QoS-Aware Proactive Data Replication for Big Data Analytics in Edge Clouds

Qiufen Xia
qiufenxia@dlut.edu.cn
Dalian University of Technology
Dalian, Liaoning, China

Luyao Bai
bailuyao1997@outlook.com
Dalian University of Technology
Dalian , Liaoning, China

Weifa Liang
wliang@cs.anu.edu.au
Australian National University
Canberra, ACT, Australia

Zichuan Xu
z.xu@dlut.edu.cn
Dalian University of Technology
Dalian, Liaoning, China

Lin Yao
yaolin@dlut.edu.cn
Dalian University of Technology
Dalian, Liaoning, China

Lei Wang
lei.wang@dlut.edu.cn
Dalian University of Technology
Dalian, Liaoning, China

ABSTRACT

We are in the era of big data and cloud computing, large quantity of computing resource is desperately needed to detect invaluable information hidden in the coarse big data through query evaluation. Users demand big data analytic services with various Quality of Service (QoS) requirements. However, cloud computing is facing new challenges in meeting stringent QoS requirements of users due to the remoteness from its users. Edge computing has emerged as a new paradigm to address such shortcomings by bringing cloud services to the edge of the operation network in proximity of users for performance improvement. To satisfy the QoS requirements of users for big data analytics in edge computing, the data replication and placement problem must be properly dealt with such that user requests can be efficiently and promptly responded. In this paper, we consider data replication and placement for big data analytic query evaluation. We first cast a novel proactive data replication and placement problem of big data analytics in a two-tier edge cloud environment, we then devise an approximation algorithm with an approximation ratio for it, we finally evaluate the proposed algorithm against existing benchmarks, using both simulation and experiment in a testbed based on real datasets, the evaluation results show that the proposed algorithm is promising.

KEYWORDS

Data replication and placement; big data analytics; edge clouds; query evaluation

ACM Reference Format:
Qiufen Xia, Luyao Bai, Weifa Liang, Zichuan Xu, Lin Yao, and Lei Wang. 2019. QoS-Aware Proactive Data Replication for Big Data Analytics in Edge Clouds. In *48th International Conference on Parallel Processing: Workshops (ICPP 2019), August 5–8, 2019, Kyoto, Japan.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3339186.3339207

ICPP 2019, August 5–8, 2019, Kyoto, Japan

ACM ISBN 978-1-4503-7196-4/19/08…$15.00
https://doi.org/10.1145/3339186.3339207

1 INTRODUCTION

Cloud platforms have been receiving ever-growing attentions in recent years to provide services in a wide range of information technology (IT) domains, and offer on-demand processing, storage and bandwidth resources. Many services have been deployed on clouds and generate big data there, the big data are analyzed to obtain hidden valuable information for business advantages and decision-makings. However, cloud computing is facing new challenges in meeting the quality of service (QoS) requirements of emerging applications, such as augmented reality, autonomous vehicles, timely query evaluation for big data analytics, to name a few. We argue that the most pressing requirement of those emerging applications is response latency, which is the time duration from submitting a request to the cloud to receiving the query result by the request user. The remote cloud data centers are not appropriate for achieving small response latencies, as it could suffer from limitations due to high transmission latency and risk of heavy workload as well as network bottlenecks.

One promising solution to tackle the mentioned challenges is Edge Computing, which can exploit processing and storage capabilities at the edge of the network as near as possible to end-users. In this regard, the deployment of edge cloudlets in network access points can achieve remarkable benefits in terms of low-latency interactions and economic computing resource. Query evaluation for big data analytics demands large quantity of computing resource and low response latency, by leveraging edge computing technologies, the response time to big data analytics queries can be significantly reduced. To this end, an important approach is to proactively replicate a large dataset to multiple data centers or cloudlets so that query users can obtain their desired query results within their specified time duration.

Although data replication and placement can improve system performance, it does not necessarily imply that more replicas will lead to better system performance, due to the fact that the maintenance of data consistency between the original dataset and its slave replicas in the network does incur cost. To maximize the benefit of query processing and dataset replications, strategic replicating and placing replicas of each dataset in a two-tier edge cloud is crucial. One fundamental problem thus is how to place the replicas of datasets to different data centers or cloudlets in the two-tier edge cloud so that big data analytics queries can be evaluated, the

volume of datasets demanded by admitted queries is maximized, without violating the resource capacity constraints and delay requirements of users. Notice that, one main reason that we aim to maximize the volume of datasets demanded by admitted queries is as follows. Cloud service providers such as Amazon offer users a *pay-as-you-go* approach for pricing [3], maximizing the volume of datasets demanded by admitted queries means that users pay more for evaluating queries to the cloud service providers who can thus obtain maximum income.

Several studies on data replication and placement have been conducted in the past [1, 6, 26]. However, most these studies considered neither data replications of the generated big data [1, 26] nor QoS requirements of users [1, 6, 26]. In addition, there are several investigations on query evaluation and data placement [17, 20]. Although some of them considered the data transmission cost, they did not incorporate the QoS requirements of users [17], or data replications and placements [20]. In this paper, we study proactive data replication and placement of query evaluation for big data analytics in a two-tier edge cloud with the aim to maximize the volume of datasets demanded by admitted queries while meeting users' QoS requirements, subject to various resource capacities on an edge cloud network.

The main contributions of this paper are as follows.

- We first formulate a novel proactive QoS-aware data replication and placement problem for big data analytic query evaluation in a two-tier edge cloud environment. We aim to maximize the volume of datasets demanded by admitted queries while meeting users' end-to-end delay requirements.
- We then propose an efficient approximation algorithm with provable approximation ratio for the problem through a primal-dual dynamic update technique.
- We finally evaluate the performance of the proposed algorithm through experimental simulations and in a testbed using real datasets. The simulation results show that the performance of the proposed algorithm is promising, placing significantly higher volume of datasets demanded by queries admitted compared to some existing work.
- To the best of our knowledge, this is the first time that the proactive QoS-aware data replication and placement problem for big data analytics query evaluation in two-tier edge clouds is considered, and an efficient approximation algorithm is devised.

The remainder of this paper is organized as follows. Section 2 introduces the system model and problem definition, followed by an approximation algorithm for the problem in Section 3. The performance evaluation of the proposed algorithm is conducted in Section 4. The related work is presented in Section 5, and conclusions are given in Section 6.

2 PRELIMINARIES

In this section, we first introduce the system model. We then give notations on big data analytics evaluation in the two-tier edge cloud under QoS requirements of users. We finally define the problem precisely.

2.1 System model

We consider a two-tier edge cloud $G = (\mathcal{BS} \cup \mathcal{SW} \cup \mathcal{CL} \cup \mathcal{DC}, E)$, which consists of a set $\mathcal{BS}$ of base stations through which users connect to edge cloudlets, a set $\mathcal{SW}$ of switches in a Wireless Metropolitan Area Network (WMAN), a set $\mathcal{CL}$ of edge cloudlets co-located with some switches in $\mathcal{SW}$, and a set $\mathcal{DC}$ of data centers located at different geographical locations that are connected to the WMAN via the Internet to/from gateway nodes in $\mathcal{SW}$.

These edge cloudlets, switches (or access points), and data centers are inter-connected by a set E of communication links, and $e \in E$ is a link between two cloudlets, two switches, a cloudlet and a switch, or a gateway node and a data center.

Let CL_i be an edge cloudlet in $\mathcal{CL}$, and DC_j be a data center in $\mathcal{DC}$. The computing resource of each edge cloudlet CL_i and each data center DC_j can be used for processing data to evaluate queries, while their storage resource is used to store the query results and data replicas. The quantity of available computing resource of each data center or edge cloudlet is limited, especially for cloudlets which usually consist of several servers to fit into small machine rooms located in metropolitan areas. Denote by $B(DC_j)$ and $B(CL_i)$ the computing capacities of data center DC_j and cloudlet CL_i, respectively. Denote by $A(CL_i)$ and $A(DC_j)$ the available computing resources of edge cloudlet CL_i and data center DC_j at the moment. Evaluating queries of big data analytics in edge cloudlets and data centers consumes their computing resources. Let r_m be the amount of computing resource allocated to process a unit data. We do not restrict the capacity of storage resource of cloudlets and data centers, as the storage resource usually is abundant and inexpensive, compared with the expensive computing resource [26].

The processing and transmission of data in G consume computing and bandwidth resources of edge clouds and thus incur processing and transmission delays. Let $d(CL_i)$ and $d(DC_j)$ be the delays incurred by processing a unit data per unit computing resource in cloudlet CL_i and data center DC_j and $d_t(e)$ the transmission delay on link $e \in E$ for transferring a unit data.

For simplicity, let $V = \{\mathcal{CL} \cup \mathcal{DC}\}$, and each node $v_l \in V$ represents either an edge cloudlet or a data center. An example of a two-tiered edge cloud G is illustrated in Fig. 1.

2.2 Big data processing in the edge cloud

With the wide adoption of cloud services, enterprise users usually have large scale of legacy services being outsourced to remote data centers, and these services generate large volume of data from their outsourced services, such as web logs, click streams, sensory data. Meanwhile, with the support of network service providers, more and more cloud services are deployed in edge cloudlets within the proximity of users to reduce the response time. To obtain valuable information and interesting patterns from such big data generated by services deployed at data centers and cloudlets, users may conduct analysis on big data that are stored in remote data centers and edge cloudlets by issuing queries.

Performing big data analytics in remote data centers causes very high latency, because large volume of intermediate results generated by processing the big data need to be transferred to edge cloudlets and join with the intermediate results there, the

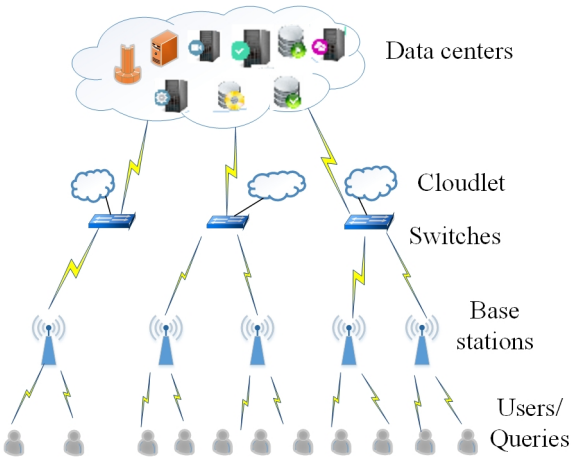


Figure 1: An example of a two-tier edge cloud G.

delay requirements required by users may be violated ultimately. Therefore, proactively replicating big data from the remote cloud to edge cloudlets is an effective way to reduce data transmission delay and guarantee the timeliness of big data analytics. Meanwhile, the computing capacity of an edge cloudlet is very limited, it takes long time to evaluate queries, sometimes the computing resource of an edge-cloudlet even cannot satisfy the resource demands of big data query evaluation, so the big data generated in edge cloudlets can be proactively placed to the remote data centers and processed there, thereby reducing the processing delay to guarantee the timeline of queries and satisfying the computing resource requirements of queries. We thus assume that the big data and their replicas can be replicated to the edge cloudlets or remote data centers in advance, such that the delay incurred by the joint analysis of datasets or transmission of intermediate results is no greater than the delay requirements of queries.

Let $\mathcal{S}$ be the collection of datasets generated by all services in remote data centers, denote by S_n a dataset in $\mathcal{S}$, where $1 \leq n \leq |\mathcal{S}|$ with $|\mathcal{S}|$ representing the number of datasets in $\mathcal{S}$. Denote by q_m a query for big data analytics. Each query q_m usually requires to be evaluated based on a collection of datasets. Let $\mathcal{S}(q_m)$ be the collection of datasets required by query q_m.

Evaluating a query q_m is to abstract the intermediate results from its requested datasets that possibly are in different data centers or cloudlets, and aggregate the intermediate results at the home location of the query. Let h_m be the home location of query q_m, which can be a data center or a cloudlet. Without loss of generality, we assume that the size of an intermediate result on each dataset S_n evaluated by query q_m is a fraction size α_{nm} of S_n, i.e., $\alpha_{nm} \cdot |S_n|$, where α_{nm} is with $0 < \alpha_{nm} \leq 1$ [21] and $|S_n|$ is the volume of dataset S_n.

2.3 User QoS requirements

As we consider query evaluation for big data analytics within stringent delay requirements, we refer to the *delay requirement* of a query as its *quality of service (QoS) requirement*, where the delay experienced by the query is defined as the duration from the query is issued to the evaluation result is received. Since the size of a query is usually small, the transfer delay of the query from a user location to the edge cloud network is negligible.

Each query may require multiple datasets or datasets' replicas placed at different locations, the processing datasets and transmitting intermediate results can be performed in parallel among different datasets, therefore the delay experienced by q_m demanding multiple datasets is the maximum sum of the delays incurred in processing a dataset and transmitting the intermediate results of the dataset in $\mathcal{S}(q_m)$ accessed by q_m, i.e., $argmax\{(d(v_l) \cdot |S_n| + d_t(p_{v_l,h_m}) \cdot |S_n| \cdot \alpha_{nm})\}$. Denote by d_{q_m} the maximum tolerable delay of query q_m, that is to say, d_{q_m} is the QoS in terms of delay requirement of query q_m. To make datasets in the two-tier edge cloud highly available, reliable and scalable, the datasets usually have several replicas, while in order to reduce the cost for data consistency, we thus assume that each dataset S_n has at most K replicas in the system with $K \in \mathbb{Z}^+$, the replication of datasets to data centers or cloudlets are conducted in advance before the evaluation of queries, and the delay incurred by dataset replications is not accounted into the QoS requirement of queries.

2.4 Problem definition

Given a collection $\mathcal{S}$ of datasets, a set of queries $Q = \{q_m \mid 1 \leq m \leq M\}$ for big data analytics, and a two-tier edge cloud network $G = (\mathcal{BS} \cup \mathcal{SW} \cup V, E)$, where $V = \mathcal{CL} \cup \mathcal{DC}$, the computing resource of each node $v_l \in V$ is capacitated. Different queries have different QoS requirements.

The *proactive data replication and placement problem for query evaluation of big data analytics* in the two-tiered edge cloud network G is to place at most K replicas for each dataset $S_n \in \mathcal{S}$ to cloudlets or data centers in advance such that the volume of placed datasets demanded by admitted queries is maximized while meeting the delay requirements of all admitted queries, subject to the computing resource capacities on edge cloudlets and data centers, where K is a given small integer with $K \geq 1$.

Here, a query is *admitted* if the QoS requirement of the query can be satisfied and the computing capacity of each cloudlet and data center is not violated, the admitted queries will be evaluated by the cloudlets or data centers. Notice that, we here only consider the proactive replication and placement for static data, as for the dynamic aspect of data, we set a threshold, which is a ratio of the volume of new generated data to the volume of original data at a time point. When the ratio of the volume of new generated data achieves the threshold, an update operation is made between the original data and its replicas to keep data consistent in the whole network.

3 AN APPROXIMATION ALGORITHM FOR PROACTIVE DATA REPLICATION AND PLACEMENT

In this section, we first give an overview of the proposed algorithm, we then formulate an Integer Linear Programming (ILP) solution to the proactive data replication and placement problem for query evaluation of big data analytics, and devise an approximation algorithm with an approximation ratio by the primal-dual dynamic-update technique, we finally analyze the correctness and time complexity of the approximation algorithm.

3.1 Algorithm overview

In the proactive data replication and placement, each query can demand several datasets each time, and a dataset can be demanded by multiple different queries at each time. It is NP-hard [5] to find an optimal solution for the problem. However, for a special case where each query demands only one dataset, there is an approximation algorithm based on the primal-dual dynamic-update technique. Therefore for a general case where each query demands multiple datasets, we can also get an approximation algorithm by invoking the proposed approximation algorithm in the special case.

3.2 Integer linear programming

We formulate the problem as an integer linear programming (ILP). We first define a set of decision variables. Recall that, in the problem, there are a set Q of queries and a collection $\mathcal{S}$ of datasets, these queries demand datasets for evaluation with different delay requirements, some replicas of the datasets should be created and placed at appropriate locations in G, such that the volume of dataset replicas demanded by admitted queries is maximized while satisfying the delay requirements of the queries, subject to the capacity constraints on data centers and edge cloudlets. As maintaining data consistency between an original dataset $S_n \in \mathcal{S}$ and its replicas incurs cost, we assume that each dataset has at most K replicas in the edge cloud. Therefore, the proactive data replication and placement problem is equivalent to determining where the replicas of each dataset should be proactively placed, and which queries should be assigned to which data centers or edge cloudlets for evaluation. Recall that $V = CL \cup \mathcal{DC}$ is the set of edge cloudlets and data centers in G, each location node $v_l \in V$ is either an edge cloudlet or a data center, $1 \le l \le |CL \cup \mathcal{DC}|$. We thus use a binary decision variable x_{nl} indicating whether a replica of dataset S_n is placed at a location node v_l in G. Similarly, we use a binary variable π_{ml} to indicate whether a query q_m is assigned to a location node v_l to access the replica of dataset $S_n \in \mathcal{S}(q_m)$. Once a query q_m is assigned to a location node v_l where the replicas demanded by q_m are placed, the processed intermediate results will be transferred to the home location node h_m of q_m, via a shortest path whose transmission delay is the minimum one.

We then formulate the objective of the proactive data replication and placement problem, which is to maximize the volume of datasets demanded by admitted queries that can be expressed by

$$maximize \sum_{q_m \in Q} \sum_{v_l \in V} |S_{q_m}| \cdot \pi_{ml} \tag{1}$$

subject to the following constraints,

$$\sum_{q_m \in Q} |S_{q_m}| \cdot r_m \cdot \pi_{ml} \le A(v_l), \forall\, v_l \in V \tag{2}$$

$$\pi_{ml} - x_{q_m l} \le 0,\ \forall\, q_m \in Q \text{ and } \forall\, v_l \in V \tag{3}$$

$$|S_{q_m}| \cdot [d(v_l) + d_t(p_{v_l,h_m}) \cdot \alpha_{q_m}] \cdot \pi_{ml} \le d_{q_m},\ \forall\, q_m \in Q, \forall\, v_l \in V \tag{4}$$

$$\sum_{v_l \in V} x_{nl} \le K,\ \forall\, S_{q_m} \in \mathcal{S} \tag{5}$$

$$\pi_{ml} \in \{0,1\}, \tag{6}$$

$$x_{nl} \in \{0,1\}, \tag{7}$$

where Constraint (2) ensures that the computing resource of node v_l allocated to evaluate queries that demand dataset S_{q_m} is no greater than the available computing resource of v_l. Constraint (3) ensures that only when the dataset S_n required by query q_m is placed at node v_l, query q_m can then be assigned to v_l. Constraint (4) guarantees that the delay requirement d_{q_m} of each query q_m is met. Constraint (5) ensures that each dataset has at most K replicas in G.

3.3 An approximation algorithm

We consider the above ILP for the proactive data replication and placement problem as the Primal problem. We first calculate the Dual of the Primal, we then devise an approximation algorithm for the Dual problem. To be specific, we define four dual variables θ_l, y_{ml}, η_{ml} and μ_{q_m}, then the dual of the Primal problem can be formulated as

$$min \sum_{v_l \in V} A(v_l) \cdot \theta_l + \sum_{q_m \in Q} \sum_{v_l \in V} d_{q_m} \cdot \eta_{ml} + \sum_{q_m \in Q} K \cdot \mu_{q_m} \tag{8}$$

subject to the following constraints,

$$|S_{q_m}| \cdot r_m \cdot \theta_l + y_{ml} + |S_{q_m}| \cdot [d(v_l) + d_t(p_{v_l,h_m}) \cdot \alpha_{q_m}] \cdot \eta_{nl} \ge |S_{q_m}|,\ \forall\, q_m \in Q \text{ and } \forall\, v_l \in V \tag{9}$$

$$\sum_{S_{q_m} \in \mathcal{S}} \mu_{q_m} - \sum_{q_m \in Q} y_{ml} \ge 0,\ \forall\, v_l \in V \tag{10}$$

$$\theta_l \ge 0, \tag{11}$$

$$y_{ml} \ge 0, \tag{12}$$

$$\eta_{ml} \ge 0, \tag{13}$$

$$\mu_{q_m} \ge 0. \tag{14}$$

The primal complementary slackness conditions are as follows:

- For each query $q_m \in Q$ and each node $v_l \in V$, if $\pi_{ml} > 0$ then

$$|S_{q_m}| \cdot r_m \cdot \theta_l + y_{ml} + |S_{q_m}| \cdot [d(v_l) + d_t(p_{v_l,h_m}) \cdot \alpha_{q_m}] \cdot \eta_{nl} = |S_{q_m}| \tag{15}$$

- For each node v_l, if $x_{q_m l} > 0$ then

$$\sum_{S_{q_m} \in \mathcal{S}} \mu_{q_m} - \sum_{q_m \in Q} y_{ml} = 0 \tag{16}$$

We apply the complementary slackness approach to the approximation algorithm by defining **relaxed complementary slackness**. The relaxed primal complementary slackness conditions are as follows:

- For each query $q_m \in Q$ and each node $v_l \in V$, if $\pi_{ml} > 0$ then

$$|S_{q_m}| \le |S_{q_m}| \cdot r_m \cdot \theta_l + y_{ml} + |S_{q_m}| \cdot [d(v_l) + d_t(p_{v_l,h_m}) \cdot \alpha_{q_m}] \cdot \eta_{nl} \le \beta \cdot |S_{q_m}| \tag{17}$$

- For each node v_l, if $x_{q_m l} > 0$ then

$$\sum_{S_{q_m} \in \mathcal{S}} \mu_{q_m} - \sum_{q_m \in Q} y_{ml} = 0 \tag{18}$$

Algorithm 1 *An approximation algorithm* Appro-S *for the proactive data placement problem where a query demands only one single dataset each time.*

Input: The set Q of queries, the set $\mathcal{S}$ of datasets, the set V of nodes in the two-tier edge cloud.
Output: The maximum volume of datasets demanded by admitted queries.

$Q' \leftarrow \emptyset$ // set of admitted queries ;
$V' \leftarrow \emptyset$ //set of nodes where the replicas of the datasets are placed;
$\mathcal{S}' \leftarrow \emptyset$ // set of placed replicas;
$\theta \leftarrow 0$, $y \leftarrow 0$,$\eta \leftarrow 0$,$\mu \leftarrow 0$;
$N \leftarrow 0$ //the volume of datasets demanded by admitted queries;
while each $\mu_{q_m} \leq K$ or $Q' \neq Q$ **do**
  Uniformly increase μ_{q_m} by 1 in a unit time, that is we create one replica of dataset S_{q_m} demanded by q_m;
  Increase y_{ml} uniformly, i.e., increase y_{ml} by 1 in a unit time. After a while, Eq. 10 becomes tight, that is $\sum_{S_{q_m} \in \mathcal{S}} \mu_{q_m} - \sum_{q_m \in Q} y_{ml} = 0$, i.e., $\mu_{q_m} - y_{ml} = 0$ as each query q_m only demands one single dataset S_{q_m} each time;
  Increase uniformly all θ_l and η_{nl} simultaneously, that is in a unit time θ_l and η_{nl} increase 1. After a while Eq. 9 becomes tight, it means we have $y_{ml} = |S_{q_m}| - |S_{q_m}| \cdot r_m \cdot \theta_l - |S_{q_m}| \cdot [d(v_l) + d_t(p_{v_l,h_m}) \cdot \alpha_{q_m}] \cdot \eta_{nl}$;
  Add q_m into Q', remove q_m from Q, that is $Q' \leftarrow Q' \cup \{q_m\}$, and $Q \leftarrow Q \setminus \{q_m\}$;
  Add S_{q_m} into $\mathcal{S}'$, i.e., $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{S_{q_m}\}$;
  Add v_l into V', i.e., $V' \leftarrow V' \cup \{v_l\}$;
  Declare that query q_m and S_{q_m} are assigned to node v_l;
  $N \leftarrow N + |S_{q_m}|$;
Return N.

Observations: From the dual we can observe the meanings of dual variables: θ_l means the computing cost for evaluating q_m on node v_l; y_{ml} represents the cost by assigning q_m to v_l; η_{ml} is the cost for satisfying the delay requirement of q_m, if q_m is assigned to node v_l; μ_{q_m} is the cost for creating a replica of a dataset demanded by query q_m. Based on the observations, for a special case where a query demands only one single dataset, we can devise an approximation algorithm that calculates the placed datasets $\mathcal{S}'$ and admitted queries Q', and satisfies the delay requirements of queries. For simplicity, we refer algorithm 1 as Appro-S.

Notice that approximation algorithm Appro-S works in a special case where each query demands only one single dataset. In contrast, for a general case where each query demands multiple datasets each time, we can still use algorithm Appro-S to derive another approximation algorithm, that is once a query demands a dataset we invoke algorithm Appro-S. The specific algorithm is detailed in algorithm 2, which is referred as Appro-G.

Theorem 1. *The approximation algorithm* Appro-S *gives an approximation ratio argmax*$(|Q|, |V|/K)$*, the approximation algorithm* Appro-G *gives an approximation ratio argmax*$(|Q| \cdot |\mathcal{S}|, |V| \cdot |\mathcal{S}|/K)$*, where* $|Q|$ *is the number of queries in the system,* $|V|$ *is the number of cloudlets and data centers,* $|\mathcal{S}|$ *is the number of datasets, and* K *is the maximum number of replicas of each dataset.*

Algorithm 2 *An approximation algorithm* Appro-G *for the proactive data placement problem where a query demands multiple datasets each time.*

Input: The set Q of queries, the set $\mathcal{S}$ of datasets, the set V of nodes in the two-tier edge cloud.
Output: The maximum volume of datasets demanded by admitted queries.

$N' \leftarrow 0$ // the total volume of datasets demanded by admitted queries;
$N \leftarrow 0$ // the volume of datasets demanded by admitted queries in Appro-S;
for each $q_m \in Q$ and each dataset $S_n \in \mathcal{S}(q_m)$ **do**
  Invoke algorithm 1 ;
  $N' \leftarrow N' + N$;
Return N'.

Proof. Let θ, y, η and μ be the returned dual-feasible solution. To prove the approximation ratio, we need to compare the maximum volume of datasets demanded by admitted queries of the approximated solution, which is $\sum_{q_m \in Q} \sum_{v_l \in V} |S_{q_m}|$, to the cost of the dual feasible solution (θ, y, η and μ), which is $\sum_{v_l \in V} A(v_l) \cdot \theta_l + \sum_{q_m \in Q} \sum_{v_l \in V} d_{q_m} \cdot \eta_{ml} + \sum_{q_m \in Q} K \cdot \mu_{q_m}$.

As described in algorithm Appro-S, after some recurrence, we have q_m and v_l which make $\mu_{q_m} - y_{ml} = 0$, based on formula (10) we have

$$|S_{q_m}| \leq \big(|S_{q_m}| \cdot r_m \cdot \theta_l + \mu_{q_m} + |S_{q_m}| \cdot [d(v_l) + d_t(p_{v_l,h_m}) \cdot \alpha_{q_m}] \cdot \eta_{nl}\big), \tag{19}$$

then we have

$$\begin{aligned}
&\sum_{q_m \in Q} \sum_{v_l \in V} |S_{q_m}| \leq \\
&\sum_{q_m \in Q} \sum_{v_l \in V} |S_{q_m}| \cdot r_m \cdot \theta_l + \sum_{q_m \in Q} \sum_{v_l \in V} \mu_{q_m} \\
&+ \sum_{q_m \in Q} \sum_{v_l \in V} |S_{q_m}| \cdot [d(v_l) + d_t(p_{v_l,h_m}) \cdot \alpha_{q_m}] \cdot \eta_{nl} \\
&= \sum_{v_l \in V} A(v_l) \cdot \theta_l \cdot \frac{\sum_{q_m \in Q} |S_{q_m}| \cdot r_m}{A(v_l)} + \sum_{q_m \in Q} \\
&\sum_{v_l \in V} d_{q_m} \cdot \eta_{nl} \cdot \frac{|S_{q_m}| \cdot [d(v_l) + d_t(p_{v_l,h_m}) \cdot \alpha_{q_m}]}{d_{q_m}} \\
&+ \sum_{q_m \in Q} K \cdot \mu_{q_m} \cdot \frac{|V|}{K} \\
&\leq \sum_{v_l \in V} A(v_l) \cdot |Q| + \sum_{q_m \in Q} \sum_{v_l \in V} d_{q_m} \cdot \eta_{nl} + \\
&\sum_{q_m \in Q} K \cdot \mu_{q_m} \cdot \frac{|V|}{K}
\end{aligned} \tag{20}$$

Therefore, the approximation ratio of algorithm Appro-S is $\max\{|Q|, \frac{|V|}{K}\}$. As each query can maximumly demand $|\mathcal{S}|$ datasets at each time,

so the approximation ratio of algorithm Appro-G is $argmax(|Q| \cdot |\mathcal{S}|, |V| \cdot |\mathcal{S}|/K)$. □

4 PERFORMANCE EVALUATION

In this section, we evaluate the performance of the proposed algorithms Appro-S and Appro-G, and investigate the impact of important parameters on the algorithmic performance, by both simulations and a proof-of-concept in a real test-bed using real datasets.

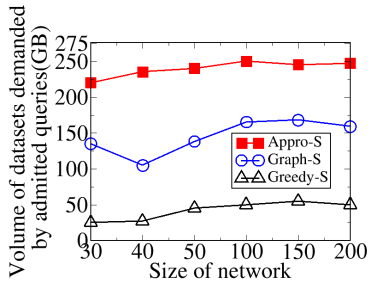
4.1 Experimental environment

For simulation, we consider a two-tier edge cloud consisting of 6 data centers, 24 cloudlets and 2 switches, there is a link between each pair of nodes (data centers, cloudlet, and switches) with a probability of 0.2, generated by the GT-ITM tool [8]. The delay The computing capacities of each data center and cloudlet are randomly drawn from a value interval [200, 700] and [8, 16] units (GHz) [26] respectively. Each user produces several Gigabytes of data, we thus emulate the volume of the dataset generated by each user is in the range of [1, 6] GB [26], and the amount of computing resource assigned to the processing of 1GB data is a value in the range of [0.75, 1.25] GHz [2, 4]. The numbers of datasets and queries in the system are randomly drawn in the range of [5, 20] and [10, 100], respectively. The number of datasets required by the query is randomly drawn from interval [1, 7]. Taking the transfer delay in real cables into consideration, the QoS in terms of delay requirement of each query depends on the size of dataset demanded by the query, the reason is to avoid some users who demand more dataset require the same delay as users who demand few dataset. Unless otherwise specified, we will adopt the default settings in our experiments. Each value in the figures is the mean of the results by applying each mentioned algorithm on 15 different topologies of the two-tier edge cloud.

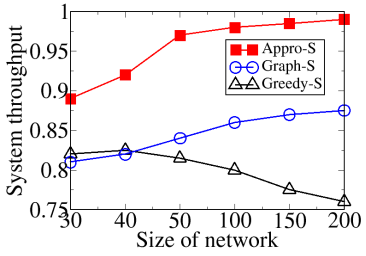
We evaluate the performance of the proposed algorithms against two benchmarks. The first benchmark adopts a greedy strategy, it selects a data center or cloudlet with largest available computing resource to place a replica of a dataset. If the delay requirement cannot be satisfied, it then selects a data center or a cloudlet with the second largest available computing resource to place the replica. This procedure continues until the query is admitted or there are already K replicas of the dataset in the system. Another benchmark is from an existing work [10] that places K replicas for each dataset at data centers or cloudlets, if the delay requirement of the query can be satisfied by evaluating the replica at the data center or the cloudlet. This procedure continues until the query is admitted or there are already K replicas of the dataset in the system. It then makes a graph partitioning with maximum volume of datasets demanded by admitted queries. For simplicity, we refer to the two benchmarks as Greedy-S and Graph-S for the special case where each query only demands a single dataset, while for the general case where each query demands multiple datasets we refer to them as Greedy-G and Graph-G, respectively.

In addition, we also evaluate the proposed algorithms in a real testbed. For which, we leased a number of virtual machines from a cloud service provider DigitalOcean [9]. These virtual machines are located at geo-distributed locations. A two-tier edge cloud is deployed by making use of both the leased virtual machines and local servers, based on which we evaluate the proposed algorithms against an existing work [13]. The benchmark work first calculates the popularity of a node (cloudlet and data center) according to the ratio of the number of dataset replicas on the node to the total number of dataset replicas of all nodes. It then selects a node with the highest popularity for each dataset, and places a replica of the dataset if the delay requirement of a query can be satisfied; otherwise, if then selects another node with the second highest popularity to place the replica; this procedure continues until the query is admitted or there are already K replicas of the dataset. As we consider the special case where each query only demands one single dataset and a general case where each query demands multiple datasets, we thus refer to the benchmark as algorithm Popularity-S for the special case and Popularity-G for the general case for simplicity, respectively.

4.2 Performance evaluation of different algorithms by simulations



(a) The volume of datasets demanded by admitted queries.

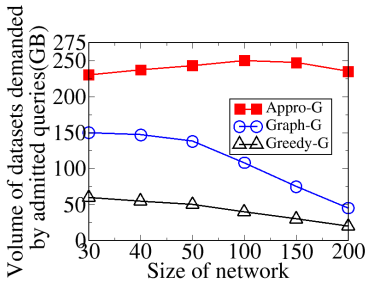


(b) The system throughput.

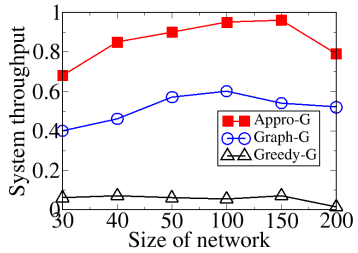
Figure 2: The performance of different algorithms Appro-S, Greedy-S and Graph-S in terms of the volume of datasets demanded by admitted queries and the system throughput, where each query demands a single dataset each time.

We first evaluate the proposed algorithm Appro-S against algorithms Greedy-S and Graph-S by varying the network size for the special case where each query demands a single dataset each time, in terms of the volume of datasets demanded by admitted queries and the *system throughput* which is a ratio of the number of admitted queries to the total number of queries in the system. It can be seen from Fig. 2(a) and Fig. 2(b) that the volume of datasets

demanded by admitted queries is over 4 times than that by algorithm Greedy-S and 2 times than that by algorithm Graph-S, the system throughput by Appro-S is 15% higher than that by algorithm Greedy-S and 10% higher than that by Graph-S, respectively. The rationale behind is that Appro-S places the replicas of datasets from an overall perspective, it jointly considers data replication and query assignment by smartly finding appropriate number and placement locations of replicas for each dataset, it also fully utilizes the available computing resource and the delay requirements of queries when placing replicas. Whereas Greedy-S intends to place a replica at a location with largest available computing resource while pays less attention to the delay requirement when choosing locations to place replicas; similarly Graph-S places replicas at locations under the constraints of location (data center or cloudlets) capacities and delay requirements of queries, it then use graph partitioning with maximum volume of datasets demanded by admitted queries, it thus can better user the resources of locations compared with Greedy-S but not fully make use of the resources and admit as many queries as possible compared with *Appro* − *S*. Notice that when the network size is too high, e.g., 200, the system throughput and volume of datasets demanded by admitted queries slightly decrease, this is because when the network size is too large, transmission delay of some paths from evaluation locations to home locations of queries has a higher probability to increase which may violate the delay requirements of some queries, thereby reducing the system throughput and the volume of datasets demanded by admitted queries.



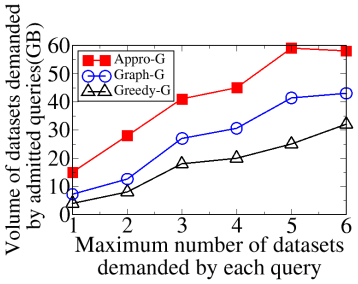
(a) The impact on the volume of datasets demanded by admitted queries.



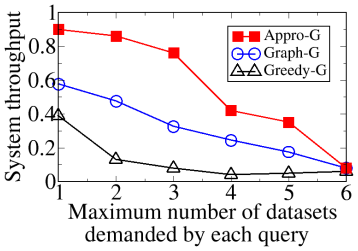
(b) The impact on the system throughput.

Figure 3: Impacts of the maximum number of datasets demanded by each query on the performance by Appro-G, Greedy-G and Graph-G.

We then evaluate the proposed algorithm Appro-G against algorithms Greedy-G and Graph-G by varying the network size for the general case where each query demands multiple datasets each time, in terms of the volume of datasets demanded by admitted queries and the system throughput. It can be seen from Fig. 3(a) and Fig. 3(b) that the volume of datasets demanded by admitted queries is 5 and 1.7 times than those by algorithms Greedy-G and Graph-G, respectively. The system throughput by Appro-G is 2.1 and 1.5 times than those by algorithms Greedy-G and Graph-G, respectively. The arguments are the same as that in Fig. 2, we do not repeat here.



(a) The volume of datasets demanded by admitted queries.

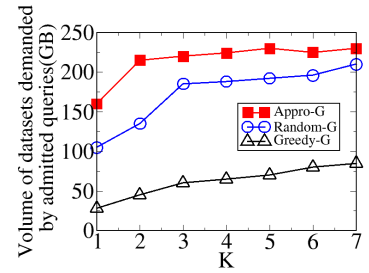


(b) The system throughput.

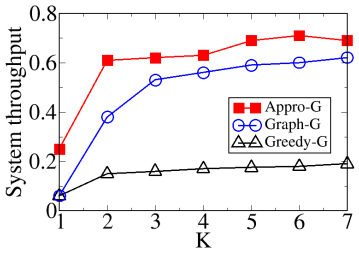
Figure 4: The performance of different algorithms Appro-G, Greedy-G and Graph-G in terms of the volume of datasets demanded by admitted queries and the system throughput, where each query demands multiple datasets each time.

Impact of the maximum number of datasets demanded by each query on the algorithmic performance: We now evaluate the impact of the maximum number of datasets demanded by each query by varying the number from 1 to 6 for the general case where each query demands multiple datasets each time, on the performance of algorithms Appro-G, Greedy-G and Graph-G, in terms of the volume of datasets demanded by admitted queries and the system throughput. Notice that we did not evaluate the impact of the maximum number of datasets demanded by each query on the algorithmic performance for the special case, as a query only demands a single dataset each time for the special case. For simplicity, we refer to the maximum number of datasets demanded by each query as *F*. From Fig. 4(a) we can see that the system throughput of three algorithms decreases with the growth of *F*,

the rationale is that a query could be admitted by the system, only when the delay for evaluating all datasets demanded by the query is no greater than the delay requirement of the query, that is to say, the more number of datasets is demanded by a query, the harder the QoS requirements of queries would be satisfied, so the harder the query would be admitted. Although the system throughput decreases with the growth of F, the volume of datasets demanded by admitted queries firstly increases with the growth of F from 1 to 5, and then slightly decreases after $F = 5$. The reason is that before $F = 5$, the total number of datasets demanded by admitted queries increases as queries demand more datasets, however when F is 6, so many queries are rejected by the system due to the violated delay requirements, the volume of datasets demanded by admitted queries thus decreases. It can be clearly seen that the volume of datasets demanded by admitted queries and system throughput by algorithm `Appro-G` is higher than those by algorithms `Greedy-G` and `Graph-G`, the reasons are similar those of Figs. 2(a) and 2(b).



(a) The impact of K on the volume of datasets demanded by admitted queries.



(b) The impact of K on the system throughput.

Figure 5: Impacts of the maximum number K of replicas of each dataset on the performance by `Appro-G`, `Greedy-G`, and `Graph-G` in terms of the volume of datasets demanded by admitted queries and the system throughput.

Impacts of the maximum number K of replicas on the algorithmic performance: We then evaluate the impact of the maximum number K of replicas of a dataset by varying K from 1 to 7 for the general case where each query demands multiple datasets each time, on the performance of `Appro-G`, `Greedy-G` and `Graph-G` in terms of the volume of datasets demanded by admitted queries and the system throughput. From Fig. 5(a) and Fig. 5(b) we can see that the volume of datasets demanded by admitted queries and system throughput are increasing with the growth of the value of K, the rationale is that as more replicas of each dataset are placed in the system, the delay requirements of queries are easier to be satisfied, thus the system throughput and volume of datasets demanded by admitted queries increase. Obviously, the volume of datasets demanded by admitted queries and system throughput achieved by `Appro-G` are significantly higher than those by `Greedy-G` and `Graph-G`. The reason is that `Appro-G` places the replicas of datasets from the perspective of all the system to optimize the use of system resources, it jointly considers data replication and query assignment by smartly finding appropriate number and placement locations of replicas for all datasets, it also fully utilizes the available computing resource and the delay requirements of queries when placing replicas.

4.3 Performance evaluation in a real test-bed

We now evaluate the performance of the proposed algorithms in a real testbed that is composed of virtual machines in different geo-locations that are provided by a cloud service provider, and a controller that executes the proposed algorithms.

Testbed settings: We lease 20 virtual machines (VMs) from a cloud service provider DigitalOcean [9], these VMs are located at locations San Francisco, New York, Toronto, and Singapore. It must be mentioned that since we focus on the replica placement in a two-tier edge cloud, we use 4 VMs to represent data centers, and 16 VMs to represent cloudlets in the edge cloud network G, we also use a local server as a controller to control the running of algorithms and 2 switches. Although the scale of each node representing a data center in this testbed may not be comparable to a large-scale data center, the implementation can be easily extended to a test-bed with large-scale data centers. An illustration of the testbed is in Fig. 6.

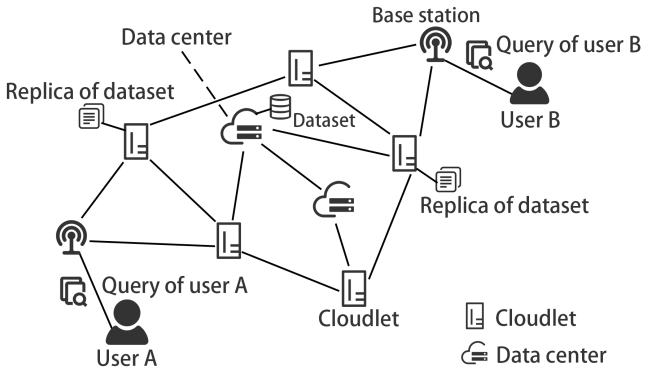
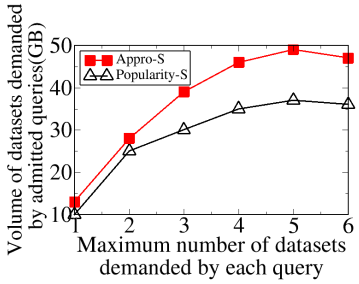


Figure 6: The topology of the testbed with leased VMs.

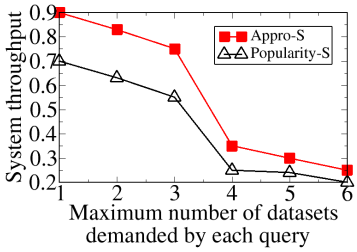
Datasets: The datasets used in the experiment are mobile application usage information from 3 million anonymous mobile users for a period of three months. We divide the data into a number of datasets according to the data creation time, and randomly distribute the datasets into the data centers and cloudlets of the testbed. Big data analytic queries are issued to find some evaluation results:

such as the most popular applications, at what time the found applications would be used, and the usage pattern of some mobile applications, etc.

Results: We first evaluate the performance of the proposed algorithm `Appro-S` against a benchmark `Popularity-S` for the special case where a query demands one single dataset each time by varying the maximum number of datasets demanded by each query. Due to page limits, we here put only a set of figures about the impact of the maximum number F of datasets demanded by each query on the performance of algorithm `Appro-S` against benchmark `Popularity-S` illustrated in Fig. 7. From Figs. 7(a) and 7(b) we can see that, algorithm `Appro-S` outperforms algorithm `Popularity-S` by delivering a higher volume of datasets demanded by admitted queries and system throughput. We can see that the volume of datasets demanded by admitted queries increases with the growth of F from Fig. 7(a), and the system throughput decreases as the value of F increases from Fig. 7(b), the arguments are similar with those in Figs. 4(a) and 4(b).



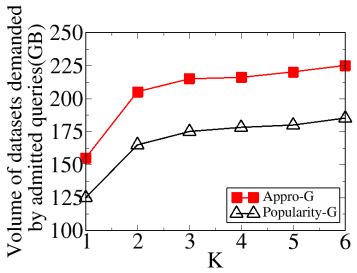
(a) The volume of datasets demanded by admitted queries by `Appro-S` and `Popularity-S` on the real testbed.



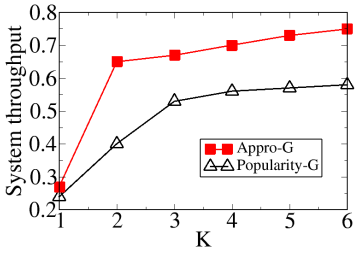
(b) The system throughput by `Appro-S` and `Popularity-S` on the real testbed.

Figure 7: The performance evaluation of the proposed algorithm `Appro-S` against benchmark `Popularity-S` on the real testbed for the special case.

We then investigate the performance of the proposed algorithm `Appro-G` against benchmark `Popularity-G` for a general case where each query demands multiple datasets each time, by varying the number K of dataset replicas. Comparably, because of page limits and pattern similarity, we here put only one set of figures about the impact of the maximum number K of replicas of each dataset on the algorithmic performance. It can be seen from Figs. 8(a) and 8(b) that `Appro-G` achieves a higher volume of datasets demanded by admitted queries and a higher system throughput than those by `Popularity-G`. The rationale behind is that `Appro-G` places the replicas of datasets from the perspective of the whole system by smartly finding appropriate number and placement locations of replicas for all datasets, it also fully utilizes the available computing resource and the delay requirements of queries when placing replicas. The volume of datasets demanded by admitted queries and the system throughput increase with the growth of K. This is because as more replicas of each dataset are placed in the system, the delay requirements of queries are easier to be satisfied, thus the volume of datasets demanded by admitted queries and system throughput increase.



(a) The volume of datasets demanded by admitted queries by `Appro-G` and `Popularity-G` on the real testbed.



(b) The system throughput by `Appro-G` and `Popularity-G` on the real testbed.

Figure 8: The performance evaluation of the proposed algorithm `Appro-G` against benchmark `Popularity-G` on the real testbed for the general case.

5 RELATED WORK

Several studies on data placement and query evaluation have been conducted in the past [1, 6, 7, 17, 18, 20, 22–26], and the others focused on multi-layered network architecture and edge clouds for dealing with big data [11, 14–16, 27]. Most of these studies either did not consider data replications of generated big data [1, 11, 14–16, 20, 26, 27] or ignored the QoS requirement of users [1, 6, 17, 23, 26], or some of them only considered traffic cost while neglecting other costs [17].

For example, Baev *et. al.* [6] considered a problem of placing replicated data in arbitrary networks to minimize the total storage

and access cost. Golab *et al.* [10] studied a data placement problem to determine where to store the data and where to evaluate data-intensive tasks with a goal to minimize the data traffic cost. Kayyoor *et. al* [17] addressed a problem of minimizing average query span, which is the number of servers involved in answering a query. They ignored other costs and QoS requirements of users [6, 17], and did not consider data replications [10]. Agarwal *et al.* [1] proposed a data placement mechanism Volley for geo-distributed cloud services to minimize the user-perceived latency. Xia *et. al* [26] considered a big data management problem in distributed cloud environments to maximize the system throughput while minimizing the operational cost of service providers. No data replications and QoS requirements of users are discussed in the two works [1, 26]. Pu *et al.* [20] presented a system for low latency geo-distributed analytics, which used an heuristic to redistribute datasets among the data centers prior to queries' arrivals, and placed the queries to reduce network bottlenecks during the query's execution. Heintz *et al.* [12] studied the tradeoff between the delay and errors of obtained results in streaming analytics in an architecture consisting of a single center and multiple edge servers. In the study [20], authors did not consider data replications of datasets. The work in [16] considered a layered architecture for the satellite-based data center infrastructure, and big data storage by leveraging such data centers. The authors [14, 15] studied a service provisioning problem in the edge cloud network, with an objective to maximize the profit of network operators. No data replication is considered in these works [14–16].

In contrast, we studied the proactive QoS-aware data replication and placement problem for query evaluation of big data analytics in a two-tier edge cloud environment, where the number of replicas of big datasets should be appropriately determined and the locations to place the replicas should be strategically selected, with an objective to maximize the volume of datasets demanded by admitted queries such that the service providers can obtain maximum benefits by offering a *pay-as-you-go* pricing approach to process the datasets, while meeting the QoS requirements of queries and resource capacity constraints.

6 CONCLUSIONS

In this paper, we studied query evaluation of big data analytics in a two-tier edge cloud network through efficient and effective data replication and placement with the aim to maximize the volume of datasets demanded by admitted queries, subject to computing resource capacities on data centers and edge cloudlets, while meeting various delay requirements of user queries. To this end, we first formulated a novel QoS-aware data replication and placement problem of query evaluation for big data analytics. We then proposed an efficient approximation algorithm with provable approximation ratio for the problem. We finally evaluated the performance of the proposed algorithm through experimental simulations in a real testbed based on real datasets. Simulation results demonstrate that the proposed algorithm achieves several times higher volume of datasets demanded by admitted queries and system throughput than existing works.

ACKNOWLEDGEMENT

The work of Qiufen Xia and Zichuan Xu is partially supported by the National Natural Science Foundation of China (Grant No. 61802047, 61802048, 61772113, 61872053), the fundamental research funds for the central universities in China (Grant No. DUT19RC(4)035, DUT19RC(5)001, DUT19GJ204), and the "Xinghai Scholar" Program at Dalian University of Technology, China.

REFERENCES

[1] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan. Volley: automated data placement for geo-distributed cloud services. *Proc. of NSDI*, USENIX, 2010.
[2] *https://aws.amazon.com/ec2/*, accessed in Jan. 2019.
[3] *https://aws.amazon.com/pricing/?nc1=h_ls*, accessed in Jan. 2019.
[4] *https://aws.amazon.com/s3/*, accessed in Jan. 2019.
[5] H. An, M. Singh, and O. Svensson. LP-based algorithms for capacitated facility location. *Proc. of FOCS'14*, IEEE, 2014.
[6] I, Baev, R. Rajaraman, and C. Swamy. Approximation algorithms for data placement problems. *SIAM J. on Computing*, Vol.38, No.4, pp.1411-1429, 2008.
[7] M. W. Convolbo, J. Chou, and S. Lu. DRASH: A data replication-aware scheduler in geo-distributed data centers. *Proc. of CloudCom*, IEEE, 2016.
[8] K. Calvert, and E. Zegura. Gt-itm: georgia tech internetwork topology models.
[9] Digital Ocean. *https://www.digitalocean.com*, accessed in Jan. 2019.
[10] L. Golab, M. Hadjieleftheriou, H. Karloff, and B. Saha. Distributed data placement to minimize communication costs via graph partitioning. *Proc. of SSDBM*, ACM, 2014.
[11] S. Guo,D. Zeng, L. Gu, and J. Luo. When green energy meets cloud radio access network: joint optimization towards brown energy minimization. *Mobile Networks and Applications*, Springer, pp.1-9, 2018.
[12] B. Heintz, A. Chandra, and R. K. Sitaraman. Trading timeliness and accuracy in geo-distributed streaming analytics *Proc. of SoCC*, ACM, 2016.
[13] T. Hou, G. Feng, S. Qin, and W. Jiang. Proactive content caching by exploiting transfer learning for mobile edge computing. *International Journal of Communication Systems*, Vol, 31, No. 2, 2017.
[14] H. Huang, and S. Guo Adaptive service provisioning for mobile edge cloud. *ZTE Communications*, Vol. 15, No. 2, pp.1-9, 2017.
[15] H. Huang, and S. Guo Service provisioning update scheme for mobile application users in a cloudlet network *Proc. of ICC*, IEEE, 2017.
[16] H. Huang, S. Guo, and K. Wang. Envisioned wireless big data storage for low-earth-orbit satellite-based cloud. *IEEE Wireless Communications*, Vol.25, No.1, pp.26-31, 2018.
[17] A. K. Kayyoor, A. Deshpande, and S. Khuller. Data placement and replica selection for improving co-location in distributed environments. Computing Research Repository (CoRR), arXiv:1302.4168, 2012.
[18] P. Li , S. Guo, T, Miyazaki, X. Liao, H. Jin, A. Y. Zomaya, and K. Wang. Traffic-aware geo-distributed big data analytics with predictable job completion time. *IEEE Trans. on Parallel and Distributed Systems*, Vol.28, No.6, pp.1785-1796, 2017.
[19] H. Li, H. Xu, and S. Nutanong. Bohr: similarity aware geo-distributed data analytics. *Open Access Media*, USENIX, 2017.
[20] Q. Pu, G. Ananthanarayanan, P. Bodik, S. Kandula, A. Akella, P. Bahl, and I. Stoica. Low latency analytics of geo-distributed data in the wide area. *Proc. of SIGCOMM*, ACM, 2015.
[21] S. Rao, R. Ramakrishnan, A. Silberstein, M. Ovsiannikov, and D. Reeves. Sailfish: a framework for large scale data processing. *Proc. of SoCC*, ACM, 2012.
[22] W. Xiao, W. Bao, X. Zhu, and L. Liu. Cost-aware big data processing across geo-distributed data centers. *IEEE Trans. on Parallel and Distributed Systems*, Vol.28, No.11, pp.3114-3127, 2017.
[23] Q. Xia, W. Liang, and Z. Xu. The operational cost minimization in distributed clouds via community-aware user data placements of social networks. *Computer Networks*, Vol.112, pp.263-278, 2017.
[24] Z. Xu and W. Liang. Operational cost minimization for distributed data centers through exploring electricity price diversity. *Computer Networks*, Vol. 83, pp.59-75, Elsevier, 2015.
[25] Z. Xu, W. Liang, and Q. Xia. Electricity cost minimization in distributed clouds by exploring heterogeneities of cloud resources and user demands. *Proc. of ICPADS'15*, IEEE, 2015.
[26] Q. Xia, Z. Xu, W. Liang, and A. Zomaya. Collaboration- and fairness-aware big data management in distributed clouds. *IEEE Trans. on Parallel and Distributed Systems*, Vol.27, No.7, pp.1941-1953, 2016.
[27] S. Yu, M. Liu, W. Dou, X. Liu, and S. Zhou. Networking for big data: A survey. *IEEE Communications Surveys & Tutorials*, Vol. 19, No.1, pp. 531-549, 2017.