QoS-Aware Data Replications and Placements for Query Evaluation of Big Data Analytics

Qiufen Xia†, Weifa Liang† and Zichuan Xu‡
† Research School of Computer Science, Australian National University, Canberra, ACT 2601, Australia
‡ Department of Electronic and Electrical Engineering, University College London, London, UK
qiufen.xia@anu.edu.au, wliang@cs.anu.edu.au, z.xu@ucl.ac.uk

***Abstract*—Enterprise users at different geographic locations generate large-volume data and store their data at different geographic datacenters. These users may also issue ad hoc queries of big data analytics on the stored data to identify valuable information in order to help them make strategic decisions. However, it is well known that querying such large-volume big data usually is time-consuming and costly. Sometimes, users are only interested in timely approximate rather than exact query results. When this approximation is the case, applications must sacrifice either timeliness or accuracy by allowing either the latency of delivering more accurate results or the accuracy error of delivered results based on the samples of the data, rather than the entire set of data itself. In this paper, we study the QoS-aware data replications and placements for approximate query evaluation of big data analytics in a distributed cloud, where the original (source) data of a query is distributed at different geo-distributed datacenters. We focus on placing the samples of the source data with certain error bounds at some strategic datacenters to meet users' stringent query response time. We propose an efficient algorithm for evaluating a set of big data analytic queries with the aim to minimize the evaluation cost of the queries while meeting their response time requirements. We demonstrate the effectiveness of the proposed algorithm through experimental simulations. Experimental results show that the proposed algorithm is promising.**

I. Introduction

Cloud service providers, such as Microsoft, Google, and Facebook are deploying datacenters globally to provide users ubiquitous access to their cloud services [3], [16], [17], [18]. With more and more people adopting cloud services, large volume data on user activities and session logs, – referred to as *big data* – are being produced at exponential rates. It is estimated that at least 2.5 quintillion bytes of data is created per day [10]. Such big data plays a significant role in enterprises and people's daily lives, especially within enterprises, as the analytic results of big data can make the enterprises better understand their customers and behaviors, and help them seize their best growth opportunities. Examples of big-data analytics include querying user logs to make advertisement decisions, query network logs to detect various network attacks such as DoS attacks, etc. However, querying big data is time-consuming and costly. Indeed, a linear scan of a dataset of PB size (10^{15} bytes) takes days using a solid state drive with a read speed of 6GB/s, and takes years if the dataset is of EB size (10^{18} bytes) [6]. In addition, big data usually is distributed in different geo-datacenters, this poses many challenges in big data analytics that the evaluation of such a big data analytic query usually needs the source data from multiple datacenters and these data must be jointly considered. One important issue related to the challenges is how to ensure the QoS requirements of users in terms of *access delays* (the query response times), given that the query results will be used in timely decision-making applications.

One promising solution to tackle the mentioned challenges is Approximate Query Processing (AQP), which evaluates the queries based on the sample data of the original data, and returns an approximate result with an error bound in accuracy. By leveraging sampling technologies, the query response time can be significantly reduced [2]. To this end, an important approach is to replicate the samples of a large dataset to multiple datacenters so that query users can obtain their desired query results with bounded errors within their specified time duration. Intuitively, a more accurate query result with a small error bound usually takes a longer time in data processing and transmission within the distributed datacenter network, while an approximate result with a large error bound may take much less time.

Although data sampling and replications can improve system performance, it does not imply that more sample replicas will lead to better system performance, since the maintenance of data consistency between data samples and their slave sample copies in the network does incur cost. To maximize the benefit by approximate query processing and sample replications, strategic replicating and placing samples of each dataset in a distributed cloud is essential by minimizing the cost of evaluating queries approximately while meeting the query response times of users. One fundamental problem thus is how to replicate and place the samples with different error bounds to different datacenters in the distributed cloud so that big data queries can be evaluated timely and accurately.

Several studies on data placement have been conducted in the past [1], [4], [7], [13]. However, most of these studies considered neither data replications of the generated big data [1], [7], [13] nor QoS requirements on the access delays of users [1], [4], [7], [13]. In addition, there are several studies on query evaluation and data distribution [8], [9], [11]. Although some of them considered the data traffic cost, they did not incorporate the QoS requirements of users [9], or data replications and placements [8], [11]. In this paper, we study data replications and placements of generated big data in a distributed cloud for big data analytics with the aim to minimize the query evaluation cost while meeting user QoS requirements (query response times).

The main contributions of this paper are as follows. We first formulate a novel QoS-aware data replication and placement

978-1-4673-8999-0/17/$31.00 ©2017 IEEE

problem for big data query evaluation in a distributed cloud environment. We aim to minimize the evaluation cost of such queries while meeting user query response time requirements, where the evaluation cost of a query includes data processing cost, data storage cost, data transmission cost and data update cost. We then propose an efficient algorithm for the problem through a non-trivial reduction. 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, reducing the evaluation cost of queries significantly compared to the other baseline algorithm. To the best of our knowledge, this is the first time that the QoS-aware data replication and placement problem for big data query evaluation in distributed clouds is considered, and an efficient algorithm is devised.

The remainder of this paper is organized as follows. Section II introduces the system model and problem definition, followed by the proposed heuristic algorithm in Section III. The performance evaluation of the proposed algorithm is conducted in Section IV. The related work is presented in Section V, and the conclusions are given in Section VI.

II. Preliminaries

A. The distributed cloud

We consider a distributed cloud $G = (\mathcal{DC}, E)$, which consists of a set $\mathcal{DC}$ of datacenters located at different geographical locations and inter-connected by a set E of communication links (or paths). Let $DC_i \in \mathcal{DC}$ be a datacenter and $e_{ij} \in E$ a link between two datacenters DC_i and DC_j. The computing resource of each datacenter DC_i is used to evaluate queries, while its storage resource is used to store data and the query results. Denote by $A(DC_i)$ and $B(DC_i)$ the amount of available computing resource and the capacity of computing resource in datacenter $DC_i \in \mathcal{DC}$, and let r_c be the amount of computing resource allocated to process one unit data. We do not restrict the capacity of storage resource of datacenters, as the storage resource usually is abundant and cheap, compared with the expensive computing resource [13]. The processing, storage of data at datacenters and the transmission of data along network links consume various cloud resources and thus incur costs of the cloud service provider. Denote by $c_p(DC_i)$ and $c_s(DC_i)$ the costs for processing and storing a unit data at DC_i. Denote by $c_t(i,j)$ and $d_t(i,j)$ the transmission cost and delay on link $e_{ij} \in E$ for transferring a unit of data.

B. Big data, repeated approximate queries, and user QoS requirement

Enterprise users, such as constituent companies, dynamically generate large volume of data from web logs, click streams, sensors, and many other sources, which is then stored at their specified datacenters. We refer to the data generated by a user as the *dataset* of the user, and term the specified datacenter for the data storage as the *home datacenter* of the data. Let $\mathcal{S}$ be the collection of datasets generated by all users, denote by S_j a dataset in $\mathcal{S}$, where $1 \leq j \leq J$ with J representing the number of datasets in $\mathcal{S}$, i.e., $J = |\mathcal{S}|$. Denote by $DC(S_j)$ the datacenter at which dataset S_j is located.

In addition to generating big data, enterprise users also consume data that are generated by themselves and other users by querying the data to explore potential business values. For example, enterprise decision makers may query their user logs from multiple datasets repeatedly to make timely advertisement decisions, or exploit the data to gain business insights on their customers. For example, a user may issue a query like 'count the number of customers satisfying the following condition... with the best possible accuracy within 1 hour'. As timeliness for many queries is more important than the accuracy of the query solutions, in this paper, we study approximate queries that help users get a 'rough picture' of multiple datasets by delivering an approximate result with a certain accuracy while meeting their stringent query response times. Specifically, we consider *repeated approximate queries*, i.e., the approximate queries are associated with frequencies and the queries issued by the same user based on relatively stable datasets, i.e., the dataset might experience minor fluctuations between two consecutive queries. Denote by $Q = \{q_m \mid 1 \leq m \leq M\}$ the set of approximate queries in the system, where M is the number of queries, and q_m is a repeated approximate query by a user with frequency f_m, which represents the number of its submissions that query q_m is issued. The evaluation of each approximate query q_m may demand several datasets distributed at different datacenters in G, and let $\mathcal{S}(q_m)$ be the collection of datasets demanded by q_m.

As we consider approximate query evaluations within stringent query response times, we refer to *the response time of a query* as the QoS of the query, where the query response time of a query is the evaluation duration of the query between the query is issued and the query result obtained at its home datacenter. Denote by d_m the maximum tolerable response time of query q_m.

C. Stratified samples and sample replication

To accurately and quickly answer each approximate query q_m, a set of samples of each dataset $S_j \in \mathcal{S}$ has been created, following the stratified sampling strategy [5]. A *stratified sample* refers to a sample that is not drawn from the whole dataset in a random way, but separately from a number of disjoint strata of the dataset in order to ensure a more representative sample. Each created stratified sample with a sample size $n_{j,k}$ is referred to as the *origin sample* of the dataset, the sample can return the query result with an error bound ϵ_k, where $k \in \mathbb{Z}^+$. Following the theory of stratified sampling [5], the error bound ϵ_k of a sample of size $n_{j,k}$ is inversely proportional to the square root $\sqrt{n_{j,k}}$ of its size $n_{j,k}$. Therefore, a stratified sample with a larger error bound usually has a smaller size, thus requires less computing resource to process and a shorter delay to deliver the result. We further assume that the origin stratified samples of each dataset S_j are materialized at the datacenter where S_j is generated.

To meet the response time requirement of every approximate query, some *slave samples* of each origin sample of dataset S_j may be created and placed at other datacenters. Data updates will be performed for each slave sample if there is any update to its origin sample to make the slave sample consistent with

its origin sample. We assume that the average data size of an update operation is $\psi \cdot n_{j,k}$, where ψ is a constant with $0 < \psi < 1$ [12] and $n_{j,k}$ is the sample size. It is obvious that more slave samples are in the system, the QoS requirements of users tend to be satisfied. However, the update and storage costs on the slave samples will subsequently grow too. Therefore, creating and placing a proper number of slave samples with certain error bounds for each origin sample is crucial.

Evaluating an approximate query q_m is to abstract intermediate results from the samples of each requested dataset, and aggregate the obtained intermediate results at the home datacenter of the query. Let $h(q_m)$ be the home datacenter of q_m. Without loss of generality, we assume that the volume of the approximate query result on each its component sample of S_j is proportional to the volume of each of the samples, i.e., $\beta \cdot n_{j,k}$ for sample $S_{j,k}$, where β is a constant with $0 < \beta \leq 1$ [12]. An approximate query q_m may demand multiple datasets and be evaluated on their samples with different error bounds, the intermediate results of these samples will be transmitted from their datacenters to the home datacenter of q_m, the *average error bound* of a query result is related to the sizes and error bounds of placed samples, which can be calculated as follows [19]. For example, assume that its component sample sizes of the query are n_{j_1,k_1}, n_{j_2,k_2} and n_{j_3,k_3} with error bounds ϵ_{k_1}, ϵ_{k_2}, and ϵ_{k_3}, respectively. Then, the average error bound of the approximate query result is $\frac{n_{j_1,k_1} \cdot \epsilon_{k_1} + n_{j_2,k_2} \cdot \epsilon_{k_2} + n_{j_3,k_3} \cdot \epsilon_{k_3}}{n_{j_1,k_1} + n_{j_2,k_2} + n_{j_3,k_3}}$.

D. Cost model

The cost of approximate query evaluation in a distributed cloud consists of four component costs. The *storage cost* is for storing both origin samples and slave samples of each dataset in $\mathcal{S}$; the *process cost* is the cost for evaluating the samples requested by approximate queries; the *update cost* refers to the cost of keeping the slave samples consistent with their origin samples; and the *transmission cost* is the cost of data transfer within the network by transferring the intermediate results of each query q_m from the datacenters where the requested samples are evaluated to the home datacenter $h(q_m)$ of q_m, and transferring the updated data from each origin sample to its slave samples.

E. Problem definition

Given a distributed cloud $G = (\mathcal{DC}, E)$, a set $\mathcal{S}$ of datasets, a set of approximate queries $Q = \{q_m \mid 1 \leq i \leq M\}$ for big data analytics, each approximate query q_m is issued with a frequency f_m, each dataset $S_j \in \mathcal{S}(q_m)$ is generated at datacenter $DC(S_j)$, a variety of origin samples with different sample sizes and error bounds for each dataset S_j are stored at datacenter $DC(S_j)$, the intermediate results evaluated on these requested datasets will be aggregated at the home datacenter of q_m, and the QoS requirement d_m of q_m is given in advance. The *QoS-aware data replication and placement problem for query evaluation of big data analytics* in G is to create a set of slave samples for each origin sample and place the slave samples at strategic datacenters in G such that the evaluation cost of all approximate queries is minimized, while meeting the specified QoS d_m of each approximate query $q_m \in Q$.

III. A HEURISTIC ALGORITHM

A. Overview

There are two issues to be addressed to the QoS-aware data replication and placement problem.

One is how many slave samples of each origin sample requested by a query should be created, and where these slave samples should be placed in the distributed cloud. Specifically, the queries in Q have not only different required datasets but also different QoS requirements, it is necessary to create a different number of slave samples for each origin sample at different datacenters; otherwise, some users' QoSs may be not satisfied. To address this issue, we jointly place each query and the samples of its required datasets, to make sure that each query is evaluated at close datacenters (to meet its delay requirement) where the samples of its required datasets has/will be placed. To this end, we reduce the problem to the problem of finding unsplittable minimum-cost multi-commodity flow in an auxiliary directed graph $G_f = (V_f, E_f; u, c)$, where the pair of a query q_m and each of its required dataset $\mathcal{S}(q_m)$ is referred to as a *commodity*. Each commodity needs to be routed to a datacenter, which corresponds to creating a slave sample of the required dataset at the datacenter or using an existing slave sample that has been placed at the datacenter already, and evaluating a query based on the placed sample.

Another issue is which slave sample should be created at a datacenter for a query, since a dataset required by the query has several origin samples with different sizes and error bounds, determining which error bound that each query should be evaluated is critical to both the quality of the query result and its evaluation cost. We thus route each commodity (q_m, S_j) by assuming that query q_m requires the sample of dataset S_j with the smallest error bound. If the computing resource of all datacenters is not enough to evaluate all queries at their smallest error bounds, the algorithm will greedily select some queries by increasing their samples' error bounds, as larger error bounds indicate smaller sample sizes, leading to less computing resource to process. This procedure continues until all commodities are routed successfully.

B. Algorithm

We now describe the detailed algorithm, which consists of two phases: (1) the construction of the flow graph $G_f = (V_f, E_f; u, c)$; and (2) routing samples to datacenters for query evaluation. We proceed by increasing the error bounds of samples gradually.

The construction of the flow graph $G_f = (V_f, E_f; u, c)$ is as follows. Since not only each query needs to be assigned to a datacenter but also the samples of its required datasets should be placed, we use a *commodity node* in G_f to represent a query q_m and each of its demanded datasets, i.e., (q_m, S_j), where $S_j \in \mathcal{S}(q_m)$ is a dataset that query q_m demands. Routing this commodity thus means both the assignment of the query q_m and the placement of the sample of its required dataset S_j. In other words, to evaluate query q_m, we need to route all the commodity nodes of the query to some datacenters in the distributed cloud G. Therefore, each datacenter $DC_i \in \mathcal{DC}$ is treated as a *datacenter node* DC_i^f in G_f, denote by $\mathcal{DC}^f$ the set of datacenter nodes. Each datacenter node $DC_i^f \in \mathcal{DC}^f$

has a *virtual datacenter node* $DC_i^{'f}$, and a directed edge $\langle DC_i^f, DC_i^{'f}\rangle$ from DC_i^f to $DC_i^{'f}$ is added with its capacity representing the volume of data that can be processed by DC_i, denote by $\mathcal{DC}^{'f}$ the set of virtual datacenter nodes. In addition, a *virtual source node* s_0 and a *virtual sink node* t_0 are added to G_f, i.e., $V_f = \{s_0\} \cup \big(\bigcup(q_m, S_j)\big) \cup \mathcal{DC}^f \cup \mathcal{DC}^{'f} \cup \{t_0\}$, where $S_j \in \mathcal{S}(q_m)$ and $1 \le m \le M$. There is a directed edge from the virtual source node s_0 to each commodity node (q_m, S_j), its capacity and cost are the volume of S_j and 0 respectively. An edge $\langle (q_m, S_j), DC_i^f\rangle$ from a commodity node (q_m, S_j) to a datacenter node DC_i^f is added to E_f, if the response time requirement of query q_m for evaluating on a sample at datacenter DC_i is satisfied. The capacity of edge $\langle (q_m, S_j), DC_i^f\rangle$ is the volume of S_j. If a slave sample of S_j has not been placed at DC_i, the cost of edge $\langle (q_m, S_j), DC_i^f\rangle$ is the sum of the storage cost for storing a unit of data at datacenter DC_i, the update cost for updating a unit of data along the shortest path with minimum cost between the datacenter where its origin sample is and DC_i, and the transmission cost for transmitting a unit of intermediate data along the path with minimum transmission cost from DC_i to the home datacenter $h(q_m)$ of q_m. Otherwise, the cost of $\langle (q_m, S_j), DC_i^f\rangle$ is set to the transmission cost for transmitting a unit of intermediate data along the path with minimum transmission cost from DC_i to $h(q_m)$, because the update and storage costs of S_j have been considered when routing other commodities. A directed edge $\langle DC_i^f, DC_i^{'f}\rangle$ from a datacenter node DC_i^f to a virtual datacenter node $DC_i^{'f}$ is added to E_f, and its capacity is the amount of data that can be processed by the available computing resource of datacenter DC_i, and its cost is the process cost of processing a unit of data at DC_i. Similarly, there is a directed edge $\langle DC_i^{'f}, t_0\rangle$ from a virtual datacenter node $DC_i^{'f}$ to the virtual sink node t_0, its capacity is infinity, and its cost is 0.

An example of the distributed cloud $G = (\mathcal{DC}, E)$ and the constructed graph $G_f = (V_f, E_f; u, c)$ are illustrated in Fig. 1, where two approximate queries q_1 and q_2 are issued. The evaluation of query q_1 requires samples of datasets S_1 and S_2, while the evaluation of query q_2 requires samples of datasets S_1, S_2 and S_3. The origin samples of datasets S_1, S_2 and S_3 are located at datacenters DC_6, DC_7 and DC_4, respectively. The response time requirement of query q_1 for evaluating on the samples of datasets S_1 and S_2 at datacenter DC_2 cannot be met, so there are no any edges between the commodity nodes of q_1 (i.e., (q_1, S_1) and (q_1, S_2)) and datacenter node DC_2^f. Similarly, there are no any edges between the commodity nodes of q_2 (i.e., (q_2, S_1), (q_2, S_2) and (q_2, S_3)) and datacenter nodes DC_1^f, DC_5^f, and DC_6^f, as the response time requirement of query q_2 for evaluating on the samples of the datasets located at datacenter DC_1, DC_5, and DC_6 cannot be satisfied.

Having constructed the auxiliary flow graph $G_f = (V_f, E_f; u, c)$, we now route all commodities in G_f one by one. As each dataset requested by a query has multiple stratified samples with different error bounds, we start routing each commodity node by routing its slave sample with the smallest error bound (largest sample size), and iteratively find

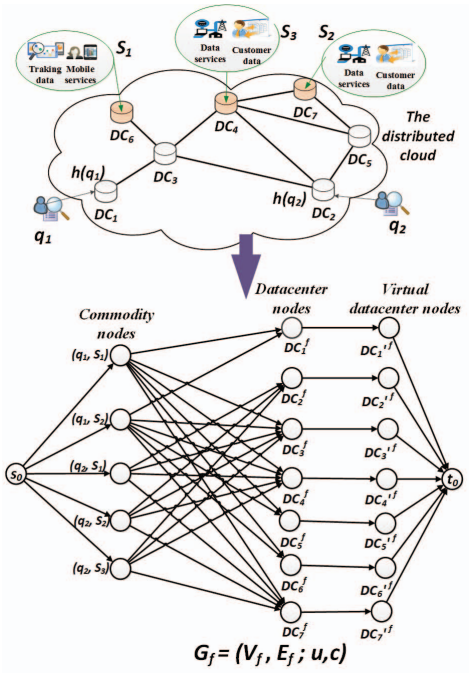


Fig. 1. An example of the distributed cloud and its auxiliary flow graph, where query q_1 requires the samples of datasets S_1 and S_2 for its evaluation, while query q_2 requires the samples of datasets S_1, S_2, and S_3 for its evaluation.

the next smallest error bounds, while each further iteration increases the error bound of the previous iteration until the QoS requirement is met.

Assume that in the beginning of the kth iteration, some queries have been admitted at error bounds from ϵ_1 to ϵ_{k-1}. We now show how to admit the rest of queries by routing the residual commodities in G_f in the kth iteration. We route each not yet routed commodity (q_m, S_j) by assuming that q_m will evaluate on the sample of S_j with an error bound ϵ_k. We rank each commodity (q_m, S_j) according to the frequency f_m of q_m and the size $n_{j,k}$ of the sample of S_j with the error bound ϵ_k, i.e., the rank of query q_m is $\sum_{S_j \in \mathcal{S}(q_m)} f_m \cdot n_{j,k}$. We then route the commodity (q_m, S_j) with the lowest rank to the destination node t_0 through a path with minimum cost in G_f. If a selected routing path passes a datacenter node DC_i^f, then create a slave sample for the sample at datacenter DC_i and update the capacities of edges in G_f. Since there is at most one salve sample of a dataset at the same datacenter DC_i, the cost of each edge in set $\{\langle (q'_m, S_j), DC_i^f\rangle \mid q'_m \in Q \setminus \{q_m\}\}$ is updated to the transmission cost by q'_m.

Notice that some queries may not be admitted after K iterations. To admit all queries, we will increase the error bound of samples requested by admitted queries. The procedure of increasing the error bounds of admitted queries continues until all queries are admitted. The detailed procedure of the heuristic algorithm is given in `Algorithm` 1.

C. Algorithm analysis

Theorem 1: Given a distributed cloud $G = (\mathcal{DC}, E)$, a set of approximate queries Q, and the response time requirement d_m of a query q_m, there is an algorithm, `Heuristic`, for the QoS-aware data replication and placement problem for query evaluation of big data analytics, which delivers a feasible solution in $O((|Q| \cdot |S| + |\mathcal{DC}|)^3)$ time.

Algorithm 1 `Heuristic`

Input: A distributed cloud $G = (\mathcal{DC}, E)$, the set of approximate queries Q with each approximate query q_m issued at the frequency of f_m, a set of datasets $\mathcal{S}$ with each dataset being generated at datacenter $DC(S_j)$, and the response time requirement d_m of each query q_m by evaluating on some datasets in $\mathcal{S}$.

Output: The placement and replication locations of the slave samples of datasets in $\mathcal{S}$.

1: Construct an auxiliary flow graph $G_f = (V_f, E_f)$;
2: Rank the queries in Q according to their frequencies and the volume of their accessed samples in a ascending order;
3: **for** Each query q_m with the minimum rank **do**
4: Get the available computing resource of each datacenter;
5: **for** Each origin sample node S_j accessed by query q_m **do**
6: Treat the origin sample as an unsplittable commodity;
7: Find a path p_{m,j,t_0} from the origin sample node to t_0 with minimum accumulated cost of all edges along the path;
8: Check whether the computing capacity of the datacenter is violated if routing the commodity to the datacenter along path p_{m,j,t_0};
9: **if** The capacity is not violated **then**
10: Route the commodity along the path p_{m,j,t_0};
11: Update the capacities of edges in G_f;
12: Update the cost of each edge in set $\{\langle (q'_m, S_j), DC_i^f \rangle \mid q'_m \in Q \setminus \{q_m\}\}$ as the transmission cost by q'_m;
13: **else**
14: Select the sample with a higher error bound as the commodity to be routed;
15: Repeat step 8 to step 12;
16: **end if**
17: **end for**
18: **end for**
19: **return** The number and locations of stratified samples with different error bounds of all datasets.

Proof Let p be a path in the auxiliary graph G_f starting from s_0 and ending at t_0, i.e., $\langle s_0, (q_m, S_j), DC_i^f, DC_i^{'f}, t_0 \rangle$. Clearly, the response time requirement of q_m can be met as there will not be an edge from (q_m, S_j) to DC_i^f if the response time requirement can not be satisfied. We then prove that flow f along p corresponds to placing a slave sample of S_j at datacenter DC_i and assigning query q_m to DC_i for evaluating on the placed slave sample. We consider two cases: (1) a slave sample of S_j has already been placed at DC_i; and (2) there is no any slave sample of S_j at DC_i. For case (1), since there is at most one slave sample with an error bound of a dataset at a datacenter, there is no need to create another identical slave sample for S_j at DC_i. This means that no extra storage and update costs are incurred, and only processing and transmission costs are incurred. As shown in Step 12 of algorithm 1, the cost of edge $\langle (q_m, S_j), DC_i^f \rangle$ is updated to the transmission cost by query q_m for transmitting a unit of data from its home datacenter $h(q_m)$ to DC_i, and the cost of edge $\langle DC_i^f, DC_i^{'f} \rangle$ is the process cost of query q_m by processing the placed sample of S_j. Thus, the flow f via path p corresponds to the assignment of query q_m to DC_i for evaluating on a sample of S_j that has already been placed in DC_i. Similarly, for case (2) we can show that the flow f via path p corresponds to the placement of a sample of S_j and the assignment of query q_m to DC_i for evaluating on the placed sample.

`Algorithm 1` first constructs an auxiliary flow graph $G_f = (V_f, E_f)$ and then routes each query q_m and its required dataset one by one through a shortest path from s_0 to t_0. G_f contains $O(|Q||S|+|\mathcal{DC}|)$ nodes and $O(|Q||S||\mathcal{DC}|)$ edges, its construction thus takes $O(|Q||S||\mathcal{DC}|)$ time. Finding a shortest path in G_f for each pair of q_m and one of its required dataset from its commodity node $\langle q_m, S_j \rangle$ to t_0 takes $O(|V_f|^2)$, where $|V_f| = O(|Q||S| + |\mathcal{DC}|)$. Thus, the algorithm takes $O(|Q||S| \cdot (|Q||S| + |\mathcal{DC}|)^2)$ time.

IV. Performance Evaluation

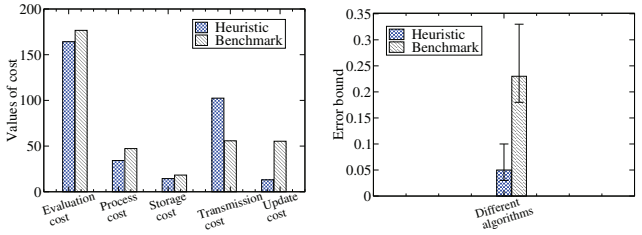
A. Simulation environment

We consider a distributed cloud consisting of 20 datacenters, there is an edge between each pair of datacenters with a probability of 0.2 generated by the GT-ITM tool. The computing capacity of each datacenter is randomly drawn from a value interval $[1,000, 2,000]$ units (GHz) [13]. Each user produces several Gigabytes of data, we thus emulate the volume of dataset generated by each user is in the range of [5, 10] GB [13], and the amount of computing resource assigned to the processing of 1GB data is a value in the range of [25, 75] GHz. The costs of transmitting, storing and processing 1 GB of data are set within [$0.05, $0.1], [$0.001, $0.0035], and [$0.05, $0.12], respectively, following typical charges in Amazon EC2 and S3. The numbers of datasets and queries in the system are randomly drawn in the range of [50, 100] and [50, 200], respectively. The frequency of a query and the number of datasets required by the query are randomly drawn from intervals [1, 10] and [1, 3]. The QoS requirement of each query is a value between [500, 1,000] ms. 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. Also, 95% confidence intervals for mean error bounds are presented.

To evaluate the performance of the proposed algorithm to the QoS-aware data replication and placement problem for query evaluation of big data analytics, a benchmark is employed as an evaluation baseline. Specifically, the benchmark first selects a set of candidate datacenters for each pair of query and one of its requested datasets, if the delay requirement of the query can be met by putting a sample with the smallest error bound of the dataset at the datacenters. It then places the sample at the datacenter with the largest available computing resource. If the datacenter cannot accommodate more samples, it then picks the next datacenter with the second largest amount of available computing resource in the set of candidate datacenters. If the available computing resource of the set of candidate datacenters cannot accommodate more samples, the algorithm then increase the error bounds of placed samples until all samples are placed. For simplicity, we refer to this benchmark as `Benchmark`, and refer to the proposed algorithm `Algorithm 1` as `Heuristic`.

B. Performance evaluation of different algorithms

We now evaluate the proposed algorithm `Heuristic` against algorithm `Benchmark`, in terms of the evaluation cost (US dollars), and the average error bound achieved by all queries in the system. It can be seen from Figs. 2(a) that the evaluation cost, the process cost, the storage cost and the update cost by algorithm `Heuristic` are substantially less than those by algorithm `Benchmark`. For example, these



(a) The various costs (b) The average error bound of all placed samples in the system

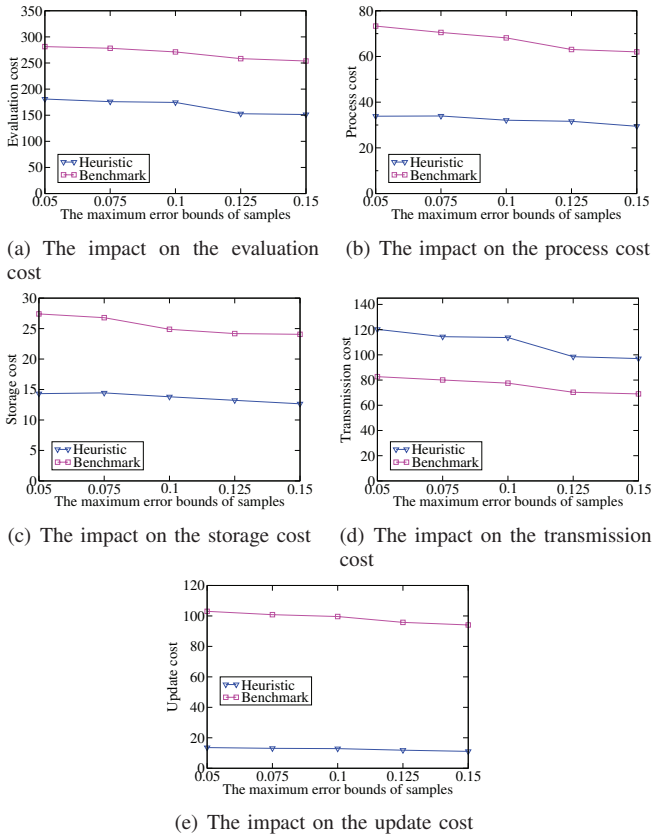
Fig. 2. The performance of different algorithms in terms of evaluation cost, the process cost, the storage cost, the update cost, and the average bound of all samples in the system.

costs by Heuristic are only about 91%, 72%, 77%, and 23% of those by algorithm Benchmark. The rational behind is that algorithm Benchmark places more slave samples for each origin sample, as it always selects the datacenter with the highest available computing resource. If a dataset is required by multiple queries, these queries may need their required sample to be replicated into multiple datacenters, depending on the workloads of datacenters at the moment when each query is evaluated. From Fig. 2(a) it can also be seen that the transmission cost by algorithm Heuristic is higher than that by algorithm Benchmark, this is because algorithm Heuristic places samples with much lower error bounds that corresponds to samples with large sizes, transmitting these samples with large sizes thus leads to a higher transmission cost. In addition, we can see from Fig. 2(b) that the average error bound of all samples placed in the system by algorithm Heuristic is only 0.05, while the one by algorithm Benchmark is 0.23, which is much higher than that by Heuristic. The reason is that algorithm Heuristic adopts a fine-grained adjustment of error bounds when there exist queries that cannot be admitted. On the other hand, algorithm Benchmark places more slave samples for each origin sample, thus, occupies more computing resource, which prevents the algorithm from placing slave samples with lower error bounds that typically have higher computing resource demands.

C. Impacts of parameters on the algorithm performance

We first evaluate the impact of the maximum error bound of samples placed in the system in terms of the evaluation cost and its component costs (US dollars), such as the process cost, the storage cost, the update cost and the transmission cost of algorithms Heuristic and Benchmark, by varying the maximum error bounds of samples from 0.05 to 0.15. From Fig. 3(a), a clear trade-off between the evaluation cost and the maximum error bound can be seen. Specifically, the evaluation costs of algorithms Heuristic and Benchmark decrease with the increase of the maximum error bound. The rationale is that with the growth of maximum error bound of samples that are placed in the system, samples with smaller sizes need to be processed and transmitted in the system, which can be evidenced by the results in Fig. 3(b), Fig. 3(c), Fig. 3(d), and Fig. 3(e), where the process, storage, transmission and update costs decrease with the increase of the maximum error bound.

We then study the impact of the number of datacenters on the performance of algorithm Heuristic against that of algorithm Benchmark. From Fig. 4, it can be seen that the



(a) The impact on the evaluation cost (b) The impact on the process cost

(c) The impact on the storage cost (d) The impact on the transmission cost

(e) The impact on the update cost

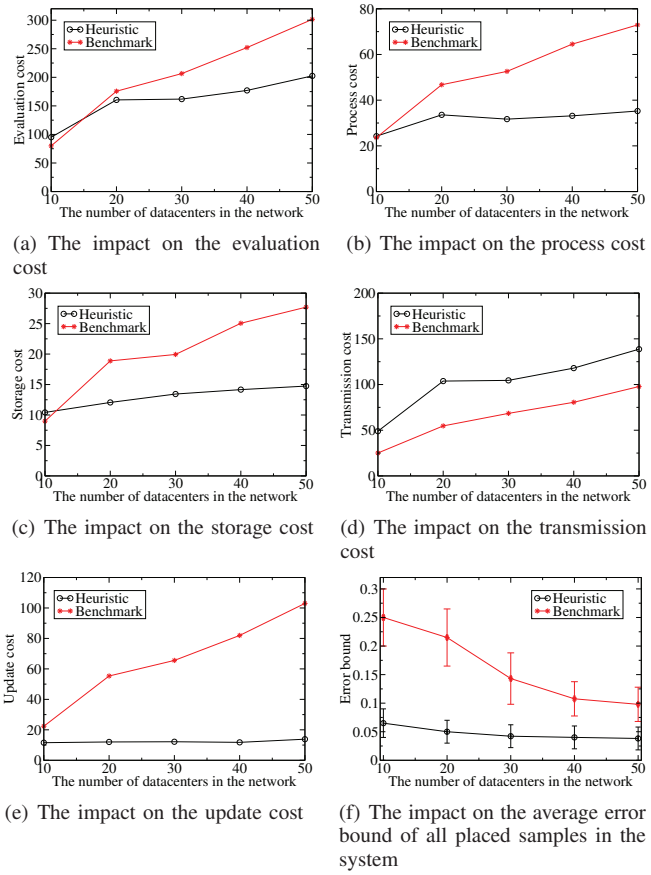
Fig. 3. Impacts of the maximum error bound of samples on the performance of algorithms Heuristic and Benchmark.

evaluation costs (US dollars) and their component costs (US dollars) of both algorithms increase with the growth of the number of datacenters, while the average error bound of all evaluated queries by algorithm Benchmark decreases with the growth of the number of datacenters. The rationale is that with the growth of the number of datacenters, more computing resource will be available to accommodate large-volume slave samples with lower error bounds, thereby increasing the costs of processing, storage, transmission and updating larger slave samples. Furthermore, as the network size grows with the increase of the number of datacenters, queries and their requested datasets tend to be placed at more datacenters.

V. Related work

Several studies on data placement and query evaluation have been conducted in the past [1], [4], [7], [8], [9], [11], [13], [14], [15], [16], [17], [18]. Most of these studies either did not consider data replications of generated big data [1], [7], [8], [11], [13] or ignored the QoS requirement of users [1], [4], [7], [9], [15], [13], or some of them only considered traffic cost while neglecting other costs [9].

For example, Baev *et. al.* [4] considered a problem of placing replicated data in arbitrary networks to minimize the total storage and access cost. Golab *et al.* [7] 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* [9] 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 [4], [7], [9], and did



(a) The impact on the evaluation cost

(b) The impact on the process cost

(c) The impact on the storage cost

(d) The impact on the transmission cost

(e) The impact on the update cost

(f) The impact on the average error bound of all placed samples in the system

Fig. 4. Impacts of the number of datacenters on the performance of algorithms `Heuristic` and `Benchmark`.

not consider data replications [7]. Agarwal *et al.* [1] proposed a data placement mechanism Volley for geo-distributed cloud services to minimize the user-perceived latency. Xia *et. al* [13] considered a big data management problem in distributed cloud environments to maximize the system throughput while minimizing the operational cost of service providers. Data replications and QoS requirements of users have not been discussed in these two studies [1], [13]. Pu *et al.* [11] presented a system for low latency geo-distributed analytics, which used an heuristic to redistribute datasets among the datacenters prior to queries' arrivals, and placed the queries to reduce network bottlenecks during the query's execution. Heintz *et al.* [8] 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 two studies [8], [11], authors did not consider data replications and samples of datasets. Xia *et al.* [15] recently investigated the placement of social networks in distributed clouds to minimize the operation cost of the service provider.

In contrast, we studied the QoS-aware data replication and placement problem for query evaluation of big data analytics in distributed cloud environments, where big data are located at different locations and users have QoS requirements in terms of query response times, with the objective to minimize the evaluation cost of all queries while meeting the QoSs of the users of these queries.

VI. Conclusions

In this paper, we studied query evaluations of big data analytics in a distributed cloud, through efficient and effective data replications and placements to minimize the query evaluation cost, while meeting the query response time requirements. We first formulated this problem as the QoS-aware data replication and placement problem for query evaluation of big data analytics. We then proposed a fast yet scalable heuristic by reducing the problem to the unsplittable multicommodity flow problem. We finally evaluated the performance of the proposed algorithm through experimental simulations. Simulation results demonstrate that the proposed algorithm is promising.

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] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica. BlinkDB: queries with bounded errors and bounded response times on very large data. *Proc. EuroSys'13*, IEEE, 2013.

[3] A. Beloglazov, J. Abawajy, and R. Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *J. of Future Generation Computer Systems*, Vol. 28, No. 5, pp.755-768, 2012.

[4] 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.

[5] S. Chaudhuri, G. Das, and V. Narasayya. Optimized stratified sampling for approximate query processing. *Tran. on Database Systems*, Vol. 32, No. 2, pp.1-50, 2007.

[6] W. Fan, F. Geerts, and F. Neven. Making queries tractable on big data with preprocessing: through the eyes of complexity theory. Proceedings of the VLDB Endowment, Vol.6, No. 9, pp.685-696, 2013.

[7] L. Golab, M. Hadjieleftheriou, H. Karloff, and B. Saha. Distributed data placement to minimize communication costs via graph partitioning. *Proc. of SSDBM*, ACM, 2014.

[8] B. Heintz, A. Chandra, and R. K. Sitaraman. Trading Timeliness and Accuracy in Geo-Distributed Streaming Analytics *Proc. of SoCC*, ACM, 2016.

[9] 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.

[10] R. Lu, H. Zhu, X. Liu, J.K. Liu, and J. Shao. Toward efficient and privacy-preserving computing in big data era. *Network, IEEE*, Vol. 28, No. 4, pp.46-50, 2014.

[11] 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.

[12] S. Rao, R. Ramakrishnan, A. Silberstein, M. Ovsiannikov, and D. Reeves. Sailfish: a framework for large scale data processing. *Proc. of SoCC*, ACM, 2012.

[13] 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.

[14] Q. Xia, W. Liang, and Z. Xu. Data locality-aware query evaluation for big data analytics in distributed clouds. To appear in *Computer Journal*, 2017.

[15] 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.

[16] Z. Xu and W. Liang. Minimizing the operational cost of data centers via geographical electricity price diversity, *Proc. of 6th IEEE International Conference on Cloud Computing*, IEEE, 2013.

[17] 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.

[18] 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.

[19] Y. Yan, L. J. Chen, and Z. Zhang. Error-bounded sampling for analytics on big sparse data. *Proc. of the VLDB Endowment*, Vol.7, No.13, pp.1508-1519, 2014.