose data to be processed and multicast, there is a feasible solution to the collaboration- and fairness-aware big data management problem if none of the routing paths and multicast trees share links in G; otherwise, the solution may not be feasible.*

Proof. We first show that there is no resource sharing between routing paths for dataset transfer and multicast trees, since the processing of a dataset in a datacenter can start after the dataset has been received by the datacenter.

We then analyze a case where no links are shared among all routing paths in G from front-end servers to candidate datacenters and the multicast trees rooted at the candidate datacenters to the home datacenters of the users in each collaboration group. This means that no two fractional flows in G_f, say f_1 and f_2 from s_0 to t_0, corresponding to paths and multicast trees in G that share links in the distributed cloud G. Also, by Lemma 1, each path in G_f from s_0 to t_0 derived by flow f corresponds to placing and processing a proportion of a dataset and multicasting the processed intermediate results to the home datacenters of all users in that collaboration group. Thus, a feasible flow f in G_f from s_0 to t_0 corresponds to a feasible solution to the problem of concern.

We finally show the case where both the paths and multicast trees share links. Suppose there are two shortest paths p_1 and p_2 from front-end servers of users u_1 and u_2 to one common candidate center v_1 in the distributed cloud G, and a link $e \in E$ is shared by both p_1 and p_2. Let g_1 and g_2 be the two groups in which u_1 and u_2 are, and let $FE_1(u_1)$ and $FE_2(u_2)$ be the front-end servers of u_1 and u_2, respectively. In the construction of G_f, p_1 and p_2 are denoted by edges $\langle FE_1^f(u_1), v_{1,g_1}^f\rangle$ and $\langle FE_2^f(u_2), v_{1,g_2}^f\rangle$, respectively. Their capacities are set the amounts of available bandwidth resources of the bottleneck links for p_1 and p_2. We assume link e is the bottleneck link of both p_1 and p_2 in G. A feasible flow f from s_0 to t_0 will saturate both edges $\langle FE_1^f(u_1), v_{1,g_1}^f\rangle$ and $\langle FE_2^f(u_2), v_{1,g_2}^f\rangle$. This however will overflow the bandwidth capacity of link e, due to the fact that link e is the bottleneck link of paths p_1 and p_2 in G. The lemma thus follows. □

The rest is to analyze the approximation ratio and time complexity of the proposed algorithm. For the sake of completeness, in the following we first introduce Theorem 1 [7] and then elaborate Theorem 2.

Theorem 1. *(see [7]) There is an approximation algorithm for the minimum cost multicommodity flow problem in a directed graph $G = (V, E; u, c)$ with n commodities to be routed from their sources to their destinations. The algorithm delivers an approximate solution with an approximation ratio of $(1 - 3\epsilon)$ while the associated cost is the minimum one, and the algorithm takes $O^*(\epsilon^{-2}m(n + m))$ time,*[1] *where $m = |E|$ and ϵ is a constant accuracy parameter that is moderately small.*

Theorem 2. *Given a distributed cloud $G = (V \cup \mathcal{FE}, E)$ consisting of $|V|$ datacenters, $|\mathcal{FE}|$ front-end servers, and a set $\mathcal{U}$ of users in K collaboration groups, there is an approximation algorithm with the approximation ratio $\frac{1}{|\mathcal{U}||V|} - \epsilon'$ for the collaboration- and fairness-aware big data management problem in G, while the cost of achieving the system throughput is no more than 2 times of the optimal solution C^*. The time complexity of the proposed algorithm is $O^*(\epsilon^{-2}(K|\mathcal{FE}||V||\mathcal{U}| + K^2|\mathcal{FE}|^2|V|^2) + |V|^3)$, where ϵ' is constant with $\epsilon' = \frac{3\epsilon}{|\mathcal{U}||V|}$.*

Proof. We first analyze the approximation ratio of Algorithm 1 in terms of the system throughput. Let f' and f^* be a feasible and the optimal solutions to the problem. Let f be the flow in G_f delivered by Garg and Könemann's algorithm, then $|f| \geq (1 - 3\epsilon)|f^*|$ by Theorem 1, where ϵ is a constant accuracy parameter. Note that if f is infeasible, it becomes feasible by scaling a factor of ρ_{min}. Thus, there is a feasible solution f' to the problem (i.e., the system throughput $|f'| \geq \rho_{min}(1 - 3\epsilon)|f^*|$). The rest is to show that $\rho_{min} \geq \frac{1}{|\mathcal{U}||V|}$ by the two cases: (i) all shortest paths from front-end servers to candidate datacenters share one common bottleneck link in G; (ii) all multicast trees to collaboration groups share one common bottleneck link in G. For case (i), recall that a set $\mathcal{U}$ of users form K groups. If all these $|\mathcal{U}|$ users have data to be processed to their candidate data centers, there will be at most $|\mathcal{U}||V|$ shortest routing paths in G from front-end servers to candidate datacenters. We thus have $\rho_{min} = \frac{1}{|\mathcal{U}||V|}$. For case (ii), the worst case is that each user has its intermediate results generated at all the $|V|$ data centers, and all users are multicasting their intermediate results to their collaborators. In total, there will be $|\mathcal{U}||V|$ multicast sessions sharing one bottleneck link in the worst case. Thus, $\rho_{min} = \frac{1}{|\mathcal{U}||V|}$. By taking both cases (i) and (ii) into consideration, we have $\rho_{min} = \frac{1}{|\mathcal{U}||V|}$. The approximation ratio of the proposed algorithm thus is $\rho_{min}(1 - 3\epsilon) = \frac{1}{|\mathcal{U}||V|} - \epsilon'$, where $\epsilon' = \frac{3\epsilon}{|\mathcal{U}||V|}$.

We then show the approximation ratio of the cost to achieve the specified system throughput. From Theorem 1, we know that the cost of the feasible flow f in G_f is minimized. This however does not mean that the cost of managing datasets is minimized too, since finding a minimum-cost Steiner tree is a classic NP-hard problem. Let C_1 be the cost of transferring and processing the data of users in $\mathcal{U}$, and C_2 be the cost of transferring intermediate results by algorithm 1. Then, $C = C_1 + C_2$. Denote by C_1^* and C_2^* the corresponding components in the optimal solution of C_1 and C_2. We then have $C_1 = C_1^*$ as the datasets are routed to candidate datacenters through the shortest paths in G. Similarly, $C_2 \leq 2C_2^*$ following the well-known approximate solution to find a minimum-cost terminal Steiner tree. We thus have $\frac{C}{C^*} = \frac{C_1 + C_2}{C_1^* + C_2^*} \leq \frac{C_1^* + 2C_2^*}{C_1^* + C_2^*} \leq 2$.

We finally analyze the running time of the proposed algorithm. The most time consuming component in the construction of G_f is to find the shortest paths from front-end servers to candidate datacenters and multicast trees for all groups. The construction of G_f thus takes $O(K^2|\mathcal{FE}||V|^2 + |V|^3)$. Delivering a feasible flow on G_f takes $O^*(\epsilon^{-2}m(n + m))$ time by Theorem 1, where

1. $O^*(f(n)) = O(f(n)\log^{O(1)} n)$.

$n = \sum_{i=1}^{|\mathcal{FE}|} |\mathcal{U}(FE_m)| = |\mathcal{U}|$ and $m = K|\mathcal{FE}||V|$. The time complexity of the proposed algorithm thus is $O^*(\epsilon^{-2}(K|\mathcal{FE}||V||\mathcal{U}| + K^2|\mathcal{FE}|^2|V|^2) + |V|^3)$. □

4 Performance Evaluation

In this section, we evaluate the performance of the proposed algorithm, and investigate the impact of important parameters on the algorithmic performance.

4.1 Simulation Environment

We consider a distributed cloud consisting of 20 datacenters and 10 front-end servers, there is an edge between each pair of nodes with a probability of 0.2 generated by the GT-ITM tool [10]. The computing capacity of each datacenter and the bandwidth capacity of each link are randomly drawn from value intervals [1,000, 2,000] units (GHz), and [1, 10] units (Gbps), respectively [6], [12], [25]. We set each time slot as one hour [28]. Each user produces several Gigabytes of data per time slot, we thus emulate the volume of dataset generated by each user per time slot is in the range of [4, 8] GB [28], and the amount of computing resource assigned to the processing of 1GB data is a value in the range of [8, 12] GHz. These generated data will be placed and processed from one to five datacenters. The costs of transmitting, storing and processing 1 GB of data are set within [\$0.05, \$0.12], [\$0.001, \$0.0015], and [\$0.15, \$0.22], respectively, following typical charges in Amazon EC2 and S3 with small variations [2], [3]. Unless otherwise specified, we will adopt these default settings in our experiments. Each value in the figures is the mean of the results by applying the mentioned algorithm 15 times on 15 different topologies of the distributed cloud.

To evaluate the performance of the proposed algorithm, referred to as `Appro-Alg`, two heuristics are employed as evaluation baselines. One is to choose a candidate datacenter with the maximum amount of available computing resource, and then place as much data of a user as possible to the datacenter. If the datacenter cannot accommodate the whole dataset of the user, it then picks the next candidate datacenter with the second largest amount of available computing resource. This procedure continues until the dataset is placed or there is not any available computing resource in the set of candidate datacenters of this user. We refer to this heuristic as `Greedy-Alg`. Another is to select a candidate datacenter randomly and place as much datasets of a user as possible to the datacenter. If the available computing resource in the chosen datacenter is not enough to process all datasets, it then chooses the next one randomly. This procedure continues until there is no available computing resource in the set of candidate datacenters of this user, or all data of this user are placed to the distributed cloud. We refer to this algorithm as `Random-Alg`.

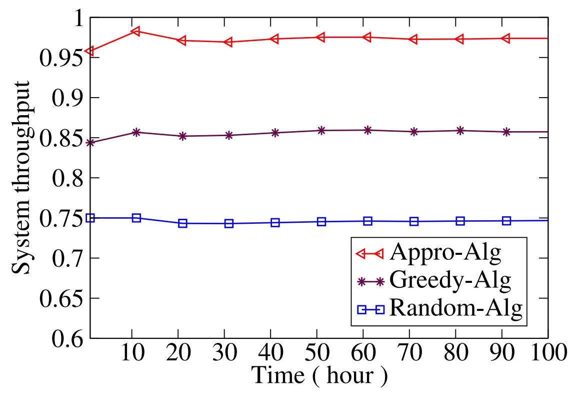
4.2 Algorithm Performance Evaluation

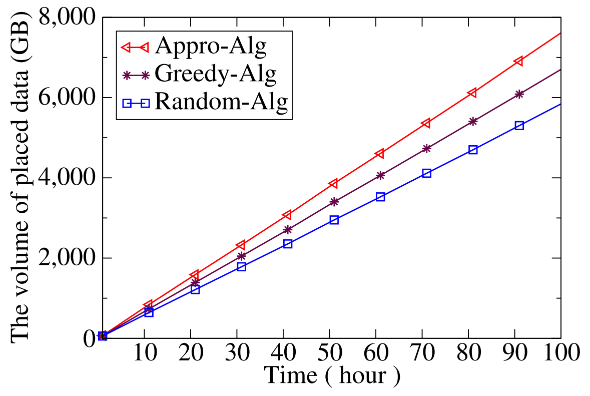
We first evaluate the performance of different algorithms, the amount of placed data, the operational cost, and the average operational cost of placing one unit of data by these algorithms. Fig. 4a plots the curves of system throughput delivered by the three mentioned algorithms `Appro-Alg`, `Greedy-Alg` and `Random-Alg`, from which it can be seen that the algorithm `Appro-Alg` achieves a nearly optimal system throughput of 98 percent, which is higher than those of algorithms `Greedy-Alg` and `Random-Alg` by 13 and 23 percent, respectively. Fig. 4b shows the volume of placed data by the three comparison algorithms, from which it can be seen that the volumes of placed data by algorithms `Appro-Alg`, `Greedy-Alg`, and `Random-Alg` are 7,600, 6,700 and 5,800 GB, respectively. Specifically, the volume of placed data by algorithm `Appro-Alg` is 14 and 31 percent larger than those by algorithms `Greedy-Alg` and `Random-Alg`. Fig. 4c illustrates the operational cost of these algorithms. It can be seen that at time slot 100 the operational costs of algorithms `Appro-Alg`, `Greedy-Alg` and `Random-Alg` are \$4,300, \$3,850 and \$3,450. These figures demonstrate that although the operational cost of algorithm `Appro-Alg` is 12 and 25 percent higher than that of the two benchmark algorithms, the amount of placed data by it is 14 and 31 percent larger than the other two, respectively. The average cost for placing one unit of data by different algorithms are shown in Fig. 4d, from which it can be seen that the unit cost of placed data by algorithm `Appro-Alg` is around 5 and 9 percent cheaper than that of algorithms `Greedy-Alg` and `Random-Alg`. This means the proposed algorithm `Appro-Alg` places more data in a more economic way (lower cost for placing one unit of data).

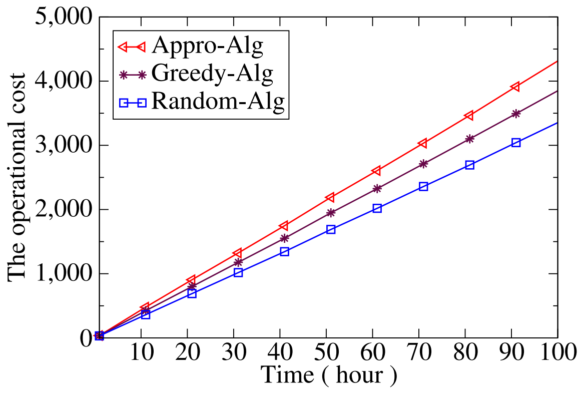
4.3 Impact of Parameters on Algorithm Performance

We then study the impact of the number of users in each group having datasets to place on the algorithm performance at each time slot, since not every one in a group at each time will have data to be placed. Assume that the number of users in a group is a value randomly drawn from $[5, 20]$, so different groups may have different numbers of users. We thus define a parameter θ that is a ratio of the number of users who have data to be placed to the number of users in a group, to evaluate the impact of these users having data to be placed on the algorithm performance, e.g., there are four users in group g_1 having their data to be placed at time slot 1, while the number of users in g_1 is 10, then $\theta = 0.4$. Fig. 5 plots the curves of system throughput and the operational costs by varying θ from 0.2 to 1.0. It can be seen from Fig. 5a that the system throughput drops from 96 to 65 percent with the growth of θ. The reason behind is that the larger θ implies that more users will place their data to the distributed cloud at this time slot, however the accumulated volume of placed data cannot exceed the processing capability of datacenters and the transmission capability of links in the distributed cloud. Fig. 5b depicts the curves of the operational costs of algorithm `Appro-Alg` by varying the value of θ, from which it can be seen that the operational cost increases with the growth of θ. Specifically, the operational cost increases from \$2,000 to \$8,000 when θ increases from 0.2 to 1.

We third evaluate the impact of the group size $|g|$ of each group g on the system throughput and operational cost of algorithm `Appro-Alg`, by varying $|g|$ from 10 to 30. Fig. 6a plots the system throughput curves, from which it can be seen that the system throughput decreases with the increase of the value of $|g|$. To be specific, the system throughput delivered by algorithm `Appro-Alg` is 98 percent when $|g| = 10$ and 50 percent while $|g| = 30$. Fig. 6b depicts the


(a) The system throughput of different algorithms


(b) The volume of placed data of different algorithms


(c) The operational cost of different algorithms

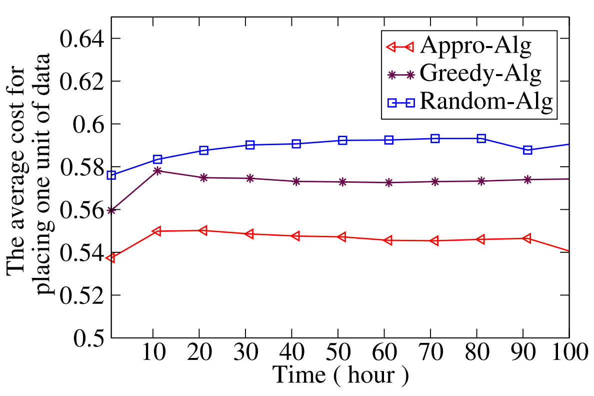
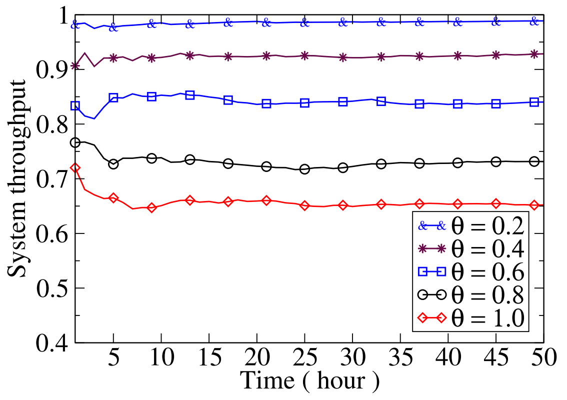

(d) The average cost for placing one unit of data by different algorithms

Fig. 4. The performance of algorithms `Appro-Alg`, `Greedy-Alg`, and `Random-Alg`, in terms of system throughput, the amounts of data placed, the operational cost, and the average cost for placing one unit of data.

operational cost of algorithm `Appro-Alg`. It can be seen that the operational cost initially increases from \$6,000 to \$10,000 when the group size varies from 10 to 20, and then drops a bit after $|g| = 20$, as the intermediate results generated by datasets will be multicast to more users in group g with the growth of the value of $|g|$. However, the available resources (both computing resource and bandwidth resource) are limited, the amounts of data that can be processed by datacenters and transmitted on Internet links are limited. With the increase of the group size in a multicast tree, the probability of sharing a common link among multicast sessions increases too, the system throughput therefore decreases with the increase of the group size $|g|$, and so is the operational cost.

We finally study the impact of the number of datacenters on the system throughput and the operational cost of the proposed algorithm `Appro-Alg`, by varying the number from 10 to 40. Fig. 7a plots the system throughput curves, from which it can be seen that the system throughput first grows very quickly from 91 to 97 percent when the number of datacenters increases from 10 to 20. The reason is that with the increase on the number of datacenters, more data


(a) The impact of θ on the system throughput

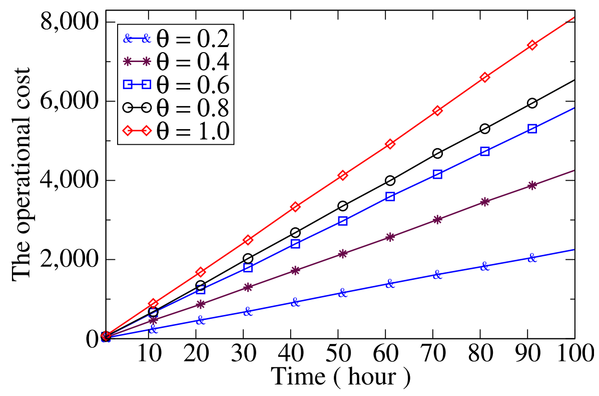
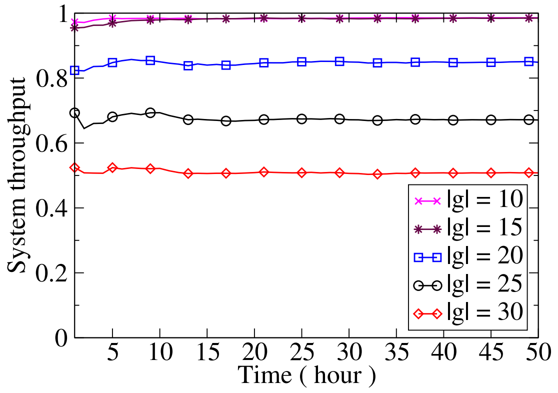
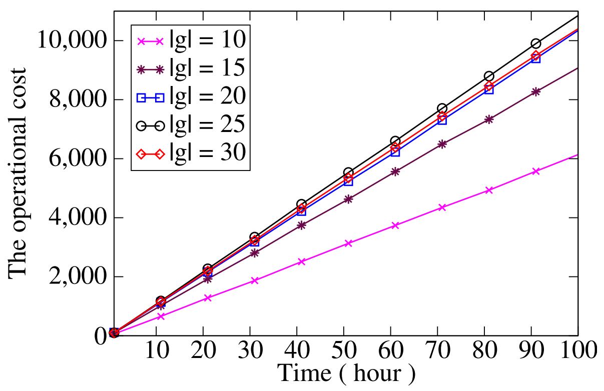

(b) The impact of θ on the operational cost

Fig. 5. Impact of θ on the performance of algorithm `Appro-Alg`, where θ is a ratio of the number of users who have data to be placed to the number of users in a group.



(a) The impact of group size $|g|$ on the system throughput



(b) The impact of group size $|g|$ on the operational cost

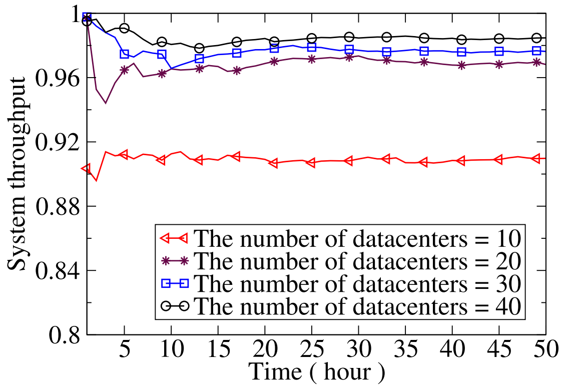
Fig. 6. Impacts of group size $|g|$ of each group on the performance of algorithm `Appro-Alg`.

can be placed and processed by the system. However, the bandwidth capacities of links now become the bottlenecks for data transmission within the system, the system throughput thus keeps stable when there are 20 datacenters in the distributed cloud. Fig. 7b depicts the operational cost curves, which grows with the increase on the number of datacenters, and keeps stable after there are 20 datacenters in the system, this is due to the fact that more cost will be incurred to manage more data.

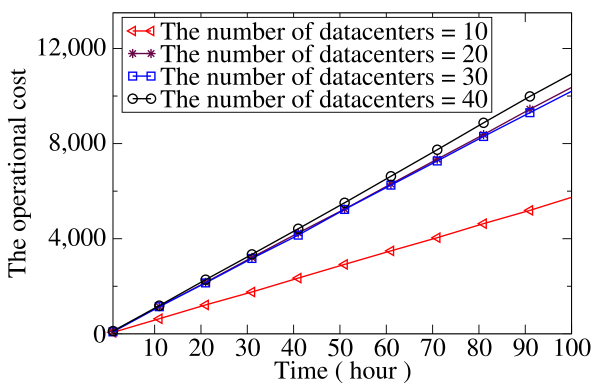
5 Related Work

Several studies on data placement in clouds have been conducted in the past [1], [5], [8], [14], [17], [26]. However, most of these studies did not consider the placement of dynamically generated big data [8], [17], [26], and focused only on the communication cost [8], [14]. Furthermore, they took neither fairness on resource allocations [1] nor the intermediate results of processed data into consideration [1], [17]. For example, Golab et al. [8] studied the problem of data placement to minimize the data communication cost for data-intensive tasks. Their goal is to determine where to store the data and where to evaluate the tasks in order to minimize the data communication cost. Jiao et al. [17] investigated multi-objective optimization for placing users data for socially aware services over multiple clouds, they aim to minimize the cost of updating and reading data, by exploring trade-offs among the multiple optimization objectives. They solve the problem by decomposing it into two subproblems: master replicas placement and slave replicas placement, and solving these two subproblems separately, thereby deriving a sub-optimal solution for the problem. Yuan et al. [26] provided an algorithm for data placement in scientific cloud workloads, which groups a set of datasets in multiple datacenters first, and then dynamically clusters newly generated datasets to the most appropriate data centers based on data dependencies. Liu and Datta [14] proposed a data placement strategy for scientific workflows by exploring data correlation, they aim to minimize the communication overhead incurred by data movement. Agarwal et al. [1] proposed an automated data placement mechanism Volley for geo-distributed cloud services, the objective is to minimize the user-perceived latency.

In this paper we studied the problem of big data management in the distributed cloud environments, consisting of placing data, processing data and transmitting the intermediate results of data processing to collaborative users located at different geographical locations, where big data are continuously generated at different locations, and the resource allocation fairness among collaborative users are incorporated, with the objective to maximize the system throughput while minimizing the operational cost of the cloud service provider, subject to the computing capacity of datacenters, the bandwidth capacity of links.



(a) The impact of the number of datacenters on the system throughput



(b) The impact of the number of datacenters on the operational cost

Fig. 7. Impacts of the number of datacenters on the performance of algorithm `Appro-Alg`.

6 Conclusion

In this paper, we considered a collaboration- and fairness-aware big data management problem in distributed cloud environments. We developed a novel optimization framework, under which we then devised a fast approximation algorithm for the problem. We also analyzed the time complexity and approximation ratio of the proposed algorithm. We finally conducted extensive experiments by simulations to evaluate the performance of the proposed algorithm. Experimental results demonstrated that the proposed algorithm is promising, and outperforms other two mentioned heuristics.

Acknowledgments

The authors appreciate the three anonymous referees for their constructive comments and valuable suggestions, which help them improve the quality and presentation of the paper.

References

[1] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, "Volley: Automated data placement for Geo-distributed cloud services," in *Proc. 7th USENIX Conf. Netw. Syst. Des. Implementation*, 2010, p. 2.

[2] Amazon EC2 [Online]. Available: http://aws.amazon.com/ec2/instance-types/, 2015.

[3] Amazon S3 [Online]. Available: http://aws.amazon.com/s3/, 2015.

[4] M. Alicherry and T. V. Lakshman, "Network aware resource allocation in distributed clouds," in *Proc. IEEE Conf. Comput. Commun.*, 2012, pp. 963–971.

[5] I. Baev, R. Rajaraman, and C. Swamy, "Approximation algorithms for data placement problems," *SIAM J. Comput.*, vol. 38, no. 4, pp. 1411–1429, 2008.

[6] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in *Proc. ACM. SIGCOMM, Conf.*, 2011, pp. 242–253.

[7] N. Garg and J. Könemann, "Faster and simpler algorithms for Multi-commodity flow and other fractional packing problems," in *Proc. IEEE 39th Annu. Symp. Found. Comput. Sci.*, 1998, pp. 300–309.

[8] L. Golab, M. Hadjieleftheriou, H. Karloff, and B. Saha, "Distributed data placement to minimize communication costs via graph partitioning," in *Proc. 26th Int. Conf. Sci. Statistical Database*, 2014, pp. 1–12.

[9] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *J. Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, 2009.

[10] [Online]. Available: http://www.cc.gatech.edu/projects/gtitm/, 2015.

[11] L. Gu, D. Zeng, P. Li, and S. Guo, "Cost minimization for big data processing in Geo-distributed data centers," *IEEE Trans. Emerging Topics Comput.*, vol. 2, no. 3, pp. 314–323, Sep. 2014.

[12] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, "SecondNet: A data center network virtualization architecture with bandwidth guarantees," in *Proc. 6th Int. Conf.*, 2010, pp. 1–12.

[13] H. V. Jagadish, Johannes Gehrke, A. Labrinidis, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, and C. Shahabi, "Big data and its technical challenges," *Commun. ACM*, Vol. 57, no. 7, pp. 86–94, 2014.

[14] X. Liu and A. Datta, "Towards intelligent data placement for scientific workflows in collaborative cloud environment," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. Workshops PhD Forum*, 2011, pp. 1052–1061.

[15] The Large Hadron Collider [Online]. Available: http://home.web.cern.ch/topics/large-hadron-collider, 2015.

[16] LOFAR [Online]. Available: http://www.lofar.org/, 2015.

[17] L. Jiao, Jun Li, W. Du, and X. Fu, "Multi-objective data placement for multi-cloud socially aware services," in *Proc. IEEE INFOCOM*, 2014, pp. 28–36.

[18] Y. Jadeja and K. Modi, "Cloud computing-concepts, architecture and challenges," in *Proc. Int. Conf. Comput. Electron. Elect. Technol.*, 2012, pp. 877–880.

[19] J. Webster, "Big Data-maximizing the flow," Technology insight paper, 2012.

[20] L. Popa, G. Kumar, M. Chowdhury, A. Krishnamurthy, S. Ratnasamy, and I. Stoica, "FairCloud: Sharing the network in cloud computing," in *Proc. ACM SIGCOMM*, 2012, pp. 1–6.

[21] S. Rao, R. Ramakrishnan, A. Silberstein, M. Ovsiannikov, and D. Reeves, "Sailfish: A framework for large scale data processing," in *Proc. ACM Symp. Cloud Comput.*, 2012, pp. 1–14.

[22] L.H. Sahasrabuddhe and B. Mukherjee, "Multicast routing algorithms and protocols: A tutorial," *J. IEEE Netw.*, vol. 14, no. 1, pp. 90–102, Jan./Feb. 2000.

[23] Z. Xu and W. Liang, "Minimizing the operational cost of data centers via geographical electricity price diversity," in *Proc. IEEE 6th Int. Conf. Cloud Comput.*, 2013, pp. 99–106.

[24] Z. Xu and W. Liang, "Operational cost minimization of distributed data centers through the provision of fair request rate allocations while meeting different user SLAs," *Comput. Netw.*, vol. 83, pp. 59–75, 2015.

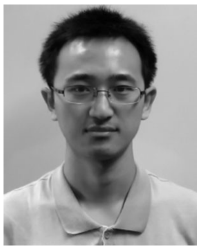
[25] Q. Xia, W. Liang and Z. Xu, "Data locality-aware query evaluation for big data analytics in distributed clouds," in *Proc. 2nd Int. Conf. Adv. Cloud Big Data*, 2014, pp. 1–8.

[26] D. Yuan, Y. Yang, X. Liu, and J. Chen, "A data placement strategy in scientific cloud workflows," *J. Future Gener. Comput. Syst.*, vol. 26, no, 8, pp. 1200–1214, 2010.

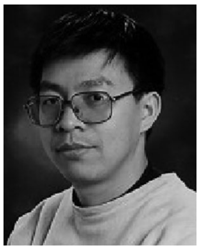
[27] L. Zhang, Z. Li, C. Wu, and M. Chen, "Online algorithms for uploading deferrable big data to the cloud," in *Proc. IEEE INFOCOM*, 2014, pp. 2022–2030.

[28] L. Zhang, C. Wu, Z. Li, C. Guo, M. Chen, and F.C.M. Lau, "Moving big data to the cloud: An online cost-minimizing approach," *IEEE J. Sel. Areas. Commun.*, vol. 31, no. 12, pp. 2710–2721, Dec. 2013.

Qiufen Xia received the BSc and ME degrees from Dalian University of Technology in China in 2012 and 2009, both in computer science. She is currently working toward the PhD degree in the Research School of Computer Science, Australian National University. Her research interests include mobile cloud computing, distributed clouds, cloud computing, and optimization problems. She is a student member of the IEEE.



Zichuan Xu received the BSc and ME degrees from Dalian University of Technology in China in 2011 and 2008, both in computer science. He is currently working toward the PhD degree in the Research School of Computer Science, Australian National University. His research interests include distributed clouds, data center networks, cloud computing, algorithmic game theory, and optimization problems. He is a student member of the IEEE.



Weifa Liang (M'99-SM'01) received the BSc degree from Wuhan University, China, in 1984, the ME degree from the University of Science and Technology of China in 1989, and the PhD degree from the Australian National University in 1998, all in computer science. He is currently an associate professor in the Research School of Computer Science, Australian National University. His research interests include design and analysis of routing protocols for wireless ad hoc and sensor networks, cloud computing, graph databases, design and analysis of parallel and distributed algorithms, combinatorial optimization, and graph theory. He is a senior member of the IEEE.

Albert Y. Zomaya (M'90-SM'97-F'04) is currently the chair professor of high-performance computing and networking with the School of Information Technologies, University of Sydney, Sydney, NSW, Australia, where he is also the director in the Centre for Distributed and High Performance Computing, which was established in 2009. He has authored more than 500 scientific papers and articles, and has authored, coauthored, or edited over 20 books. He serves as an associate editor of 22 leading journals, such as, the *ACM Computing Surveys* and *Journal of Parallel and Distributed Computing*. He received the IEEE Technical Committee on Parallel Processing Outstanding Service Award (2011), the IEEE Technical Committee on Scalable Computing Medal for Excellence in Scalable Computing (2011), and the IEEE Computer Society Technical Achievement Award (2014). He is a chartered engineer, and a fellow of the American Association for the Advancement of Science and the Institution of Engineering and Technology (United Kingdom). His research interests are in the areas of algorithms, parallel and distributed computing, and complex systems. He is a fellow of the IEEE.