Advance Access publication on 1 March 2017
doi:10.1093/comjnl/bxw101

# Data Locality-Aware Big Data Query Evaluation in Distributed Clouds

QIUFEN XIA, WEIFA LIANG* AND ZICHUAN XU

*Research School of Computer Science, The Australian National University, Canberra, ACT 2601, Australia*
**Corresponding author: wliang@cs.anu.edu.au*

**With more and more businesses and organizations outsourcing their IT services to distributed clouds for cost savings, historical and operational data generated by the services have been growing exponentially. The generated data that are referred to as big data, stored at different geographic datacenters, now become an invaluable asset to these businesses and organizations, as they can make use of the data through analysis to identify business advantages and make strategic decisions. Big data analytics thus has been emerged as a main research topic in cloud computing. To efficiently evaluate a big data analytic query in a distributed cloud consisting of multiple datacenters at different geographic locations interconnected by the Internet, it poses great challenges: (i) the source data of the query typically are located at different datacenters; and (ii) the resource demands of the query may be beyond the supplies of any single datacenter at that moment. In this paper, we formulate an online query evaluation problem for big data analytic queries in distributed clouds, with an objective to maximize the query acceptance ratio while minimizing the accumulative query evaluation cost, for which we first propose a novel metric to model the usages of different resources in the distributed cloud, by incorporating the capacities and workloads of different datacenters and links, as well as resource demands of different queries. We then devise efficient online algorithms for query evaluations under both unsplittable and splittable source data assumptions. We finally conduct extensive experiments by simulations to evaluate the performance of the proposed algorithms. Experimental results demonstrate that the proposed algorithms are promising, and outperform other heuristics at 95% confidence intervals.**

*Keywords: query evaluation optimization; minimum cost multicommodity flow; big data analytics; data locality; distributed clouds*

*Received 28 September 2015; revised 28 August 2016; editorial decision 21 November 2016*
Handling editor: Joemon Jose

## 1. INTRODUCTION

With the advances in information and communication technologies, various data are generated at exponential rates. For example, there are 4.5 quintillion bytes of data generated daily (IBM 2015) [24], 90% of which have been created in the last 2 years. Big data have emerged as a strategic property of nations and organizations, and there are driving needs to generate values from these data. Big data analytics that is a practice of rapidly crunching big data to identify interesting patterns and improve business strategies, has become a rapidly evolving field in the technology-driven business world [23, 45]. Private and public organizations are eagerly waiting to collect the promised results from such data analysis. Moreover, as data pile up, efficiently managing and analyzing the data become crucial in creating competitive advantage, answering science questions and making effective decisions.

Evaluating queries of big data analytics requires large quantity of storage, computing and network resources that can be met by cloud computing platforms [46]. Cloud computing has emerged as the main computing paradigm in the 21st century [3, 15, 42–45], by providing a plethora of cloud services, including online shopping, data analysis and IT service outsourcing. The current *de facto* architecture of cloud computing, i.e. the centralized datacenters, has demonstrated the limited success in big data analytics, e.g. MapReduce and Hadoop. However, to meet ever-growing resource demands by users, the centralized datacenters are built larger and larger, consuming more and more electricity, thus this is not

eco-sustainable. In contrast, the distributed cloud, consisting of many small- and medium-sized datacenters located at different geographic regions and interconnected by high-speed communication links, has been envisioned as the premier architecture of the next-generation computing platform [2].

Query evaluation for big data analytics in distributed clouds typically requires lots of computing, storage and communication resources across multiple datacenters. However, such resource demands may be beyond the supplies of any single datacenter at that moment. In addition, such query evaluation may incur huge communication cost among the datacenters, by replicating the source data of the query from their datacenters to the datacenters where the query will be evaluated. To efficiently evaluate a query for big-data analytics in a distributed cloud, two important issues must be addressed: one is to identify a set of datacenters with sufficient computing and storage resources to meet the resource demands of the query evaluation; another is to minimize the communication cost of query evaluation, as large quantity of data transfers among datacenters during the query evaluation are needed, and the bandwidth availability between different datacenters significantly varies over time which usually is the bottleneck of such an evaluation [3, 16, 39]. To motivate our study, we here use an example to illustrate the query processing for big data analytics in a distributed cloud (see Fig. 1), where there is a query whose source data are located at two datacenters, $v_1$ and $v_2$, respectively. A naive evaluation plan for the query is to replicate its source data from one datacenter to another, e.g. from $v_1$ to $v_2$, or from $v_2$ to $v_1$, and then evaluate the query at $v_2$ or $v_1$, as shown in Fig. 1(a). This evaluation plan however may not be feasible if neither $v_1$ nor $v_2$ has enough available computing and storage resources to meet the resource demands of the query. A better solution is to find an ideal datacenter $v_3$ with sufficient computing and storage resources that is not far away from both of them, as depicted in Fig. 1(b). Unfortunately, it is very likely that such an ideal datacenter may not exist when all datacenters are working at their high workloads at this moment. To respond to the query on no time, sometimes multiple datacenters must be employed so that their aggregate available resources can meet the resource demands of the query (e.g. $v_3$ and $v_4$ are employed). Thus, the available communication bandwidth between $v_3$ and $v_4$, $v_1$ and $v_4$ and $v_2$ and $v_3$ will be crucial in order to meet the Service Level Agreement (SLA) requirement (the response time requirement) of the query, as shown in Fig. 1(c). Notice that, to reduce the cost of data transfer in the network, we need to deal with data transfer within the network carefully. Specifically, for each big data analytic query, we first try to move query analytics to the datacenter hosting the source data of the query. Only when the hosting datacenter does not have enough available computing resource to evaluate the query at the moment (although its computing capacity may meet the resource demands while the resource is being occupied by existing jobs), the source data will be duplicated to other datacenters for processing.

Motivated by the mentioned example in Fig. 1, in this paper, we deal with online query evaluation for big data analytics in a distributed cloud. That is, for a given monitoring period, user queries of big data analytics arrive into the system one by one, the source data of each query are located at different datacenters in the distributed cloud [35, 39, 40] and need to be replicated to the other datacenters with enough computing resource for its evaluation. We here consider two different types of source data transfers between datacenters: one is that the source data are unsplittable and must be transferred to only one datacenter; another is that the source data are splittable and can be replicated to multiple datacenters. The rationale behind splittable and unsplittable source data lies in that the analysis of big data involves different sorts of data. For example, compressed data (in GZip formats) cannot be retained if they are split into different datacenters and uncompressed separately. Further, data splitting may make the big data analytics deliver useless results in some specific

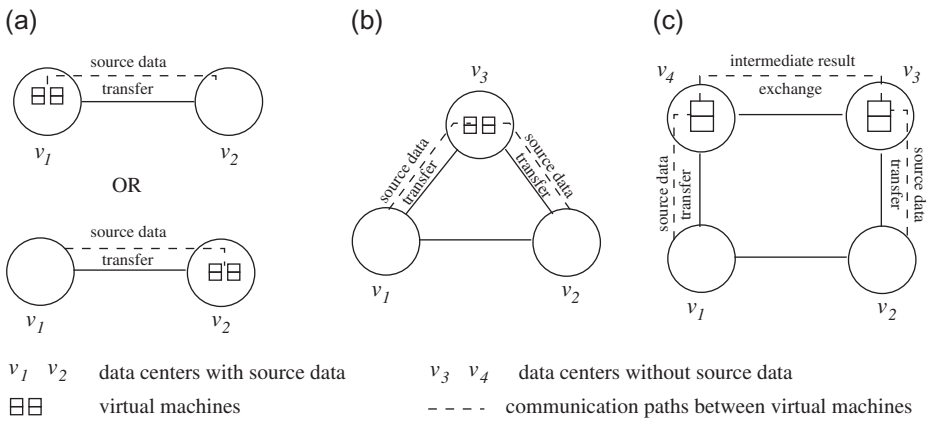


**FIGURE 1.** A motivation example of query evaluation for big data analytics.

applications due to high correlations between different segments of the data, such as stock-price prediction on massive data containing various correlated observations and variables. On the other hand, the analysis based on the text data can be partitioned into different blocks and each block can be analyzed in parallel. The online query evaluation problem under both splittable and unsplittable source data assumptions is to admit as many big data analytic queries as possible (i.e. the query acceptance ratio) as long as there are enough resources to support the evaluations of the admitted queries, while minimizing the accumulative communication cost of the admitted queries, where the accumulated communication cost is defined as the total communication cost incurred by source data replicating and intermediate data transferring for each of the queries over the entire monitoring period.

Although extensive studies on query evaluation in clouds have been taken in the past several years [1, 5, 12, 18, 22, 27, 28, 32, 36–38, 48], most of them focused mainly on minimizing the computing cost [5, 27, 32], storage cost [32], the server running cost [22] or the response time in a single datacenter [28], little attention had been paid to the communication cost when replicating source data and exchanging intermediate results of queries among datacenters, not to mention the impact of the source data locality on the cost of query evaluation. Despite that some of the studies [6, 9, 34, 38, 41, 48] considered the data locality issue, they focused only on a single datacenter, not on multiple datacenters at different geographic locations. In contrast, in this paper, we consider query evaluation for big data analytics in a distributed cloud.

The main contributions of this paper are summarized as follows. We first propose a novel metric to model the consumptions of computing, storage and network resources in a distributed cloud. We then devise online evaluation algorithms for queries of big data analytics in the distributed cloud with the aim to maximize the query acceptance ratio while keeping the accumulative evaluation cost minimized, under the assumptions of the source data being either splittable or unsplittable. We finally conduct experiments by simulations to evaluate the performance of the proposed algorithms. Experimental results demonstrate that the proposed algorithms are promising.

The reminder of this paper is organized as follows. Section 2 introduces the system model and problem definition. The online evaluation algorithms for big data analytics are proposed in Sections 3 and 4. The performance evaluation of the proposed algorithm is given in Section 5, followed by the related work in Section 6. The conclusion is given in Section 7.

## 2. PRELIMINARIES

In this section, we first introduce the system model and query evaluation for big data analytics in distributed clouds. We then define the problems precisely.

### 2.1. System model

We consider a distributed cloud $G = (V, E)$ that consists of a number of datacenters located at different geographical locations and interconnected by high speed links, where $V$ and $E$ are the sets of datacenters and high speed links, respectively. Let $v_i$ be a datacenter in $V$ and $e_{i,j}$ a link in $E$ between datacenters $v_i$ and $v_j$. Denote by $C(v_i)$ and $C(e_{i,j})$ the computing and bandwidth resource capacities of $v_i \in V$ and $e_{i,j} \in E$, respectively. Assume that each datacenter $v_i \in V$ operates in an Infrastructure-as-a-Service environment to lease its virtualized resources (virtual machines) to users, and each link $e_{i,j}$ has bandwidth resource for lease too [43]. Since query evaluation for big data analytics usually is both computing and bandwidth intensive, to evaluate the query, the computing resource in datacenters and the communication bandwidth on links between inter-datacenters must meet its resource demands. Following existing studies [21, 37, 48], we assume that the computing resource demand of each query is given in advance, represented by the number of virtual machines (VMs). Even if a query does not specify its demanded number of VMs, the demand can be derived through analyzing its historic evaluations or the demand of other similar query evaluations, by offline predictions and online calibrations. By referring to the amount of data processed by a VM as the *data chunk size*, we further assume that the data processing rate of a VM is given. Notice that the data processing rate of a VM for IO-intensive operations may be easy to obtain, while the data processing rate of a VM for CPU-intensive operations is hard to get. Since profiling the data processing rates of VMs for different types of operations is out of the scope of this paper, we thus assume that the data processing rate of a VM is given and fixed.

In the rest of this paper, we assume that time is slotted into equal *time slots*, the resources in $G$ are scheduled at the beginning of each time slot. The amounts of available resources of $v_i \in V$ and $e_{i,j} \in E$ at different time slots may be significantly different, depending on their workloads. Denote by $B(v_i, t)$ the amount of the available computing resource in datacenter $v_i$ and $B(e_{i,j}, t)$ the amount of available communication bandwidth on link $e_{i,j}$ at the beginning of time slot $t$. In this paper, we focus on the inter-datacenter communications (bandwidth consumptions) between datacenters, while ignoring the intra-datacenter communications within each datacenter, as the former usually is the bottleneck in such query evaluations [8]. We here assume that the global information of all datacenters in the distributed cloud can be monitored by a *hypervisor*, data transfers among datacenters can be executed asynchronously, and the synchronization can be carried out at some certain stages of the query evaluation. Such monitoring can be implemented by the Software-Defined Networking (SDN) techniques through a centralized SDN controller.

### 2.2. Query evaluation

Given a query $Q$ for big data analytics with its source data located at multiple datacenters in $G$, let $V_Q \subseteq V$ be the set of datacenters in which the source data are located, and $S(v_i, Q)$ the size of the source data of $Q$ at datacenter $v_i \in V_Q$. The evaluation of query $Q$ usually involves not only source data replication to datacenters with enough resources but also intermediate result migrations among the datacenters. This means that the datacenters in which the query will be evaluated should be close to each other for intermediate result migrations and they should not be far away from datacenters in $V_Q$ to reduce the communication cost of data replication. The evaluation of query $Q$ therefore consists of two stages: identify a set $V_P$ of datacenters that are close to each other to meet the resource demands of $Q$; and choose a subset $V_S \subseteq V_P$ of datacenters such that the achieved communication cost of evaluating query $Q$ is minimized.

The *communication cost* of evaluating $Q$ is the sum of the communication cost incurred by replicating its source data from the datacenters in $V_Q$ to the datacenters in $V_S$ and the communication cost between the datacenters in $V_S$, due to intermediate result exchanges.

To replicate the source data from a datacenter in $V_Q$ to another datacenter in $V_S$ or migrate intermediate results between two datacenters in $V_S$, a routing path between the two datacenters must be built if the source data is unsplittable; otherwise, multiple paths may be identified. Let $p_{i,k}$ be a routing path in $G$ between a pair of datacenters $v_i$ and $v_k$, and $\rho_{i,k}$ the portion of $v_i$'s source data that is routed to $v_k$. The communication cost incurred by transferring source data $S(v_i, Q)$ from $v_i \in V_Q$ to $v_k \in V_S$ along $p_{i,k}$ is $S(v_i, Q) \cdot \rho_{i,k} \cdot c(p_{i,k})$, where $c(p_{i,k}) = \sum_{e \in p_{i,k}} c(e)$ is the cost of replicating a unit of data along $p_{i,k}$, and $e$ is a link in $p_{i,k}$ [47, 48]. Notice that the choice of $p_{i,k}$ will be dealt with later. The intermediate results generated at each datacenter $v_k \in V_S$ may need to migrate to the other datacenters in $V_S$ for the sake of query evaluation. Let $I(v_k, Q)$ be the size of the intermediate result of $Q$ in $v_k \in V_S$, the communication cost incurred is $(I(v_k, Q) + I(v_l, Q)) \cdot c(p_{k,l})$ by exchanging its intermediate result with the one in another datacenter $v_l \in V_S$ via a routing path $p_{k,l}$, where $c(p_{k,l}) = \sum_{e \in p_{k,l}} c(e)$ is the accumulative cost of replicating a unit of data via each edge $e \in p_{k,l}$. Denote by $\Gamma_Q$ the communication cost of evaluating query $Q$, then,

$$\Gamma_Q = \sum_{v_i \in V_Q} \sum_{v_k \in V_S} S(v_i, Q) \cdot \rho_{i,k} \cdot c(p_{i,k}) + \sum_{v_k, v_l \in V_S} (I(v_k, Q) + I(v_l, Q)) \cdot c(p_{k,l}), \quad (1)$$

where the first item in the right-hand side of Eq. (1) is the sum of communication costs between the datacenters containing source data and the datacenters processing query $Q$, while the second item is the communication cost among the datacenters in $V_S$ by exchanging intermediate results of $Q$.

We consider a monitoring period that consists of $T$ time slots. Queries may arrive at any time slot within the monitoring period, they will be either admitted or rejected at the beginning of each time slot. A rejected query can be put back to the waiting queue and treated as a 'new query' in the next time slot. Let $\Delta Q(t)$ be the set of queries arrived between time slots $t-1$ and $t$, and $\Delta A(t)$ the set of admitted queries by the system at each time slot $t$ with $1 \leq t \leq T$. Denote by $r(T)$ the *query acceptance ratio* for a monitoring period $T$, which is the ratio of the number of admitted queries to the number of arrived queries during this monitoring period, i.e.

$$r(T) = \frac{\sum_{t=1}^{T} |\Delta A(t)|}{\sum_{t=1}^{T} |\Delta Q(t)|}. \quad (2)$$

The *accumulative communication cost* of evaluating all admitted queries for a period of $T$, $\Gamma(T)$, is thus

$$\Gamma(T) = \sum_{t=1}^{T} \sum_{Q \in \Delta A(t)} \Gamma_Q. \quad (3)$$

### 2.3. Problem definitions

Given a distributed cloud $G = (V, E)$ and a monitoring period $T$, a sequence of queries for big data analytics arrives one by one without the knowledge of future arrivals. Assume that for each query $Q$, the computing resource demand $R(Q)$ (the number of VMs required by it) and its source data set $V_Q$ ($\subseteq V$) are given in advance, the *data locality-aware online query evaluation problem with unsplittable source data* in $G$ for a monitoring period $T$ is to deliver an query evaluation plan for each admitted query, such that the query acceptance ratio $r(T)$ is maximized, while the accumulative communication cost $\Gamma(T)$ is minimized. Similarly, the *data locality-aware online query evaluation problem with splittable source data* in $G$ for a monitoring period $T$ is to deliver a query evaluation plan for each admitted query that its source data are allowed to be split and distributed into multiple datacenters, such that the query acceptance ratio $r(T)$ is maximized, while minimizing the accumulative cost $\Gamma(T)$.

The data locality-aware online query evaluation problems with unsplittable and splittable source data are NP-hard through simple reductions from two NP-hard problems—the unsplittable minimum cost multi-commodity problem [29] and the minimum cost multi-commodity flow problem [11], respectively. For example, we can reduce the unsplittable minimum cost multi-commodity problem to the data locality-

aware online query evaluation problem with unsplittable source data as follows. Consider a special case of the data locality-aware online query evaluation problem with unsplittable source data, i.e. each link has infinity bandwidth resource and the query evaluation has no intermediate result exchanges. A virtual sink $t_0$ is added to the distributed cloud, and there is a link between each datacenter and $t_0$. Evaluating a query in this special case is exactly the unsplittable minimum-cost multi-commodity flow problem, which is to route its required source data (commodities) into their common sink $t_0$, where the communication cost corresponds to the path cost of routing the commodities from their sources to sink $t_0$. Since unsplittable minimum cost multi-commodity problem is NP-Hard [29], the data locality-aware online query evaluation problem with unsplittable source data is NP-Hard too. Similar reduction techniques can be applied to the data locality-aware online query evaluation problem with splittable source data.

The symbols used in this paper are summarized in Table 1.

## 3. ALGORITHM WITH UNSPLITTABLE SOURCE DATA

In this section, we devise an efficient evaluation algorithm for the data locality-aware online query evaluation problem with unsplittable source data. The algorithm is executed at the beginning of each time slot $t$. For each arrived query $Q \in \Delta Q(t)$, the algorithm first checks whether the available VMs (computing resource) of datacenters in the distributed cloud can meet its VM demands. If yes, the query will be processed; otherwise it is rejected. A rejected query can be sent back to the query waiting pool as a new query for scheduling in the next time slot. In the following, we deal with the evaluation of query $Q$ in two stages (3.2 and 3.3).

### 3.1. Overview

Evaluating a query includes examining its source data that is distributed in multiple datacenters and exchanging intermediate results among different datacenters. One key to the evaluation is how to select a set of datacenters that have not only enough computing resource to analyze its source data, but also abundant bandwidth resources to exchange intermediate results and transfer source data when the selected datacenters do not have its source data. The basic idea of the proposed algorithm is to first find a set of potential datacenters that have enough computing resource and the links between them have enough bandwidth resource to exchange intermediate results. The algorithm then finds a subset of the potential datacenters that are 'close' to the datacenters that store the source data to reduce the cost incurred by source data migrating.

**TABLE 1.** Symbols.

| Symbols | Meaning |
|---|---|
| $G$ | The distributed cloud |
| $V$ | Set of datacenters in the distributed cloud $G$ |
| $E$ | Set of links among datacenters |
| $v_i$ | A datacenter in $V$ |
| $e_{i,j}$ | A link in $E$ between datacenters $v_i$ and $v_j$ |
| $C(v_i)$ | Computing resource capacity of a datacenter $v_i$ |
| $C(e_{i,j})$ | Bandwidth resource capacity of a link $e_{i,j}$ |
| $T$ | The monitoring period |
| $t$ | The current time slot $t$ |
| $B(v_i, t)$ | Amount of available computing resource in datacenter $v_i$ at time slot $t$ |
| $B(e_{i,j}, t)$ | Amount of available communication bandwidth on link $e_{i,j}$ at time slot $t$ |
| $B(G, t)$ | Total available computing resource of $G$ at time slot $t$ |
| $B(V_P, t)$ | Total available computing resource of all datacenters in $V_P$ at time slot $t$ |
| $Q$ | A big data analytic query |
| $V_Q$ | Set of datacenters in which the source data of $Q$ is located |
| $S(v_i, Q)$ | The size of the source data of $Q$ at datacenter $v_i \in V_Q$ |
| $p_{i,k}$ | A routing path in $G$ between a pair of datacenters $v_i$ and $v_k$ |
| $\rho_{i,k}$ | The portion of $v_i$'s source data that is routed to $v_k$ |
| $c(p_{i,k})$ | Cost of replicating a unit of data along path $p_{i,k}$ |
| $e$ | A link in $G$ |
| $I(v_k, Q)$ | The size of the intermediate result of $Q$ generated at datacenter $v_k$ |
| $\Gamma_Q$ | The communication cost of evaluating a query $Q$ |
| $\Delta Q(t)$ | Set of queries arrived between time slots $t-1$ and $t$ |
| $\Delta A(t)$ | Set of admitted queries by the system at each time slot $t, 1 \leq t \leq T$ |
| $r(T)$ | The *query acceptance ratio* for a monitoring period $T$ |
| $\Gamma(T)$ | The accumulative communication cost of evaluating all admitted queries for a period of $T$ |
| $V_P$ | Set of potential datacenters to evaluate a query |
| $V_S$ | Set of datacenters to evaluate a query, $V_S \subseteq V_P$ |
| $R(Q)$ | The computing resource demand of a query $Q$ |
| $\Phi(v_i, t)$ | The datacenter metric of a datacenter $v_i$ at time slot $t$ |
| $a$ | a constant with $a > 1$ |
| $\Psi(e_{i,j}, t)$ | The link metric of a link $e_{i,j}$ at time slot $t$ |
| $b$ | a constant with $b > 1$ |
| $d(e_{i,j}, t)$ | Length of a link $e_{i,j}$ at time slot $t$, $d(e, t) = \frac{1}{\Psi(e,t)}$ if $\Psi(e, t) > 0$ |
| $NR(v_i, t)$ | The rank of a datacenter $v_i$ at time slot $t$ |
| $L(v_i)$ | Set of links incident to $v_i$ in $G$ |
| $v_c$ | The 'center' of the set $V_P$ of datacenters |
| $G' = (V', E')$ | An auxiliary directed flow graph |
| $t_0$ | A virtual sink node |

### 3.2. Identification of a set $V_P$ of potential datacenters

To identify a set of datacenters that meets the VM demands of query $Q$, a metric measuring the usage cost of computing resource among the datacenters is needed. Such a metric should take into account not only the quantity of available computing resource but also the utilization ratio of the resource at each datacenter. Typically, the computing ability of a datacenter $v_i$ decreases with the increase of its utilization ratio, as a datacenter with high resource utilization has a higher probability of violating user resource demands or SLAs, i.e. a datacenter with more available computing resource and low utilization ratio is a good candidate to evaluate a query $Q$. The computing ability of datacenter $v_i$ thus is modeled as its *datacenter metric*, denoted by $\Phi(v_i, t)$ at time slot $t$, then

$$\Phi(v_i, t) = B(v_i, t) \cdot a^{\frac{B(v_i,t)}{C(v_i)}}, \tag{4}$$

where $a$ is a constant with $a > 1$ that reflects the weighting in which degree the usage cost of a resource is, $B(v_i, t)$ is the amount of available computing resource of $v_i$, and $\frac{B(v_i,t)}{C(v_i)}$ is the utilization ratio of computing resource of $v_i$. A higher $\Phi(v_i, t)$ means that $v_i$ has more available computing resource and a lower usage cost of the resource, and has a higher probability to become a potential processing datacenter for query $Q$. Similarly, the *link metric* $\Psi(e_{i,j}, t)$ of a link $e_{i,j}$ between two datacenters $v_i$ and $v_j$ at time slot $t$ is defined by

$$\Psi(e_{i,j}, t) = B(e_{i,j}, t) \cdot b^{\frac{B(e_{i,j},t)}{C(e_{i,j})}}, \tag{5}$$

where $b > 1$ is a similar constant, and $B(e_{i,j}, t)$ is the available bandwidth resource of link $e_{i,j}$ at time slot $t$. The arguments for ratio $\frac{B(e_{i,j},t)}{C(e_{i,j})}$ and $\Psi(e_{i,j}, t)$ are similar as the ones for $\frac{B(v_i,t)}{C(v_i)}$ and $\Phi(v_i, t)$.

The allocated VMs for evaluating query $Q$ require communications with each other in order to exchange their intermediate results. This can be implemented through building multiple routing paths between the datacenters accommodating the VMs. To find a cheaper routing path $p$, the 'length' of path $p$ is defined as the sum of lengths of links in $p$. Let $d(e, t)$ be the length of link $e$ in $p$ at time slot $t$, if $\Psi(e, t) > 0$, then $d(e, t) = \frac{1}{\Psi(e,t)}$; $d(e, t) = \infty$ otherwise. This implies that the shorter the length of a link, the more available bandwidth it has.

Having defined the cost metrics of resource usages in a distributed cloud, we now identify a set $V_P$ of potential datacenters for evaluating query $Q$. Notice that, to enable efficient intermediate result exchanges during the evaluation of query $Q$, such a set of datacenters should be 'close' to each other, and the links interconnecting them should have both enough available bandwidth resource and low utilization. Thus, to find such a set of datacenters, we first identify the 'center' of the set $V_P$ by assigning each datacenter $v_i \in V$ *a rank*, $NR(v_i, t)$, that is *the product* of datacenter metric $\Phi(v_i, t)$ of $v_i$ and the accumulative metric of links incident to $v_i$, i.e.

$$NR(v_i, t) = \Phi(v_i, t) \cdot \sum_{e_{i,j} \in L(v_i)} \Psi(e_{i,j}, t), \tag{6}$$

where $L(v_i)$ is the set of links incident to $v_i$ in $G$. The rationale behind Eq. (6) is that the more available computing resources a datacenter $v_i$ has, the more available accumulative bandwidth of links incident to it, and the higher rank the datacenter $v_i$ will have. A datacenter with the highest rank will be selected as the 'center' of the set of datacenters $V_P$, denoted by $v_c$. If the available computing resource $\Phi(v_c, t)$ of $v_c$ cannot meet the resource demands of query $Q$, the next datacenter will be chosen and added to $V_P$ greedily. Specifically, each datacenter $v_i \in V \backslash V_P$ is assigned a rank by the product of the inverse of $\Phi(v_i, t)$ and the accumulative shortest length from $v_i$ to all the selected datacenters $v_j \in V_P$, i.e.

$$\frac{1}{\Phi(v_i, t)} \cdot \sum_{v_j \in V_P} \sum_{e_{i,j} \in p_{i,j}} d(e_{i,j}, t), \tag{7}$$

where $p_{i,j}$ is the one with the minimum accumulative length of links, i.e. the sum of costs of links between $v_i \in V \backslash V_P$ and each selected datacenter $v_j \in V_P$ is the smallest one. The datacenter with the smallest rank is chosen and added to $V_P$. This procedure continues until the accumulative computing resource of all chosen datacenters in $V_P$ meets the demanded number of VMs of query $Q$.

### 3.3. Selecting a subset $V_S$ of $V_P$

Recall that the source data of query $Q$ in datacenter $v_i \in V_Q$ will be replicated to another datacenter for the query evaluation, i.e. we assume that this source data cannot be split and replicated to multiple datacenters for independent processing. Such an assumption is purely for the sake of simplicity of discussion, which will remove this assumption in Section 4. Stage 2 of the evaluation algorithm is to identify a subset $V_S$ of $V_P$ to reduce the communication cost between the datacenters hosting the source data and the datacenters performing the query evaluation. To this end, we reduce the problem of selecting datacenters in $V_P$ and routing paths in this stage into an unsplittable minimum cost multi-commodity flow problem in an auxiliary directed flow graph $G' = (V', E')$ whose construction is as follows.

A *virtual sink node* $t_0$ and all datacenter nodes in $G$ are added to $G'$, i.e. $V' = V \cup \{t_0\}$. There is a directed link from each node $v_j \in V_P$ to the virtual sink node $t_0$. The capacity of edge $e_{v_j,t_0}$ is the volume of source data that $v_j$ can process at time slot $t$, and its cost is set to zero. Given a pair of datacenters, there are two directed edges in $G'$ between them if there is an edge in $G$ between them. The capacity of each edge in $E' \backslash \{\langle v_j, t_0\rangle | v_j \in V_P, V_P \subseteq V\}$ is set to the total volume of

source data of query $Q$, and the cost of each such edge is set to the communication cost by replicating a unit of source data along it. The source data of query $Q$ in each datacenter $v_i \in V_Q$ are treated as *a commodity* with demand $S(v_i, Q)$ at a source node in $G'$, which will be routed to the destination node $t_0$ through a potential datacenter $v_j \in V_P$, where the potential datacenter $v_j$ will be added into $V_S$ in which the source data of $Q$ will be migrated and evaluated. Specifically, to find a feasible solution in $G'$, we first find a shortest routing path in terms of link cost for each commodity from the source node $v_i \in V_Q$ to a datacenter $v_j \in V_P$. We then calculate the ratio of the shortest path cost to the source data size $|S(v_i, Q)|$. We finally route the commodity from $v_i$ to the datacenter $v_j$ along a routing path with the minimum ratio.

Figure 2 uses an example to illustrate the construction of the flow graph $G'$, where the set of source data locations of query $Q$ is $V_Q = \{v_2, v_3, v_4, v_5\}$. Due to insufficient computing resource of datacenters in $V_Q$, the source data of $Q$ have to be migrated to other set $V_P$ of potential datacenters with sufficient computing resource, where $V_P = \{v_7, v_8, v_9, v_{10}, v_{11}, v_{12}\}$. Each source data in $V_Q$ is treated as a commodity, which will be routed to a virtual destination node $t_0$ through a potential datacenter in $V_p$, and this datacenter will be added into $V_S \subseteq V_P$. Here, $V_S = \{v_8, v_9, v_{12}\}$. This procedure continues until all commodities are successfully routed. The detailed algorithm is described in `Algorithm 1`.

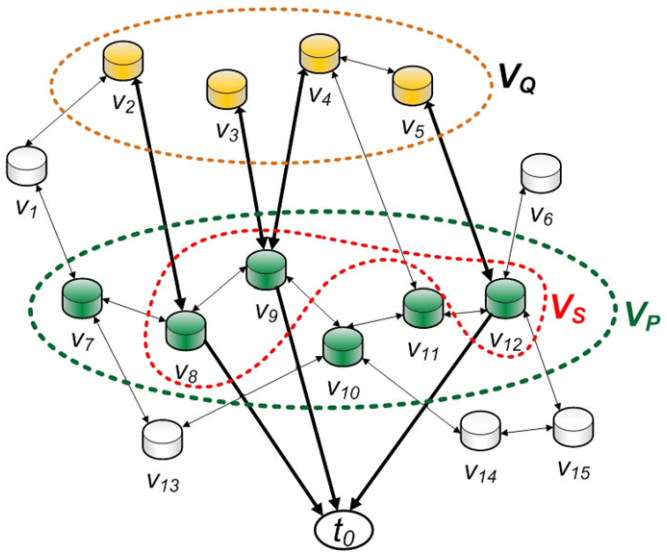


**FIGURE 2.** An example of the auxiliary directed flow graph $G'$ in Stage 2 of Algorithm 1, where $V_Q$ is the set of datacenters in which the source data of query $Q$ are located, $V_P$ is a set of potential datacenters for evaluating $Q$, $V_S \subseteq V_P$ is the set of datacenters for the evaluation of $Q$, and the highlighted edges are the edges via which data are replicated to their destination datacenter. Notice that $V_P$ is identified by Stage 1 of Algorithm 1, and $V_S$ ($\subset V_P$) is computed in its Stage 2.

THEOREM 3.1. *Given a distributed cloud $G = (V, E)$ for a given monitoring period of $T$ time slots, assume that queries for big data analytics arrive one by one without future arrival knowledge. There is an online algorithm, i.e.* `Algorithm 1`, *for the data locality-aware online query evaluation problem with unsplittable source data, which takes* $\sum_{t=1}^{T} O(|\Delta Q(t)| \cdot (|V|^3 \log|V| + |V|^2 \cdot |E|))$ *time, where* $\Delta Q(t)$ *is a set of arrived queries during each time slot* $t$, $1 \leq t \leq T$.

*Proof.* To evaluate each query $Q \in \Delta Q(t)$, `Algorithm 1` consists of two stages: identifying a set $V_P$ of potential datacenters that not only have enough computing resource but also are interconnected by links with abundant bandwidth resource; and selecting a subset $V_S$ of $V_P$ to evaluate query $Q$.

In Stage 1, `Algorithm 1` identifies a cluster $V_P$ of potential datacenters according to the defined node and link metrics. Specifically, it first selects a datacenter with the highest rank as defined in Eq. (6), which takes $O(|V|)$ time. It then iteratively adds datacenters into the cluster one by one until the cluster has enough computing resource to evaluate the query, and this procedure takes $O(|V|^2)$ time. Stage 1 thus takes $O(|V|^2)$ time.

Stage 2 of the algorithm finds a subset $V_S$ of $V_P$ as the datacenters to evaluate query $Q$ by transferring the problem into an unsplittable minimum cost multi-commodity flow problem in an auxiliary graph $G' = (V', E')$. Clearly, the construction of $G'$ takes $O(|V| + |E|)$ time. Then, the algorithm routes each commodity in the auxiliary graph by selecting a commodity $S(v_i, Q)$ with the minimum ratio of $\min_{v_i \in V} \frac{S(v_i, Q)}{\sum_{e \in p_{i,t_0}} c(e)}$, where $p_{i,t_0}$ is the shortest path between $v_i$ and $t_0$. This takes $O(|V|^2 \log|V| + |V| \cdot |E|)$ time to find all pairs of shortest paths in $G$ for the $|V|$ commodities. The total amount of time of routing all commodities is $O(|V| \cdot (|V|^2 \log|V| + |V| \cdot |E|)) = O(|V|^3 \log|V| + |V|^2 \cdot |E|)$. There are $|\Delta Q(t)|$ queries at time slot $t$, `Algorithm 1` thus takes $O\big(|\Delta Q(t)| \cdot (|V|^3 \log|V| + |V|^2 \cdot |E|)\big)$ time at time slot $t$.

The total amount of time taken by the online algorithm for the monitoring period $T$ thus is $\sum_{t=1}^{T} O\big(|\Delta Q(t)| \cdot (|V|^3 \log|V| + |V|^2 \cdot |E|)\big)$. □

## 4. ALGORITHM WITH SPLITTABLE SOURCE DATA

So far, we have assumed that the source data of query $Q$ at each datacenter can only be replicated to a single datacenter. In reality, the source data of many query evaluations can be split and distributed to multiple datacenters. We here devise an efficient evaluation algorithm for the data locality-aware online query evaluation problem with splittable source data, by modifying `Algorithm 1`. Recall that `Algorithm 1` consists of two stages: selecting a set $V_P$ of

**Algorithm 1** Algorithm for the data locality-aware online query evaluation problem with unsplittable source data.

**Input**: The distributed cloud graph $G = (V, E)$, query set $\Delta Q(t)$, the monitoring period consisting of $T$ time slots.
**Output**: The query acceptance ratio and the accumulative communication cost of evaluating admitted queries.

**for** $t \leftarrow 1$ to $T$ **do**
  $Q \in \Delta Q(t)$ is the query being evaluated;
  Get the amount of available computing resources of $G$ at $t$, i.e. $B(G, t)$, and resource demand of $Q$, i.e. $R(Q)$;
  **if** $B(G, t) < R(Q)$ **then**
    $Q$ is rejected;
  **else**
    /* Stage 1 */:
    $V_p \leftarrow \varnothing$;
    Calculate rank $NR(v_i, t)$ for each datacenter $v_i \in V$ according to Eq. (6), and select the datacenter with the maximum rank, i.e. $v_c$;
    $V_P \leftarrow V_P \cup \{v_c\}$;
    Get the total amount of available computing resource of all datacenters in $V_P$ at time slot $t$, i.e. $B(V_P, t)$;
    **while** $B(V_P, t) < R(Q)$ and $V \setminus V_p \neq \varnothing$ **do**
      Find $v_i \in V \setminus V_p$, with minimum value of Eq. (7), $V_p \leftarrow V_P \cup \{v_i\}$;
    /* Stage 2 */:
    $n_Q \leftarrow |V_Q|$; /* the number of commodities of $Q$ */
    **while** $n_Q > 0$ **do**
      Create an auxiliary graph $G' = (V', E')$, $E' \leftarrow E$, and $V' \leftarrow V \cup t_0$, where $t_0$ represents a virtual destination for all commodities of $Q$, i.e. $S(v_i, Q)$, $\forall v_i \in V_Q$;
      **for** each potential datacenter $v_j \in V_P$ **do**
        Add an edge $e_{j,t_0}$ from $v_j$ to $t_0$;
        The cost of $e_{j,t_0}$ is 0, and its capacity is the volume of source data that $v_j$ can process at time slot $t$;
      **for** each edge $e$ in $E'$ **do**
        The capacity of $e$ is set to the total volume of source data of query $Q$;
        The cost of $e$, $c(e)$, is set to the communication cost for replicating one unit of data of $Q$;
      **while** $n_Q > 0$ and there is one datacenter in $V_P$ that can accommodate one of the left commodities **do**
        **for** each commodity $S(v_i, Q)$ **do**
          Find a path $p_{i,t_0}$ from $v_i$ to $t_0$ with minimum accumulated cost of all edges along the path;
          Calculate the ratio $\sum_{e \in p_{i,t_0}} c(e)/S(v_i, Q)$;
        Route the commodity with the minimum ratio $\sum_{e \in p_{i,t_0}} c(e)/S(v_i, Q)$; delete this commodity;
        $n_Q \leftarrow n_Q - 1$;
      Update capacities and costs of edges in $G'$;
    **if** $n_Q > 0$ and $V \setminus V_p \neq \varnothing$ **then**
      Add the datacenter in $V \setminus V_p$ with minimum value of Eq. (7) into $V_P$;
      Update available computing resources of datacenters and bandwidth resources of links in $G$;
    **else**
      **if** $n_Q > 0$ and $V \setminus V_p = \varnothing$ **then**
        $Q$ is rejected;
        **Break**;
    **if** $n_Q = 0$ **then**
      $Q$ is admitted;
      **Return**;
  $\Delta Q(t) \leftarrow \Delta Q(t) \setminus \{Q\}$;
**Return** the query acceptance ratio and the accumulative communication cost of evaluating admitted queries.

potential datacenters, and replicating the source data to a subset $V_S$ of $V_P$. Source data replications usually dominate the query response time due to large amounts of source data and intermediate results to be replicated through the network [25]. We thus devise a fast routing algorithm for the source data replication as follows.

Having selected the set $V_p$ of potential datacenters, we first construct an auxiliary graph $G' = (V', E')$ as follows. A *virtual sink node* $t_0$ and all datacenter nodes in $G$ are added to $G'$, i.e. $V' = V \cup \{t_0\}$. There is a directed link from each node $v_j \in V_P$ to the virtual sink node $t_0$. The capacity of link $e_{v_j,t_0}$ is the amount of source data that $v_j$ can process at time slot $t$, and the cost of the link is set to zero. For each pair of datacenters, there are two directed edges in $G'$ between them if there is a link in $G$ between them. The capacity of each link in $E' \setminus \{\langle v_j, t_0\rangle \mid v_j \in V_P, V_P \subseteq V\}$ is set to the total volume of source data of query $Q$, since all source data of query $Q$ may be routed through an edge in $E' \setminus \{\langle v_j, t_0\rangle \mid v_j \in V_P, V_P \subseteq V\}$, and the cost of each such edge is set to the communication cost by replicating a unit of source data along it. The source data in each datacenter $v_i \in V_Q$ are treated as *a commodity* with demand $S(v_i, Q)$ in $G'$, which will be routed to the destination node $t_0$.

We then reduce the source data replication problem to the minimum cost multi-commodity flow problem in $G'$. A feasible solution to the minimum cost multi-commodity flow problem in $G'$ will return a feasible solution to the source data replication problem in $G$. To speed up the source data replication, we employ a fast approximation algorithm for the minimum cost multi-commodity flow problem, due to Garg and Könemann [13]. Note that the use of Garg and Könemann's algorithm allows us to explore a fine-grained trade-off between the source data replication accuracy and its running time by an appropriate accuracy $\varepsilon$ with $0 < \varepsilon \leq 1/3$. Only the queries whose source data can be fully replicated will be admitted by the system; otherwise, the

**Algorithm 2.** Algorithm for the data locality-aware query evaluation problem with splittable source data.

**Input**: The distributed cloud graph $G = (V, E)$, a set of queries $\Delta Q(t)$, the set of VMs to evaluate each query $Q \in \Delta Q(t)$ at each time slot $t$, the monitoring period consisting of $T$ time slots, the accuracy parameter $\varepsilon$.

**Output**: The query acceptance ratio and the accumulative communication cost of evaluating admitted queries.

**for** $t \leftarrow 1$ to $T$ **do**
  $Q \in \Delta Q(t)$ is the query being evaluated;
  Get the total amount of available computing resources of $G$ at time slot $t$, i.e. $B(G, t)$;
  Get the total amount of computing resource to process all source data of $Q$, i.e. $R(Q)$;
  **if** $B(G, t) < R(Q)$ **then**
    $Q$ is rejected;
  **else**
    /* stage 1 */:
    $V_p \leftarrow \varnothing$; /* the set of potential datacenters to process $Q$ */
    Calculate rank $NR(v_i, t)$ for each datacenter $v_i \in V$ according to Eq. (6), and select the datacenter with the maximum rank, i.e. $v_c$;
    $V_P \leftarrow V_P \cup \{v_c\}$;
    Get the total available computing resources of all datacenters in $V_P$ at time slot $t$, i.e. $B(V_P, t)$;
    **while** $B(V_P, t) < R(Q)$ and $V \setminus V_p \neq \varnothing$ **do**
      Find $v_i \in V \setminus V_p$, with minimum value of Eq. (7), $V_p \leftarrow V_P \cup \{v_i\}$;
    /* stage 2 */:
    Create an auxiliary graph $G' = (V', E')$, $E' \leftarrow E$, and $V' \leftarrow V \cup t_0$, where $t_0$ represents a virtual destination for all commodities of $Q$, i.e. $S(v_i, Q)$, $\forall v_i \in V_Q$;
    **for** each potential datacenter $v_j \in V_P$ **do**
      Add an edge $e_{j,t_0}$ from $v_j$ to $t_0$;
      The cost of $e_{j,t_0}$ is 0, and its capacity is the volume of source data that $v_j$ can process at time slot $t$;
    **for** each edge $e$ in $E'$ **do**
      The capacity of $e$ is set to the total volume source data of query $Q$;
      The cost of $e$, $c(e)$, is set to the communication cost for replicating one unit of source data of $Q$;
    Call Garg and Könemann's algorithm on flow graph $G'$ with $|V_Q|$ commodities;
    **if** All source data of $Q$ are routed **then**
      $Q$ is admitted;
    **else**
      $Q$ is rejected;
  $\Delta Q(t) \leftarrow \Delta Q(t) \setminus \{Q\}$;
**Return** the query acceptance ratio and the accumulative communication cost of all admitted queries.

query will be rejected. The details of the algorithm are described in Algorithm 2.

THEOREM 4.1. *Given a distributed cloud $G = (V, E)$ for a given monitoring period of $T$ time slots, assume that queries for big data analytics arrive into the system one by one without the knowledge of future arrivals, and the source data of each query can be split to multiple datacenters for independent evaluation. There is an online algorithm, i.e.* Algorithm 2, *for the data locality-aware online query evaluation problem with splittable source data, which takes* $\sum_{t=1}^{T} O^*\left(|\Delta Q(t)| \cdot \varepsilon^{-2} |V|^4\right)$ *time,*[1] *where* $\varepsilon$ *is a given accuracy parameter with* $0 < \varepsilon \leq 1/3$.

*Proof.* The difference between Algorithm 2 and Algorithm 1 lies in Stage 2 that selects a subset $V_S$ from the identified set $V_P$ of potential datacenters. Following Theorem 3.1, Stage 1 takes $O(|V|^2)$ time to identify the set of datacenters $V_P$ with sufficient computing resource to meet the VM demands of query $Q \in \Delta Q(t)$. We thus only analyze the time complexity of Stage 2 of Algorithm 2 as follows.

Stage 2 constructs an auxiliary graph $G' = (V', E')$, and transfers the problem of selecting a set $V_S$ of datacenters from $V_P$ to the minimum cost multi-commodity problem by treating the source data of each query as a commodity. The construction of the auxiliary graph $G'$ takes $O(|V| + |E|)$ time, where $|V'| = |V|$ and $|E'| = O(|E|)$. Following Garg and Könemann's algorithm for the minimum cost multi-commodity problem, it takes $O^*(\varepsilon^{-2} M^2 |V'|^2)$ time to route $M$ commodities from their sources to a common destination $t_0$ in the auxiliary graph $G'$, where $|V'| = |V| + 1$. There are $|\Delta Q(t)|$ queries at each time slot $t$, and each query has $|V_Q|$ commodities (source data) to route, $|V_Q| \leq |V|$, i.e. $M = |\Delta Q(t)| \cdot |V_Q|$. Stage 2 of Algorithm 2 takes $O^*(\varepsilon^{-2} |V'|^4) = O^*(\varepsilon^{-2} |V|^4)$ time. Algorithm 2 thus takes $\sum_{t=1}^{T} O^*\left(|\Delta Q(t)| \cdot \varepsilon^{-2} (|V|^4\right)$ time. □

## 5. PERFORMANCE EVALUATION

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

### 5.1. Simulation settings

We consider a distributed cloud $G(V, E)$ consisting of 20 datacenters. There is an edge between a pair of nodes with a probability of 0.2 generated by the GT-ITM tool [17]. The computing capacity of each datacenter and the bandwidth capacity of each link are randomly drawn from value intervals [1000, 3000] GHz, and [100, 1000] Mbps, respectively [4, 14]. The total volume of source data of each query is in the range of [128, 512] GB, which is randomly distributed between 1 and 4 datacenters. Each VM has the computing capacity of 2.5 GHz and can process 256 MB data chunk [31], according to the settings of Amazon EC2 Instances [3]. Parameters $a$ and $b$ are set as $2^4$ and $2^6$ in default settings, respectively. We assume that the monitoring period $T$ is 800 time slots with each time slot lasting 5 minutes. We further assume that the number of queries issued within each time slot is ranged between 5 and 45, and each query evaluation spans between 1 and 10 time slots. According to the on-demand pricing of Amazon CloudFront, users pay only for the contents that are delivered to them through the network without minimum commitments or up-front fees, and the fee charged for transmitting 1 GB data is in the range of [$0.02, $0.06]. Unless otherwise specified, we will adopt these default settings. Each value in the figures is the mean of the results by applying the mentioned algorithm 15 times. Also, 95% confidence intervals for these mean values are presented in all figures.

To evaluate the proposed algorithms, two heuristics are used as evaluation baselines: one is to choose a datacenter with the maximum number of available VMs and then replicate as much source data as possible to the datacenter. If the datacenter cannot meet the query resource demands, it then picks the next datacenter with the second largest number of available VMs. This procedure continues until the VM demands of the query are met. Another is to select a datacenter randomly and places as much source data as possible to the datacenter. If the available number of VMs in the chosen datacenter is not enough to process the query, it then chooses the next one randomly. This procedure continues until the VM demands of the query are satisfied. For simplicity, we refer to the proposed Algorithm 1 for the data locality-aware online query evaluation problem with unsplittable source data, Algorithm 2 for data locality-aware online query evaluation problem with splittable source data, and the two baselines algorithms as algorithms DL-Unsplittable, DL-Splittable, Greedy and Random, respectively.

### 5.2. Algorithm performance evaluation

We first evaluate the performance of the proposed algorithms DL-Unsplittable and DL-Splittable, in terms of the query acceptance ratio, the total accumulative communication cost, the accumulative communication cost for source data replication, and the accumulative communication cost for intermediate result exchanges, where the accumulative communication cost is the average communication cost of all admitted queries during a monitoring time period $T$.

Figure 3(a) plots the curves of query acceptance ratios of DL-Unsplittable, DL-Splittable, Greedy and Random, respectively, from which it can be seen that the query acceptance ratios of algorithms DL-Unsplittable

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

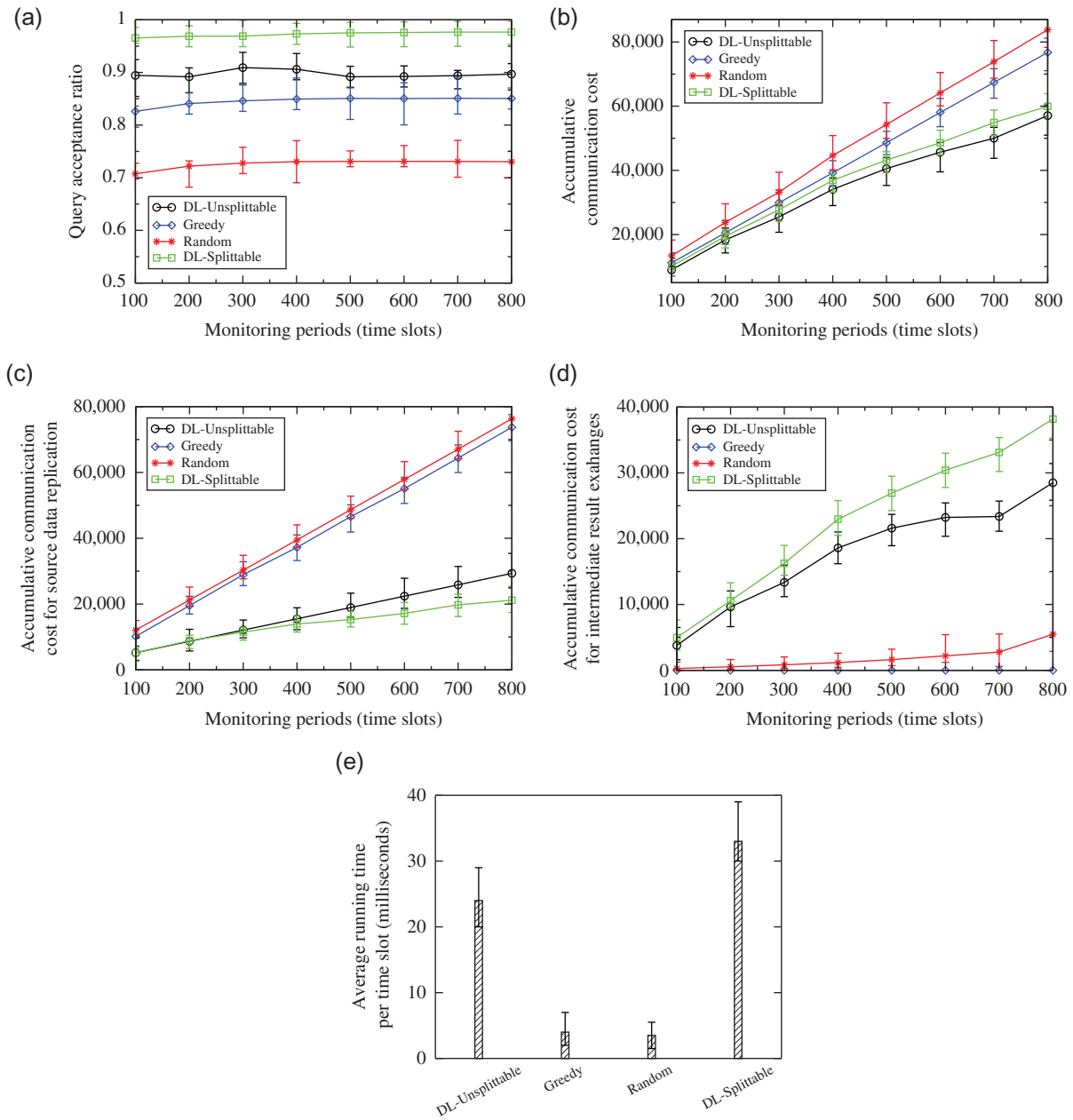


**FIGURE 3.** Performance evaluation of different algorithms. (**a**) The query acceptance ratio of different algorithms. (**b**) The accumulative communication cost of different algorithms. (**c**) The accumulative communication cost for source data replication of different algorithms. (**d**) The accumulative communication cost for intermediate result exchanges of different algorithms. (**e**) The average running times of different algorithms per time slot.

and `DL-Splittable` are much higher than these of the other algorithms, because algorithms `DL-Unsplittable` and `DL-Splittable` consider the data-locality of queries and identify datacenters with enough computing and bandwidth resources. Furthermore, algorithm `DL-Splittable` delivers the highest acceptance ratio among the mentioned algorithms, which is 3%, 11% and 22% higher than those of algorithms `DL-Unsplittable`, `Greedy` and `Random`, respectively. The rationale is that algorithm `DL-Splittable` can move portions of source data through multiple paths to multiple datacenters with the minimum communication cost. The query thus has a higher probability to be accepted by the system, and the total accumulative communication cost for evaluating the query and replicating its source data is reduced, while algorithm `DL-Unsplittable` may not move the source data along a path to a datacenter with the minimum

communication cost, as the datacenter may not meet the computing resource demand of the source data. For algorithms Greedy and Random, the former selects the datacenter with the most computing resource while ignoring the paths from source data to the selected datacenter. This causes a higher communication cost, and the latter randomly chooses a datacenter, neglecting both computing resource and bandwidth resource demands. This may decrease the acceptance ratio and increase the communication cost, as there is no guarantees that the selected datacenters and their links have enough resources for the query evaluation.

Figure 3(b) plots the curves of the accumulative communication costs of the four mentioned algorithms. Clearly, the accumulative communication costs of algorithms Greedy and Random are worse in comparison with these of algorithms DL-Unsplittable and DL-Splittable, which are around 1.6 times that of algorithms DL-Unsplittable and DL-Splittable. The accumulative communication cost of algorithm DL-Splittable is higher than that of algorithm DL-Unsplittable, because it accepts more queries than that of algorithm DL-Unsplittable as shown by Fig. 3(a).

Figure 3(c) depicts the accumulative communication cost of source data replication of different algorithms. It can be seen that algorithm DL-Splittable has the lowest accumulative communication cost. However, as shown by Fig. 3(d), it does incur a higher accumulative communication cost on the intermediate result exchanges than those of other algorithms. The reason is that each source data of algorithm DL-Splittable is split into multiple portions with each moving to a different datacenter, these portions need to exchange their intermediate results to form the final result, which result in the highest cost of exchanging intermediate results. In addition, from Fig. 3(d), it can be seen that algorithm Greedy has almost zero communication cost for intermediate result exchanges, as this algorithm always chooses a datacenter with the maximum available computing resource, which may satisfy the demand of all source data of a query, meaning that all source data of the query may have been moved to one datacenter, and there is no communication incurred by exchanging intermediate results.

Figure 3(e) depicts the running time of different algorithms, from which it can be seen that the running time of algorithm DL-Splittable is the longest one, algorithm DL-Unsplittable takes the second highest running time, while algorithms Greedy and Random spend much less time. Although algorithms DL-Splittable and DL-Unsplittable take more running time than those of the other two algorithms, they however achieve much better performance, i.e. higher query acceptance ratios.

Notice that the query acceptance ratios of different algorithms may drop with the arrivals of large number of queries that demand high quantity of computing resource within very short time, as the accumulative computing resource in the system is limited. This may lead to an unstable system if such queries are rejected frequently due to lack of resources. Although query evaluation in such a scenario is out of the scope of this paper, our solution can be extended to enhance the system stability in this mentioned scenario. That is, a certain fraction of the resources in a distributed cloud can be reserved to handle queries demanding high quantity of resources. The proposed algorithm can be applied to the admissions of such queries, by considering the reserved amounts of computing and bandwidth resources as 'resource capacities', and the reserved amounts can be calculated according to historical arrival patterns of queries. In case there is no resource reserved, resource sharing can be enabled to allow the resources to be shared with other queries.

### 5.3. Impact of the number of datacenters on the performance of different algorithms

We then study the impact of the number of datacenters on the query acceptance ratio, the accumulative communication cost, the accumulative communication cost for source data replication and the accumulative communication cost for intermediate result exchanges of algorithms DL-Unsplittable and DL-Splittable, by varying the number from 10 to 40. For the sake of convenience, we use $n$ to represent the number of datacenters in Figs. 4 and 5.

Figures 4(a) and 5(a) plot the query acceptance ratio curves of algorithms DL-Unsplittable and DL-Splittable, from which it can be seen that the query acceptance ratios first grow with the increase of $n$, and then keep stable after $n = 30$. Specifically, the acceptance ratios grow by 27% and 25% when the value of $n$ increases from 10 to 20 in Figs 4(a) and 5(a), respectively. The reason is that with the increase of the number of datacenters, more and more queries are admitted by the system as more computing resource is available. The query acceptance ratio approaches 100% when $n = 40$. Figure 4(b) illustrates the accumulative communication cost curve of algorithm DL-Unsplittable. As shown in Fig. 4 (b), the accumulative communication cost by algorithm DL-Unsplittable decreases, with the growth of the number of datacenters $n$. This is because that more datacenters mean more routing paths with lower communication costs from source data transfers. In addition, more datacenters also imply the selected datacenters have more computing resource with high bandwidth resource in their links. Figure 4(c) depicts curves of the accumulative communication cost for source data replications of algorithm DL-Unsplittable, from which it can be seen that the accumulative communication cost for source data replication first increases and then decreases with the growth of $n$. The reason is that when $n$ is small, the number of datacenters to which source data will be replicated is small, the cost of routing paths along which source data are replicated is small. While with the increase of the number of datacenters, some longer routing paths with

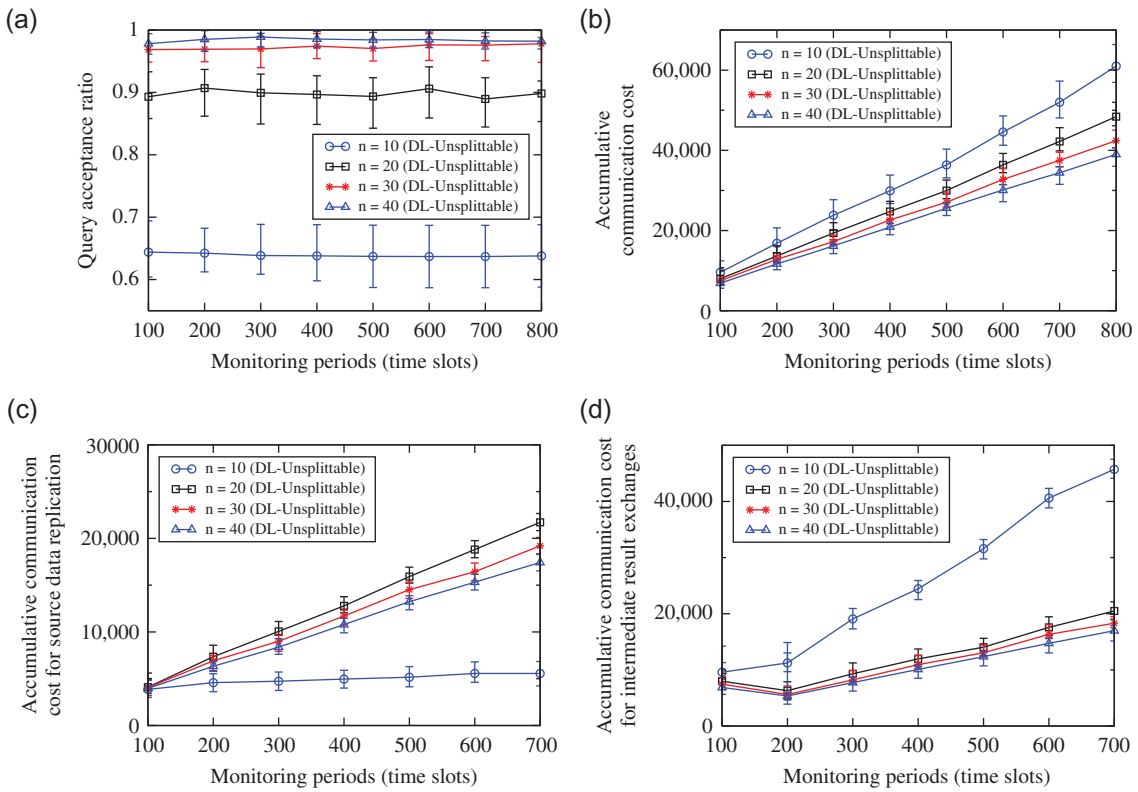


**FIGURE 4.** Impacts of the number of datacenters on the performance of algorithm `DL-Unsplittable` over various monitoring periods. (**a**) The impact on the query acceptance ratio. (**b**) The impact on the accumulative communication cost. (**c**) The impact on the accumulative communication cost on source data replication. (**d**) The impact on the accumulative communication cost on intermediate result exchanges.

abundant computing resource are selected to replicate the source data. When $n$ is sufficiently large, say $n = 30$, some datacenters that have abundant computing resource and are close to the source data locations are selected, the accumulative communication cost for source data replication thus decreases.

The accumulative communication cost of algorithm `DL-Splittable` first increases from $n = 10$ to $n = 20$, and then decreases with the increase of the value of $n$, as depicted in Fig. 5(b). The rationale is that more queries are admitted from $n = 10$ to $n = 20$ as shown in Fig. 5(a). Due to the limited available computing resource when $n = 20$, portions of the source data of each query need to be moved to the chosen datacenters for their query evaluations, thereby greatly increasing the communication costs for intermediate result exchanges as shown in Fig. 5(d). The decrease on the accumulative communication cost from $n = 20$ to $n = 40$ is due to the fact that algorithm `DL-Splittable-Alg` has more opportunities to choose datacenters that are close to each other to reduce the communication cost of intermediate result exchanges, as clearly shown by Fig. 5(d). The accumulative communication cost of source data replication decreases with the increase of $n$ as depicted in Fig. 5(c), since more datacenters imply more routing paths with lower communication costs for source data transfers. Notice that although the communication cost (i.e. bandwidth resource consumption) increases from $n = 10$ to $n = 20$, the query acceptance ratio increases too, as shown in Fig. 5(a). With the increase of $n$, the bandwidth consumption of each query may increase, the computing resource capacity of the distributed cloud increases too, allowing to admit more queries.

### 5.4. Impact of the number of source data locations on algorithm performance

We now evaluate the impact of the maximum number of data sources of each query on the query acceptance ratio, the accumulative communication cost, the accumulative communication cost for source data replication and the accumulative communication cost for intermediate result exchanges by varying the value from 2 to 8. Figures 6 and 7 are the result charts, where $S$ ($Q$) is employed to represent the maximum number of source data locations of query $Q$.

Figures 6(a) and 7(a) plot the query acceptance ratio curves of algorithms `DL-Unsplittable` and `DL-Splittable`, respectively. From Fig. 6(a), it can be seen that the query

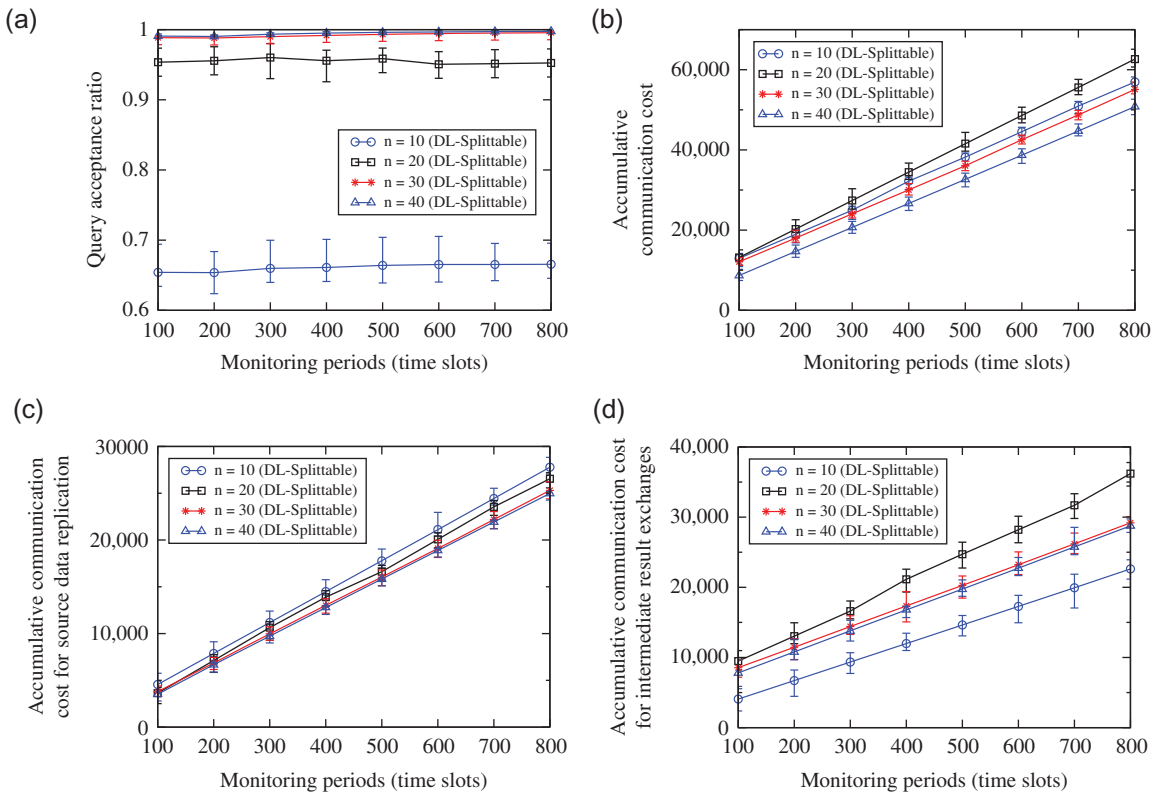


**FIGURE 5.** Impacts of the number of datacenters on the performance of algorithm `DL-Splittable` over various monitoring periods. (**a**) The impact on the query acceptance ratio. (**b**) The impact on the accumulative communication cost. (**c**) The impact on the accumulative communication cost on source data replication. (**d**) The impact on the accumulative communication cost on intermediate result exchanges.

acceptance ratio of algorithm `DL-Unsplittable` grows first and then decreases with the increase of $S(Q)$. For example, in Fig. 6(a) the query acceptance ratio increases from 88% to 94% when $S(Q) = 2$ and $S(Q) = 4$, respectively, and then decreases to 55% and 30% when $S(Q) = 6$ and $S(Q) = 8$. The rationale is that fewer data sources mean that it has a larger volume of source data at each of source data locations (datacenters). This may also increase the probability of query rejection rate due to the lack of computing resource. However, if the number of data sources $S(Q)$ is quite high (i.e. the source data of the query are distributed more datacenters), then more frequent source data replication and intermediate result exchanges are needed. Such queries also tend to be rejected by the system as limited bandwidth resource is imposed between datacenters. In contrast, the query acceptance ratio by algorithm `DL-Splittable` steadily decreases with the growth of $S(Q)$. This is because the volume of source data of a query at each datacenter does not affect the admission decision of the query, as the source data may be split to multiple datacenters, and more source data replications are therefore incurred, which increases the rejection probability of the query.

Figures 6(b) and 7(b) plot the accumulative communication cost curves of algorithm `DL-Unsplittable`. It can be seen that the accumulative communication cost increases with the growth of $S(Q)$. For example, in Fig. 6(b), when $S(Q)$ is 8, the accumulative communication cost is 1.1 times, 1.3 times and 2.3 times of that when the values of $S(Q)$ are 2, 4 and 6, respectively, while the cost for intermediate result exchanges can be seen from Figs 6(d) and 7(d). The accumulative cost for source data replication of algorithm `DL-Unsplittable` as depicted in Fig. 6(c) decreases with the increase of $S(Q)$. The cost of source data replication by algorithm `DL-Splittable` as illustrated in Fig. 7(c) first increases and then decrease with the growth of $S(Q)$.

### 5.5. Impact of parameters $a$ and $b$ on the performance of different algorithms

We finally investigate the impact of parameters $a$ and $b$ on algorithms `DL-Unsplittable` and `DL-Splittable` on the query acceptance ratio and accumulative communication cost, by varying their values from $2^1$ to $2^{11}$ when $T = 800$. It can be seen from Fig. 8(a) that the query acceptance ratios of algorithms `DL-Unsplittable` and `DL-Splittable` reach their peaks when $a = 2^4$. The rationale behind is that when $a < 2^4$, the datacenter metric of each datacenter

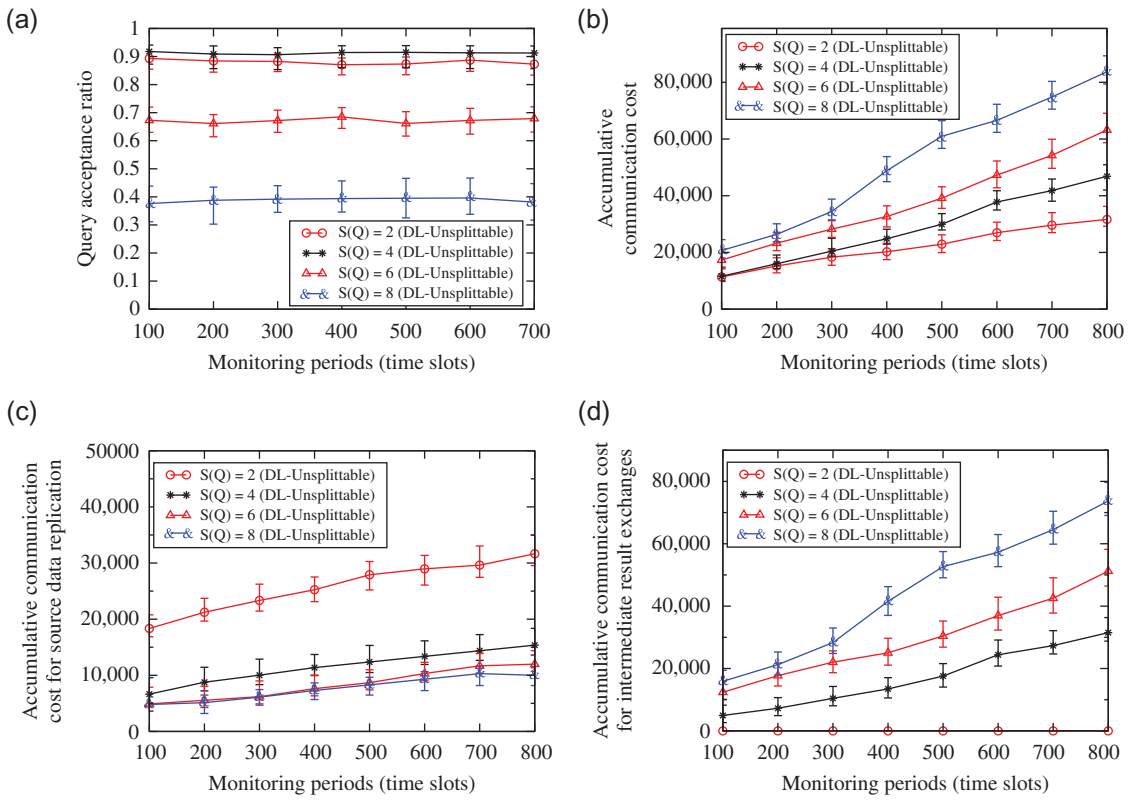


**FIGURE 6.** The impact of the number of source data locations on the performance of algorithm `DL-Unsplittable` over different monitoring periods. (**a**) The impact of the number of source data locations on acceptance ratio. (**b**) The impact of the number of source data locations on accumulative communication cost. (**c**) The impact of the number of source data locations on the accumulative communication cost for source data replication. (**d**) The impact of the number of source data locations on accumulative communication cost for intermediate result exchanges.

$v_i$, $\Phi(v_i, t)$, is not large enough to impact its ranking in the selection of processing datacenters. This results in that the algorithms may choose datacenters with less available computing resource, thereby increasing the rejection probability of queries with large volume of source data. On the other hand, when $a > 2^4$, the algorithms may select datacenters whose incident links have less available bandwidth resource, since the datacenter metric dominates the ranking of datacenters, queries with high communication demands tend to be rejected. Similar behavior patterns can be observed in Fig. 8(b). Figure 8 (c) demonstrates that the acceptance ratios of the two algorithms reach their peaks when $b = 2^6$ and then decrease when $b > 2^6$. The rationale is that when $b < 2^6$, the link metric $(\Psi(e_{i,j}, t))$ of incident links of a datacenter is too low to dominate the ranking of the datacenter. This may lead to some datacenters with insufficient network resource on their incident links to be selected, thereby increasing the rejection probability of queries with more communication demands. In contrast, when $b > 2^6$, the ranking of a datacenter will be dominated by its incident link metric $\Psi(e_{i,j}, t)$, and the selected datacenters may reject some queries as they may not have enough available computing resource. Similarly, it can be seen from Fig. 8(d) the accumulative communication cost is its minimum when $b = 2^6$.

In summary, the values of $a$ and $b$ impact not only the selected datacenters directly but also the query acceptance ratio and accumulative communication cost. Such impact can be seen from Fig. 8. Although it is difficult to derive the 'optimal' values of $a$ and $b$ for a distributed cloud, they can be easily set (or adjusted dynamically) in experiments by a simple rationale. That is, larger values for $a$ and $b$ lead to higher marginal costs in resource usages, implying allocating overloaded resources conservatively.

## 6. RELATED WORK

Existing studies on big data analytics in clouds have been conducted in recent years [1, 5, 7, 12, 18, 21, 22, 26–28, 30, 32, 36, 37, 48]. Among the studies, Mian *et al.* [32] examined query evaluation in a public cloud, by selecting a configuration for the query such that the sum of computing and storage costs of the configuration is minimized, assuming that all data accessed by the query are the local data. Kllapi *et al.* [28] provided a distributed query processing platform, Optique, to reduce the query response time. However, none of them considered the communication cost of query

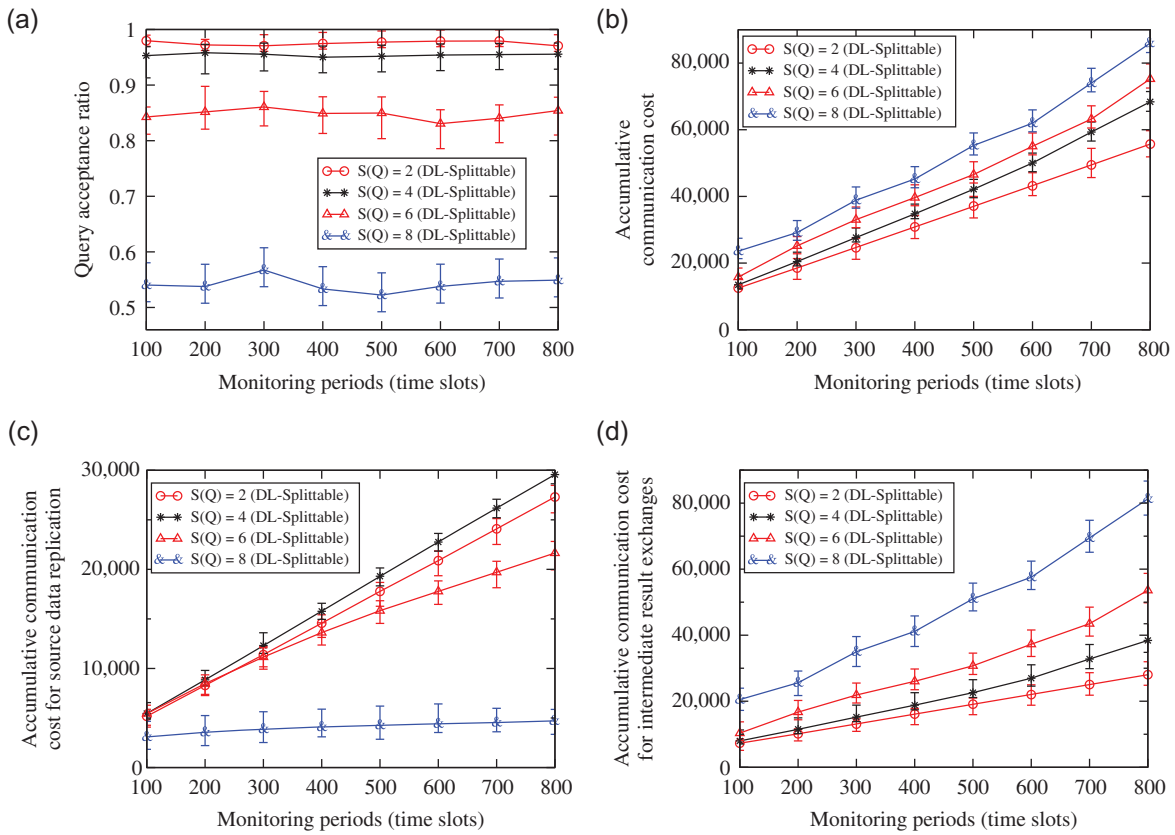


**FIGURE 7.** The impact of the number source data locations on the performance of algorithm `DL-Splittable` over different monitoring periods. (**a**) The impact of the number of source data locations on acceptance ratio. (**b**) The impact of the number of source data locations on accumulative communication cost. (**c**) The impact of the number of source data locations on accumulative communication cost for source data replication. (**d**) The impact of the number of source data locations on accumulative communication cost for intermediate result exchanges.

evaluation, which in fact cannot be ignored due to the massive data migration and the limited bandwidth among servers in a datacenter. Bruno *et al.* [5] proposed an optimization framework for continuous queries in cloud-scale systems. Particularly, they continuously monitored the query execution, collected runtime statistics and adopted different execution plans during the query evaluation. If a new plan is better than the current one, they will adopt the new plan with minimal cost. A similar problem was studied in [27], the only difference is that [27] lies in a small sample of data drawn for query execution to estimate the cost of the query evaluation plan. However, providing an accurate execution plan is time-consuming due to the massive data analysis in cloud-scale datacenters, and suboptimal plans can be disastrous with large data sets. Liu *et al.* [30] proposed three information retrieval for ranked query schemes to reduce the query overhead on the cloud. They assumed that users can choose the query ranks to determine the percentage of matched results to return. To this end, a mask matrix is used to filter out a certain percentage of matched results. Their primary motivation is providing users a scheme to restrict the number of results. Unfortunately, the correct filtering and the returned results are difficult to decide. There are other studies that focused mainly on minimizing the computing cost [5, 27, 32, 41], the storage cost [32], the query response time [28] or the server running cost [22]. Little attention has ever been paid to the communication cost in big data analytic query evaluations in a distributed cloud. Also, source data locality is another important issue which impacts the cost of query evaluation. Although several recent papers [34, 38, 41] considered data locality when dealing with query evaluation, they focused only on one single location (datacenter). For example, Tung *et al.* [38] investigated the query evaluation on databases, each of which is represented by a rooted, edge-labeled directed graph, i.e. a distributed graph. Authors in [22] tackled the resource allocation problem in clouds based on big data system, by developing a VM allocation model to handle big data tasks. Their objective is to minimize the cost for running physical servers instead of the communication cost of data transfers between datacenters. In addition, some approaches are designed with offline processing style and involves high overhead for starting and executing queries, thus they are not ideal

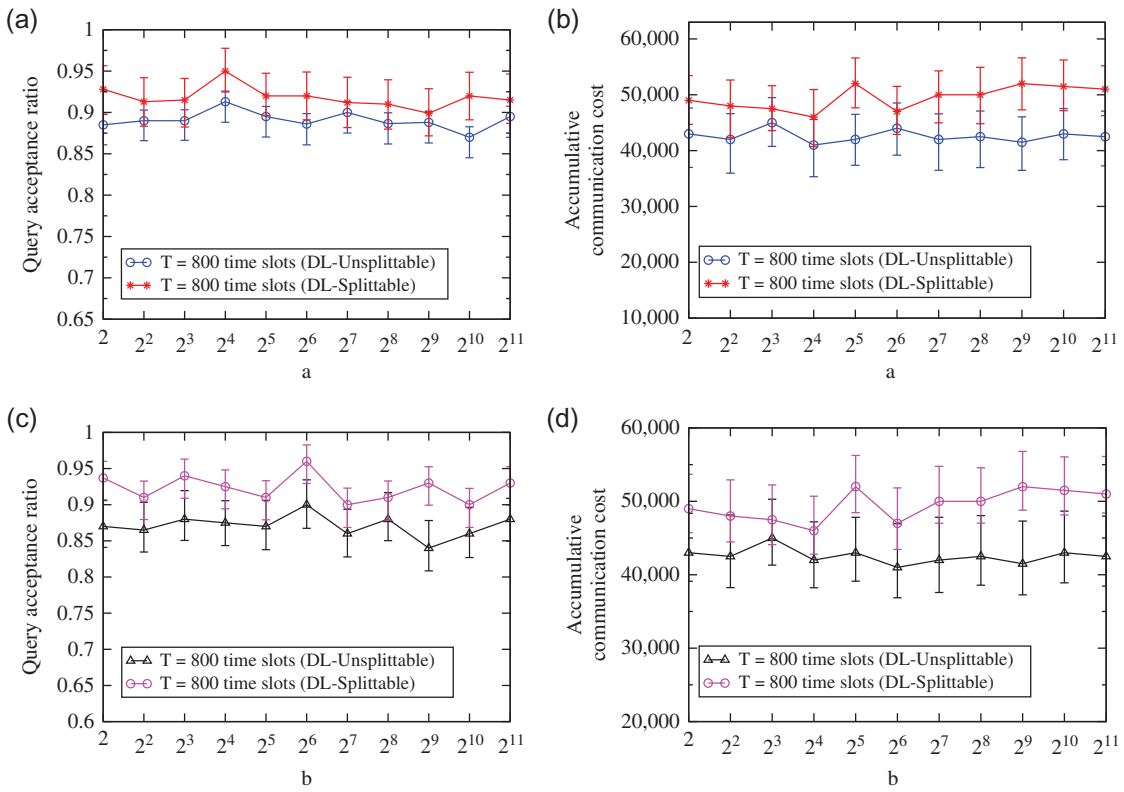


**FIGURE 8.** The impact of $a$ and $b$ on the query acceptance ratio and the accumulative communication cost of algorithm `DL-Splittable` under $T = 800$ time slots. (**a**) The impact of a on the query acceptance ratio. (**b**) The impact of a on the accumulative communication cost. (**c**) The impact of b on the query acceptance ratio. (**d**) The impact of b on the accumulative communication cost.

for online big data analytics. For example, authors in [36] discussed offline batching disk I/Os to improve MapReduce performance, which is not suitable for real-time query processing. They assumed that all related data are sent to one datacenter for processing. However, the available cloud resources in this datacenter may not meet the resource demands of the query. Other studies focused only on delivering solutions through adopting different query evaluation strategies that are commonly used in RDBMS [10], which ignored the optimization of the query evaluation cost.

There are several studies that focused on geo-distributed data analytics [19, 20, 33, 35, 39, 40]. For example, Heintz *et al*. [19] considered streaming data analytics by designing aggregation algorithms to optimize WAN traffic and the delay in obtaining the analysis results. Pu *et al*. [33] proposed a system to reduce query response times by optimizing the placement of both data and queries, and devised an online heuristic to redistribute data sets among the datacenters prior to query arrivals and to place the queries to reduce their response times. Rabkin *et al*. [35] considered real-time analysis of data that is continuously created across wide-area networks. Vulimiri *et al*. [39] considered the problem of evaluating big-data queries on data distributed in a wide-area network, they designed an architecture and algorithms to optimize query execution plans and data replication to minimize bandwidth cost, by formulating the problems into an integer linear program or a non-linear integer program, which however might not be scalable if the problem size is large. These studies however limit their discussions on specific big-data analytics such as real-time streaming data analytics in geo-distributed datacenters [19, 35], or only focused on bandwidth resource capacities while ignored the computing capacity of datacenters [33, 39, 40].

Different from these mentioned studies, in this paper, we consider efficient query evaluation for big data analytics that are computationally intensive in a distributed cloud by taking into account not only the computing capacity of each datacenter but also the bandwidth capacity of each link. The main feature of this study is how to tightly couple the source data and the computing resource demands of each query, since the source data of a query usually are located at different datacenters. A query can be efficiently evaluated only when its source data are easily accessible and its computing resource demand can be met by the allocated datacenters. We term this problem as the online query evaluation problem by jointly considering source data localities, query evaluation datacenter selection, source data replication and intermediate evaluation result exchanges, with an objective to maximize

the query acceptance ratio while minimizing the accumulative communication cost of the query evaluation.

## 7. CONCLUSIONS AND FUTURE WORK

In this paper, we considered online query evaluation for big data analytics in a distributed cloud, with an objective to maximize the query acceptance ratio while keeping the accumulative communication cost minimized, for which we first proposed a novel metric to model the usages of various cloud resources at different datacenters of the distributed cloud. We then devised efficient online algorithms for the problem under both splittable and unsplittable source data assumptions based on the proposed cost model. We finally conducted extensive experiments by simulation to evaluate the performance of the proposed algorithms, and experimental results demonstrate that the proposed algorithms are promising, and outperform two mentioned heuristics at 95% confidence intervals.

We will explore the following topics based on this work as our future work: the consideration of big data queries based on dynamically changing source data and resource sharing between different queries within each datacenter. In our future work, we will explore efficient updating and synchronizing mechanisms of placed source data to avoid source data transfers if there are only small changes in source data. In addition, different queries can share resources with each other for cost savings. Since resource sharing only happens if the VMs of different queries are allocated in the same physical server, we will propose further refinement of query evaluations within each datacenter.

## ACKNOWLEDGEMENTS

We appreciate the anonymous referees for their constructive comments and valuable suggestions, which help us significantly improve the quality and presentation of the paper.

## REFERENCES

[1] Abadi, D.J. (2009) Data management in the cloud: limitation and opportunities. *IEEE Data Engg. Bull.*, **32**, 3–12.

[2] Alicherry, M. and Lakshman, T.V. (2012) Network Aware Resource Allocation in Distributed Clouds. In *Proc. INFOCOM 12*, Orlando, Florida, March 25–30, pp. 963–971. IEEE, Washington.

[3] Amazon EC2. http://aws.amazon.com/ec2/instance-types/

[4] Ballani, H., Costa, P., Karagiannis, T. and Rowstron, A. (2011) Towards Predictable Datacenter Networks. In *Proc. SIGCOMM 11*, Toronto, Canada, August 15–19, pp. 242–253. ACM, New York.

[5] Bruno, N., Jain, S. and Zhou, J. (2013) Continuous cloud-scale query optimization and processing. *Proc. VLDB Endow.*, **6**, 961–972.

[6] Bu, X., Rao, J. and Xu, C. (2013) Interference and Locality-aware Task Scheduling for MapReduce Applications in Virtual Clusters. In *Proc. HPDC 13*, New York, USA, June 17–21, pp. 227–238. ACM, New York.

[7] Chaudhuri, S., Dayal, U. and Narasayya, V. (2011) An overview of business intelligence technology. *Commun. ACM*, **54**, 88–98.

[8] Chen, Y., Jain, S., Adhikari, V.K., Zhang, Z. and Xu, K. (2011) A First Look at Inter-Data Center Traffic Characteristics Via Yahoo! Datasets. In *Proc. INFOCOM 11*, Shanghai, China, April 10–15, pp. 1620–1628. IEEE, Washington.

[9] Cardosa, M., Wang, C., Nangia, A., Chandra, A. and Weissman, J. (2011) Exploring MapReduce Efficiency with Highly-Distributed Data. In *Proc. MapReduce*, San Jose, California, June 8–11, pp. 27–34. ACM, New York.

[10] Kossmann, D. (2000) The state of the art in distributed query processing. *ACM Comput. Surv.*, **32**, 422–469.

[11] Even S., Itai A., and Shamir A. (1975) On the Complexity of Time Table and Multi-Commodity Flow Problems. In *Proc. FOCS 75*, Berkeley, California, October 13–15, pp. 184–193. IEEE, Washington.

[12] Fan, W., Wang, X. and Wu, Y. (2012) Performance guarantees for distributed reachability queries. In *Proc. VLDB Endow.*, **5**, 1304–1316.

[13] Garg, N. and Könemann, J. (1998) Faster and Simpler Algorithms for Multi-Commodity Flow and Other Fractional Packing Problems. In *Proc. FOCS*, Palo Alto, California, November 8–11, pp. 300–309. IEEE, Washington.

[14] Guo, C., Lu, G., Wang, H.J., Yang, S., Kong, C., Sun, P., Wu, W. and Zhang, Y. (2010) SecondNet: A Data Center Network Virtualization Architecture with Bandwidth Guarantees. In *Proc. Co-NEXT 10*, Philadelphia, Pennsylvania, November 30–December 3, pp. 1–12. ACM, New York.

[15] Google Cloud Platform. https://cloud.google.com/

[16] Greenberg, A., Hamilton, J., Maltz, D.A. and Patel, P. (2009) The cost of a cloud: research problems in data center networks. *ACM SIGCOMM Comput. Commun. Rev.*, **39**, 68–73.

[17] http://www.cc.gatech.edu/projects/gtitm/

[18] Gu, L., Zeng, D., Li, P. and Guo, S. (2014) Cost minimization for big data processing in geo-distributed data centers. *IEEE Trans. Emerg. Top. Comput.*, **2**, 314–323.

[19] Heintz, B., Chandra, A. and Sitaraman, R.K. (2015) Optimizing Grouped Aggregation in Geo-Distributed Streaming Analytics. In *Proc. HPDC 15*, Portland, Oregon, USA, June 15–19, pp. 133–144. ACM, New York.

[20] Hung, C., Golubchik, L. and Yu, M. (2015) Scheduling Jobs Across Geo-Distributed Datacenters. In *Proc. SoCC 15*, Hawai'i, USA, August 27–29, pp. 111–124. ACM, New York.

[21] Herodotou H., Lim H., Luo G., Borisov N., Dong L., Cetin F. B. and Babu S. (2011) Starfish: A Self-Tuning System for Big Data Analytics. In *Proc. CIDR 11*, Asilomar, California, January 9–12, pp. 261–272. Computing Community Consortium, Washington.

[22] Hassan, M.M., Song, B., Hossain, M.S. and Alamri, A. (2014) QoS-Aware Resource Provisioning for Big Data Processing in Cloud Computing Environment. In *Proc. CSCI 14*, Las Vegas, Nevada, March 10–13, pp. 107–112. IEEE, Washington.

[23] Hashema, I.A.T., Yaqooba, I., Anuara, N.B., Mokhtara, S., Gania, A. and Khanb, S.U. (2015) The rise of 'big data' on cloud computing: review and open research issues. *J. Inf. Syst.*, **47**, 98–115.

[24] (2015) IBM interConnect 2015 day 2 recap: a new way to work with wtorage. http://servicemanagement360.com/ibm-interconnect-2015-day-2-recap-new-way-work-storage/

[25] Jagadish, H.V., Cornell, J.G., Labrinidis, A., Papakonstantinou, Y., Patel, J.M., Ramakrishnan, R. and Shahabi, C. (2014) Big data and its technical challenges. *Commun. ACM*, **57**, 86–94.

[26] Ji, C., Li, Y., Qiu, W., Awada, U. and Li, K. (2012) Big Data Processing in Cloud Computing Environments. In *Proc. ISPAN 12*, San Marcos, Texas, December 13–15, pp. 17–23. IEEE, Washington.

[27] Karanasos K., Balmin A., Kutsch M., Ozcan F., Ercegovac V., Xia C. and Jackson J. (2014) Dynamically Optimizing Queries Over Large Scale Data Platforms. In *Proc. SIGMOD 14*, New York, June 22–27, pp. 943–954. ACM, New York.

[28] Kllapi, H., Bilidas, D., Horrocks, I., Ioannidis, Y., Jimenez-Ruiz, E., Kharlamov, E., Koubarakis, M. and Zheleznyakov, D. (2012) Distributed Query Processing on the Cloud: The Optique Point of View (short paper). In OWL: Experiences and Directions Workshop 12, Montpellier, France, May 26–27.

[29] Kolliopoulos, S.G. and Stein, C. (1997) Improved Approximation Algorithms for Unsplittable Flow Problems. In *Proc. FOCS 97*, Miami Beach, October 20–22, pp. 426–436. IEEE, Washington.

[30] Liu, Q., Tan, C.C., Wu, J. and Wang, G. (2014) Towards differential query services in cost-efficient clouds. *Trans. Parall. Distrib. Syst.*, **25**, 1648–1658.

[31] Understanding MapReduce input VS. file chunk sizes. https://www.mapr.com/developercentral/code/understanding-mapreduce-input-vs-file-chunk-sizes#.VDcolKVlnng

[32] Mian, R., Martin, P. and Vazquez-Poletti, J.L. (2013) Provisioning data analytic workloads in a cloud. *Future Gener. Comput. Syst.*, **29**, 1452–1458.

[33] Pu, Q., Ananthanarayanan, G., Bodik, P., Kandula, S., Akella, A., Bahl, P. and Stoica, I. (2015) Low Latency Geo-Distributed Data Analytics. In *Proc. SIGCOMM 15*, London, United Kingdom, August 17–21, pp. 421–434. ACM, New York.

[34] Palanisamy, B., Singh, A., Liu, L. and Jain, B. (2011) Purlieus: Locality-Aware Resource Allocation for MapReduce in a Cloud. In Proc. SC 11, Seattle, Washington, November 12–18, pp. 1–11. ACM, New York.

[35] Rabkin, A., Arye, M., Sen, S., Pai, V.S. and Freedman, M.J. (2014) Aggregation and Degradation in JetStream: Streaming Analytics in the Wide Area. In *Proc. NSDI 14*, Seattle, WA, USA, April 2–4, pp. 275–288. USENIX Association Berkeley, CA, USA.

[36] Rao, S., Ramakrishnan, R., Silberstein, A., Ovsiannikov, M. and Reeves, D. (2012) Sailfish: A Framework for Large Scale Data Processing. In *Proc. SoCC 12*, San Jose, California, October 14–17, pp. 1–14. ACM, New York.

[37] Sakr, S., Liu, A., Batista, D.M. and Alomari, M. (2011) A survey of large scale data management approaches in cloud environments. *IEEE Commun. Surv. Tutor.*, **13**, 311–336.

[38] Tung, L., Nguyen-Van, Q. and Hu, Z. (2013) Efficient Query Evaluation on Distributed Graphs with Hadoop Environment. In *Proc. SoICT 13*, Danang, Vietnam, December 5–6, pp. 311–319. ACM, New York.

[39] Vulimiri, A., Curino, C., Godfrey, P.B., Jungblut, T., Padhye, J. and Varghese, G. (2015) Global Analytics in the Face of Bandwidth and Regulatory Constraints. In *Proc. NSDI 15*, Oakland, CA, USA, May 4–6, pp. 323–336. USENIX Association Berkeley, CA, USA.

[40] Vulimiri, A., Curino, C., Godfrey, B., Karanasos, K. and Varghese, G. (2015) WANalytics: Analytics for a Geo-Distributed Data-Intensive World. In *Proc. CIDR 15*, Asilomar, California, USA, January 4–7, ACM New York.

[41] Wu, S., Li, F., Mehrotra, S., and Ooi B.C. (2011) Query Optimization for Massively Parallel Data Processing. In *Proc. SOCC 11*, Cascais, Portugal, October 26–28, pp. 1–13. ACM, New York.

[42] Xu, Z. and Liang, W. (2013) Minimizing the Operational Cost of Data Centers Via Geographical Electricity Price Diversity. In *Proc. IEEE CLOUD 13*, Santa Clara, California, June 28–July 3, pp. 99–106. IEEE, Washington.

[43] Xu, Z., Liang, W. and Xia, Q. (2014) Efficient Virtual Network Embedding Via Exploring Periodic Resource Demands. In *Proc. LCN 14*, Edmonton, Alberta, Canada, 8–11 September, pp. 90–98. IEEE, Washington.

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

[45] Xia, Q., Liang, W. and Xu Z. (2014) Data Locality-Aware Query Evaluation for Big Data Analytics in Distributed Clouds. In *Proc. CBD 14*, Huangshan, Anhui, China, November 20–22, pp. 1–8. IEEE, Washington.

[46] Xia, Q., Xu, Z., Liang, W. and Zomaya, A. (2016) Collaboration- and fairness-aware big data management in distributed clouds. *IEEE Trans. Parall. Distrib. Syst.*, **27**, 1941–1953.

[47] Zhang, L., Li, Z., Wu, C. and Chen, M. (2014) Online Algorithms for Uploading Deferrable Big Data to the Cloud. In *Proc. INFOCOM 14*, Toronto, Ontario, Canada, April 27–May 2, pp. 2022–2030. IEEE, Washington.

[48] Zhang, L., Wu, C., Li, Z., Guo, C., Chen, M. and Lau, F.C. M. (2013) Moving big data to the cloud: an online cost-minimizing approach. *J. Sel. Area. Commun.*, **31**, 2710–2721.