Received August 29, 2020, accepted September 20, 2020, date of publication September 29, 2020, date of current version October 9, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3027675

Distributed Data Strategies to Support Large-Scale Data Analysis Across Geo-Distributed Data Centers

TAMER Z. EMARA[1,2,3], **(Member, IEEE), AND JOSHUA ZHEXUE HUANG**[1,2]
[1]National Engineering Laboratory for Big Data System Computing Technology, Shenzhen University, Shenzhen 518060, China
[2]College of Computer Science and Software Engineering, Big Data Institute, Shenzhen University, Shenzhen 518060, China
[3]Higher Institute of Engineering and Technology Kafrelsheikh, 33514 Kafrelsheikh, Egypt

Corresponding author: Joshua Zhexue Huang (zx.huang@szu.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61972261.

ABSTRACT As the volume of data grows rapidly, storing big data in a single data center is no longer feasible. Hence, companies have developed two scenarios to store their big data in multiple data centers. In the first scenario, the company's big data are distributed in multiple data centers without data replication. In the second scenario, data are also stored in multiple data centers but important data are replicated in these data centers to increase data safety and availability. However, in these scenarios, analyzing big data distributed in multiple data centers becomes a challenging task. In this paper, we propose two data distribution strategies to support big data analysis across geo-distributed data centers. In these strategies, we use the recent Random Sample Partition data model to convert big data into sets of random sample data blocks and distribute these data blocks into multiple data centers either without replication or with replication. In analyzing big data in multiple data centers without replication, we randomly select samples of data blocks from multiple data centers and download the sample data blocks to one data center for analysis. In the second strategy with replication of data blocks, we can analyze big data on any data center by randomly selecting a sample of data blocks replicated from other data centers. This strategy avoids data transformation between data centers. We demonstrate the performance of the two strategies in big data analysis by using simulation results produced on one local data center and four AWS data centers in North America, Asia, and Australia.

INDEX TERMS Big data analysis, cloud data centers, distributed computing, random sample partition, wide area analytics.

I. INTRODUCTION

The big companies that operate over vast geographical areas need multiple data centers to collect and store data from their clients or users. For example, airline companies collect data from different departments that operate in many countries. The data is usually stored in multiple data centers, and its volume grows rapidly, often exceeding terabytes in each data center. Those companies need to analyze their big data in multiple data centers, as a whole, for many reasons: making personalized recommendations to targeted customers, delivering advertisements to global users, and producing global reports for decision-makers [1]. When taking data in multiple data centers into consideration as a whole, analyzing such big data efficiently and effectively becomes a big challenge to current big data technologies.

Apache Hadoop [2] and Spark [3] are the most popular frameworks for parallel and distributed big data processing and analysis [4]. However, these frameworks are designed to process data locally within the same data center, and hence, they are in need to copy all data to a single data center before processing a locally distributed computation [5], [6]. Copying the complete data from different data centers to one data center can be an operation bottleneck due to the bandwidth limitation, communication cost, data privacy and security.

In order to tackle the problem of analyzing big data across multiple data centers, many solutions on distributed execution have been proposed. Distributed execution is a strategy to distribute the analytic jobs or tasks among geo-distributed data centers and then aggregate the intermediate results for further processing [5]. For example, the authors in [7]–[11]

The associate editor coordinating the review of this manuscript and approving it for publication was Pasquale De Meo.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

have proposed different task schedulers to reduce the volume of data traffic. Volley [12] and Iridium [13] considered the data placement issue in order to reduce job completion times by redistributing data across the geo-distributed data centers according to the users' requests or queries. In [14], the authors improved the shuffle stages by proactively moving data in the shuffle phase from mapper to reducer tasks to improve data locality.

Despite their promising perspectives, these strategies may not achieve the optimal outcome due to the level of abstraction needed to solve the problem. For example, the existing works assumed that intermediate data sizes are known beforehand, even though this is rarely the case in practice. Also, different task assignment strategies lead to different flow patterns across data centers, and ultimately, different job completion times. Besides, the statistical distribution of data, as well as the sizes of data, changes over time across different data centers. Furthermore, the available capacities of data centers vary as well. For example, the heavy utilization of one or more data centers by non-analytical jobs can lead to limited availability of resources, with the end result of slow system response and poor performance. Finally, the fluctuation of network bandwidth is also a critical factor because the data centers with low bandwidth can become a bottleneck in case of the equal spreading of tasks among all data centers.

Random sample partition (RSP) data model [15], [16] was recently proposed to support distributed big data analysis. In the RSP data model, a big data file is represented as a set of non-overlapping data blocks, called *RSP data blocks*, each being a random sample of the entire big data. A two-stage data processing algorithm [17] was proposed to generate RSP data blocks from other HDFS data files. A big data management system (BDMS) was proposed in [18] to manage RSP data blocks on a computing cluster. A Spark library, called RRPlib [19], was implemented in BDMS for functions of RSP data block generation and RSP block-level sampling.

In this paper, we are interested in the problems of analyzing big data across multiple data centers. In practice, companies have developed two scenarios to store their big data in multiple data centers either without replication or with replications. Considering the two scenarios and based on the RSP data model, we propose two data distribution strategies to enable big data analysis across geo-distributed data centers. In the first strategy, the BDMS system is installed in each data center to generate and store RSP data blocks of big data sets in the data center. If a big data set is stored in multiple data centers, each containing a subset of the big data, the RSP data blocks of each subset of the big data are stored as an RSP data model in the local BDMS. When analyzing the big data as a whole, a set of RSP data blocks is randomly selected in each data center and downloaded to the data center where the RSP data blocks of different data centers are merged to form a new set of RSP data blocks representing a set of random samples of the entire big data. The new set of RSP data blocks is analyzed in the data center and the results are used as the approximate results of the entire big data. This strategy supports the concept of the data lake [20] and isolates the data analysis process from the storage.

The second strategy we propose is to replicate RSP data blocks in multiple data centers for improving data availability and safety. In this strategy, the BDMS system is also installed in all data centers. The big data set in each data center is converted to RSP data blocks managed by BDMS. Subsets of RSP data blocks in each big data set are replicated in other data centers. Since the data blocks of the RSP data models for the same big data set are replicated in all data centers, every data center contains the RSP data blocks randomly selected from the RSP data models in other data centers. By merging the RSP data blocks from different RSP data models, a set of new RSP data blocks, representing random samples of the entire big data set, are obtained. These new RSP data blocks can be analyzed to obtain the approximate results of the entire data set. Since sampling and analysis of RSP data blocks are carried out in one data center, there is no data communication requirement in this strategy.

In this paper, we present the specific steps and algorithms to implement the two strategies for big data analysis across geo-distributed data centers. To demonstrate the performance of big data analysis using the proposed strategies, we built a simulation environment consisting of one local data center and four AWS data centers in North America, Asia and Australia, and used this environment to conduct several experiments on simulated big data. In these experiments, the performance was measured in two stages, the data downloading stage and data analysis stage. The remote AWS data centers were used to measure the data downloading stage and the local data center was used to evaluate the performance of data analysis stage. The data downloading stage is only required in the first strategy and the downloading performance is determined by the network speed. In practice, the size of an RSP data block is small compared to the big data. Our experiment results show that downloading a small set of RSP data blocks from a remote data center took time in minutes. In the data analysis stage, our experiment results on the local data center show that it is computationally efficient to analyze RSP data blocks on a computing cluster and a small set of RSP data blocks is enough to get an approximate result close to the result from the whole data. Moreover, using the second strategy, we can build component models in different data centers and ensemble them into an ensemble model by transferring the component models to the data center where the ensemble model is built. Since the size of component models is very small, the component model transfer can be completed in a few minutes. The ensemble models are more accurate than the single model built from the entire big data.

The remainder of this paper is organized as follows. **Section** II discusses the related works. An overview of big data management system is presented in **Section** III. The proposed strategies are introduced in **Section** IV. In **Section** V, we illustrate the results of our experiments. **Section** VI discusses the results. Finally, **Section** VII concludes the paper.

II. RELATED WORKS

In this section, we review the related works in wide-area data analytics and data replications in cloud environments.

A. WIDE AREA DATA ANALYTICS

Research on analyzing data across geo-distributed data centers has increasingly gained popularity in recent years. The primary objective of the existing efforts is to effectively deploy the data analytic jobs across multiple data centers. For example, Vulimiri *et al.* [7], [8] and Kloudas *et al.* [9] proposed task assignment strategies for the sake of reducing data transfers across data centers. Liu *et al.* [14] focused on the placement of shuffle inputs to facilitate task placement algorithms. Although data transfers among data centers can be reduced, these solutions do not necessarily shorten job completion times, as the bandwidth capacities of the network vary over time and the available resources also change from time to time in different data centers.

Pu *et al.* [13] proposed an online heuristic to replace both data and tasks across data centers. Hung *et al.* [21] proposed a greedy scheduling heuristic to schedule the analytic jobs across geo-distributed data centers. They focused on allocating the computing resources through coordinated job scheduling, assuming that the task assignment is predetermined so that the execution order of all assigned tasks can be scheduled in each data center. Hung *et al.* [22] considered both computing and network resources for task placement in order to minimize the network transfer. Unfortunately, they assume that the data centers are connected with a congestion-free scenario, which is not realistic. Hu *et al.* [23] removed this unrealistic assumption, and formulated a lexicographical minimization problem of task assignment for a single stage of one job. Chen *et al.* [10] considered the same problem for multiple jobs.

B. DATA REPLICATION IN CLOUD ENVIRONMENTS

Many recent works in literature act towards data replications in cloud systems. Boru *et al.* [24] proposed a data replication technique for cloud data centers. The proposed technique optimizes energy consumption related to the network bandwidth, and communication delay in both intra- and inter-data centers. Gill and Singh [25] proposed an algorithm to optimize the replication cost using the concept of the knapsack problem. The idea behind this algorithm is to re-transfer the replicas to a lower-cost data center when the cost of replication exceeds the user budget. Liu and Shen [26] proposed a popularity-aware multi-failure resilient and cost-effective replication (PMCR) scheme. In this scheme, the cloud storage system is divided into two tiers, the primary tier and the backup tier. The proposed scheme manages both correlated and independent failures by replicating the data on two servers in the primary tier and one server in the backup tier. The simulation results showed that PMCR guarantees high data availability and durability.

Many works have proposed dynamic replication strategies to minimize the latency for high availability [27]–[30]. Dynamic replication strategies mainly rely on creating replicas of frequently accessed data close to the user devices. Mansouri and Javidi [27] proposed a dynamic replication strategy called prefetching-aware data replication (PDR). The proposed algorithm pre-fetches the most popular files based on the correlations of the data files in the file access history. It works through three stages. The first stage is to build a dependency matrix by calculating the dependencies between all files. The second stage is to determine the most popular file according to the total average of file accesses. In the third stage, unnecessary replicas are replaced with more popular replicas to save the storage space of each node.

Matri *et al.* [28] introduced a write-enabled dynamic replication scheme that leverages the decentralized architecture of decentralized storage systems such as Dynamo [31] or Voldemort [32]. They also proposed an approximate object location method to enable clients to locate the closest data replica tentatively.

Mansouri *et al.* [29] proposed a method to optimize the offline cost for data migration and dynamic replication among different cloud data centers. Limam *et al.* [30] proposed a dynamic replication strategy to improve the availability and satisfy of tenant's data by taking into account tenant budget and provider's profit. The authors investigated a cost model to calculate the minimum number of replicas that can assure high data availability. Based on the cost model, the replica can only be created before the minimum number of replicas is reached or when the required response time is not exceeded. Besides, the authors focused only on the replication of read-only data.

In summary, the above works are focused on the techniques to facilitate the creation of data replicas for saving the cost and reducing the execution time. In this paper, we propose a new data replication strategy to support big data analysis.

III. OVERVIEW OF BDMS

The big data management system (BDMS) [18] was recently proposed based on the random sample partition (RSP) data model [16]. The design objective of this system was to store big data sets in the RSP data model as distributed RSP data blocks, and to support block-level sampling of RSP data blocks for approximate analysis of big data on computing clusters. This system offers major functionalities for management of RSP data blocks such as RSP data block generation through random samples data partitioning, data blocks organization, and data blocks sampling. Organizing and managing big data sets as RSP data blocks can effectively and efficiently support block-based sampling for various data analysis tasks carried out on computing clusters. The significant impact of this system can be extended to approximate big data analysis across data centers.

The BDMS architecture, as shown in Fig. 1, consists of three main components: the block manager, the block registry, and the BDMS client. The HDFS system is used as the underlying distributed file system for storing RSP data blocks as data block files on the local nodes of a cluster. The block

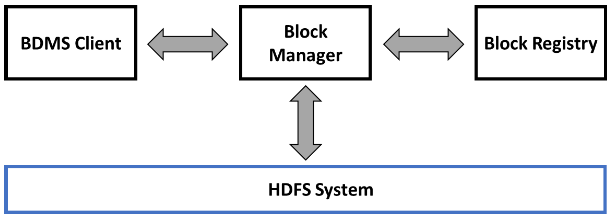

FIGURE 1. The high-level architecture of BDMS.

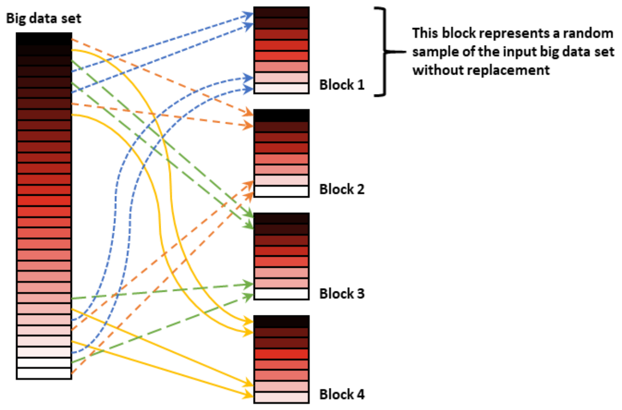
Big data set
Block 1
Block 2
Block 3
Block 4
This block represents a random sample of the input big data set without replacement

FIGURE 2. Round-random partitioning scheme.

manager is responsible for converting the big data files to RSP data blocks, and also for organizing the logical structure of the file blocks. The block registry is a data dictionary that provides the statistical summary and all metadata information about the RSP data blocks. The BDMS client provides a library for interfacing the BDMS to other applications.

When a big data file is imported to BDMS, the block manager converts the big file into a set of data blocks, each being a random sample of the entire data set in the data file. This operation is executed based on the round-random partitioning scheme (RRPS), which is defined as *a horizontal partitioning scheme which distributes the records in a random order to P blocks, i.e., RRPS maps the i-th record to block (k mod P), where k is a random number selected from* $\{1, \ldots, n\}$ *without replacement, and n is the number of records.* The RRPS scheme is illustrated in Fig. 2, where the left block represents a big data file, and the blocks on the right are the data blocks converted from the big data file.

Based on the RRPS scheme, two algorithms were implemented in BDMS to partition the big data set into a set of RSP data blocks. The round-random partitioner (RRP) algorithm is an in-memory algorithm to convert a big data set to RSP data blocks. It is used for a data set less or equal to one terabyte. The round-random partitioner for massive data (Massive-RRP) algorithm uses a divide-and-conquer strategy to convert big data greater than one terabyte. It cuts a big data file into small files that can be processed in memory and iteratively executes the RRP algorithm to convert the small files into RSP data blocks, which are integrated into a final set of RSP data blocks for the entire big data. The Massive-RRP algorithm is useful for the cluster with limited memory.

We remark that the foundation of BDMS is the random sample partition data model [16] which is rephrased as follows:

Definition (RSP data model): Let $\mathbb{D}$ be a big data set of N records which are observations of independently and identically distributed random vectors $X_1, X_2, \ldots, X_N$ where N is very large. Assume $\mathbb{F}(x)$ is the empirical distribution of $\mathbb{D}$. Let $\mathbb{T}$ be a partition operation which divides $\mathbb{D}$ into a set of subsets $\mathrm{T} = \{D_1, D_2, \cdots, D_K\}$, each containing n records. T is called a *random sample partition* of $\mathbb{D}$ if

(1) $\bigcup_{k=1}^{K} D_k = \mathbb{D}$;

(2) $D_i \cap D_j = \emptyset$, where $i, j \in \{1, 2, \cdots, K\}$ and $i \neq j$;

(3) $E[\mathbb{F}_k(x)] = \mathbb{F}(x)$ *for each* $k = 1, 2, \cdots, K$, where $\mathbb{F}_k(x)$ denotes the empirical distribution of D_k and $E[\mathbb{F}_k(x)]$ denotes its expectation.

We call each D_k an RSP data block of $\mathbb{D}$ and T an RSP data model of $\mathbb{D}$.

IV. TWO DATA DISTRIBUTION STRATEGIES

As the data volume grows rapidly, storing big data in a single data center may no longer be practical for many companies. In practice, modern businesses choose to store big data in multiple geo-distributed data centers using the following two scenarios:

- **Scenario 1: storing data *without replication***
  In this scenario, each data set is stored only in one data center without replication in other data centers. Different data sets are stored in different data centers for easy access, and even one big data can be distributed across different systems and locations because it is more efficient to store the data where it is generated [5]. This scenario saves storage space and simplifies data management.
- **Scenario 2: storing data *with replication***
  In this scenario, companies choose to replicate some important data sets in multiple data centers for the following reasons: (1) The data that spans multiple data centers ensure the fastest response time for customers and workforces who are geographically separated [33]. (2) Data safety and availability are protected in case that a single data center experiences a disaster [34].

In this section, we propose two strategies for analyzing big data distributed on multiple data centers following the two scenarios.

A. PROBLEM DEFINITION

Let $\mathfrak{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_\mathfrak{u}\}$ be a big dataset divided into $\mathfrak{u}$ subsets, each stored in a separate data center, where

(i) $\bigcup_{i=1}^{\mathfrak{u}} \mathcal{D}_i = \mathfrak{D}$

(ii) $\mathcal{D}_i \cap \mathcal{D}_j = \phi$ for $i, j \in \{1, 2, .., \mathfrak{u}\}$ and $i \neq j$.

In reality, the second condition may not be true but we can carry out some data processing to satisfy that condition.

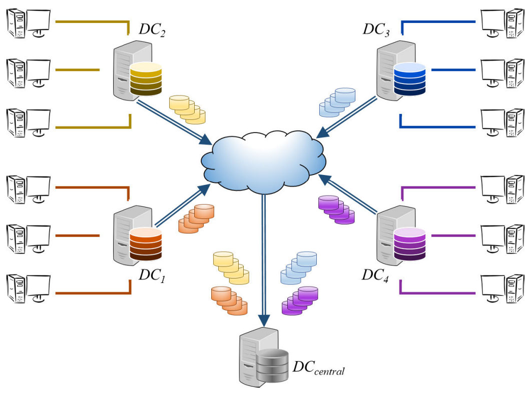

FIGURE 3. Process of Strategy 1 to store data on multiple data centers: randomly select a set of RSP data blocks from each data center and then download the selected RSP data blocks to the central data center for data analysis.

Depending on the scenarios used, each subset $\mathcal{D}_i$ can be stored in one data center only or stored in one data center but also replicated in other data centers. Our research objective is to enable the analyses of big data $\mathfrak{D}$ as a whole through approximate analyses of random samples from datasets $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_\mathfrak{u}$ stored in multiple data centers.

The problem can be described as follows. Given a data analysis algorithm $\mathfrak{B}$, we want to use $\mathfrak{B}$ to estimate a statistical property $\mathcal{F}$ of $\mathfrak{D}$ or build a model for $\mathfrak{D}$. The naive method is to copy $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_\mathfrak{u}$ from different data centers to a single data center $DC_{central}$ and use $\mathfrak{B}$ to analyze $\mathfrak{D}$ on it. However, this naive method is not practically applicable because the download cost and the size of $\mathfrak{D}$ which is too large to process on $DC_{central}$.

To conquer this problem, we propose two strategies for implementation of approximate analysis steps which take the two scenarios into consideration.

B. STRATEGY 1

The first strategy is proposed to analyze data stored in Scenario 1, i.e., without data replication. We use BDMS to store the data in each data center as a set of RSP data blocks. After that, a subset of the RSP data blocks is selected randomly from each data center and downloaded to a central data center $DC_{central}$. Next, these subsets are combined to form a set of new RSP data blocks which are random samples of $\mathfrak{D}$. This process is illustrated in Fig. 3. The big data is distributed in four data centers $\{DC_1, DC_2, DC_3, DC_4\}$, and each data center stores data as RSP data blocks. Four RSP data blocks are randomly selected from each data center and downloaded to the data center $DC_{central}$. In $DC_{central}$, the four sets of RSP data blocks are merged into one set of four new RSP data blocks, each being formed by merging four RSP data blocks randomly selected without replacement from each set of downloaded RSP data blocks. The set of new RSP data blocks can be analyzed to provide the estimated results or models of the whole big data.

The steps of data analysis using Strategy 1 are presented as follows:

- **Step 1:** Generating RSP.
  Let $\mathcal{D}_i \in \mathfrak{D}$ be a big data file stored in the ith data center. Use BDMS to convert $\mathcal{D}_i$ to a set of RSP data blocks $\mathcal{R}_i$ as
  $$\mathcal{R}_i = RSP(\mathcal{D}_i);$$
  where $RSP(.)$ is the conversion function to an RSP data model.
- **Step 2:** Sampling RSP data blocks.
  In this step, select b data blocks randomly from $\mathcal{R}_i$ without replacement to form a sample set $\mathcal{S}^i$ for the ith data center, as
  $$\mathcal{S}^i = \{S_1^i, S_2^i, \ldots, S_b^i\}, \mathcal{S}^i \subset \mathcal{R}_i$$
  where $b \leq B$, and B is the total number of data blocks in $\mathcal{R}_i$. According to the RSP theory, each data block S_j^i is a random sample of $\mathcal{D}_i$; therefore, $\mathcal{S}^i$ is a set of random samples of $\mathcal{D}_i$. This step is operated on the data in each data center.
- **Step 3:** Downloading the selected RSP data blocks to the central data center.
  In this step, we download the $\mathfrak{u}$ samples, each sample being a set of selected RSP data blocks, to $DC_{central}$ to construct a collection of subsets $\mathfrak{S}$
  $$\mathfrak{S} = \{\bar{S}_1, \bar{S}_2, \ldots, \bar{S}_b\}$$
  where each element $\bar{S}_j$ is
  $$\bar{S}_j = \bigcup_{i=1}^{\mathfrak{u}} S_j^i,$$
  which is a random sample data block of $\mathfrak{D}$, according to Theorem 1 in [17]. Hence, $\mathfrak{S}$ is also a random sample of $\mathfrak{D}$ and can be used to analyze $\mathfrak{D}$.
- **Step 4:** Analyzing $\mathfrak{S}$.
  In this step, we use $\mathfrak{S}$ to estimate a statistic $\mathcal{F}$ of $\mathfrak{D}$ or build a model for $\mathfrak{D}$. Since $\mathfrak{S}$ is a random sample of $\mathfrak{D}$ and has statistical properties as $\mathfrak{D}$, analyzing $\mathfrak{D}$ can obtain approximate results of $\mathfrak{D}$. The size of $\mathfrak{S}$ is decidedly smaller than the size of $\mathfrak{D}$, so it can be efficiently processed in $DC_{central}$ in a parallel and distributed fashion.
  To analyze $\mathfrak{S}$ in $DC_{central}$, we can use an RSP-based ensemble method [16] as follows. A base learning algorithm $\mathfrak{B}$ is applied on each RSP block in $\mathfrak{S}$ to build a base learner model f. Then, an ensemble model $\mathcal{F}$ is built as
  $$\mathcal{F} = \Psi(f_1, f_2, .., f_b)$$
  where Ψ is the consensus function of the ensemble model.

We can implement these four steps in two algorithms, as shown in Algorithm 1 and Algorithm 2. Algorithm 1 implements the first two steps to generate RSP data blocks and randomly select a sample of data blocks in each data center. Algorithm 2 performs the analysis on the data

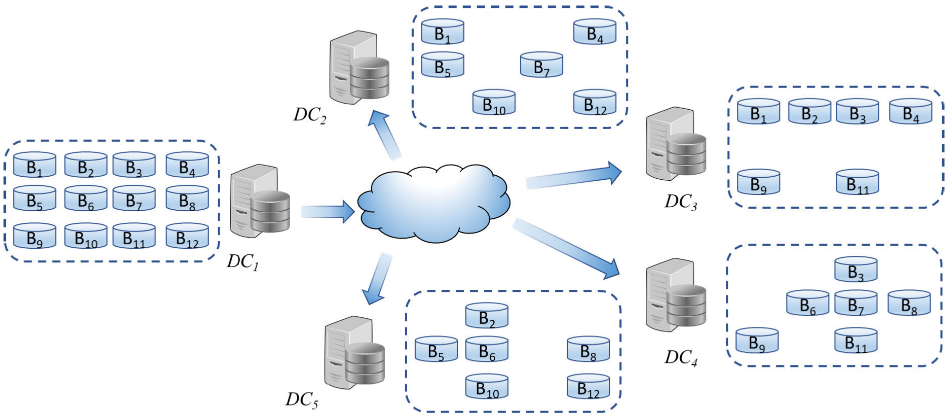

FIGURE 4. This example illustrates replicating a data file stored in the data center DC_1 into another 4 data centers using the block-level data replication method. The data file is first divided to a set of blocks (12 blocks in this example) and then these blocks are replicated randomly to the other data centers with a replication factor 3.

Algorithm 1 Data Sampling

```
Input:
    $\mathcal{D}$: stored data in the data center;
    $b$: number of blocks to be selected.
Output:
    $\mathcal{S}$: a random sample of $\mathcal{D}$.
Method:
1: $\mathcal{R} \leftarrow RSP(\mathcal{D})$
2: $B \leftarrow \mathcal{R}.size$
3: if $B < b$ then
4:     $\mathcal{S} \leftarrow \mathcal{R}$
5: else
6:     $\mathcal{S} \leftarrow sampleWithoutReplacement(\mathcal{R}, b)$
7: end if
8: return $\mathcal{S}$
```

samples. Algorithm 1 is executed independently on each data center, while Algorithm 2 runs in the central data center $DC_{central}$.

C. STRATEGY 2

Strategy 2 is proposed to analyze data stored in the second scenario, i.e., with replication. We first introduce our new data replication method, which is called *block-level data replication*, and then discuss the proposed strategy. In the block-level data replication, we replicate the RSP data blocks in each data center instead of individual data files. Let $\mathcal{D}$ be a big data file stored in a data center. First, we use BDMS to convert $\mathcal{D}$ as a set of RSP data blocks. Second, each RSP data block is replicated randomly to γ data centers, where γ is a replication factor. Fig. 4 shows an example of replicating RSP data blocks of $\mathcal{D}$ in the data center DC_1 into other four data centers $\{DC_2, DC_3, DC_4, DC_5\}$. The big data set $\mathcal{D}$ is stored in DC_1 as a set of RSP data blocks $\{B_1, B_2, \ldots\}$. We assume the replication factor is 3. Hence, each RSP data block is replicated randomly to other two data centers. For instance,

Algorithm 2 Data Analysis

```
Input:
    $\mathfrak{u}$: number of data centers;
    $\mathfrak{B}$: the base learning algorithm;
    $b$: the sample size.
Output:
    $\mathcal{F}$: an approximate model of $\mathcal{D}$.
Method:
1: $\mathbb{S} = \phi$
2: $\mathfrak{S} = \phi$
3: for $i = 1$ to $\mathfrak{u}$ do
4:     $\mathcal{S}$ $\leftarrow$ request a sample from data center $i$ using Algorithm 1($i.data$, $b$)
5:     append $\mathcal{S}$ to $\mathbb{S}$
6: end for
7: for $j = 1$ to $b$ do
8:     $S_{combined}$ $\leftarrow$ create an empty RSP data block
9:     for each $\mathcal{S} \in \mathbb{S}$ do
10:        $S_{selected}$ $\leftarrow$ select the $j$th RSP data block from $\mathcal{S}$
11:        $S_{combined}$ $\leftarrow$ $S_{combined} \cup S_{selected}$
12:    end for
13:    append $S_{combined}$ to $\mathfrak{S}$
14: end for
15: $f_{set} = \phi$
16: for each $\mathcal{S} \in \mathfrak{S}$ do
17:     $f$ $\leftarrow$ $\mathfrak{B}(\mathcal{S})$
18:     append $f$ to $f_{set}$
19: end for
20: $\mathcal{F} \leftarrow \Psi(f_{set})$
21: return $\mathcal{F}$
```

B_1 is replicated in DC_2 and DC_3, while B_2 is replicated in DC_3 and DC_5, and so on. Finally, every data center has a set of RSP data blocks of $\mathcal{D}$. Similarly, as shown in Fig. 5, five data files are stored separately in the five data centers as RSP data blocks. By replication, every data center has a set of RSP data blocks from other data centers.

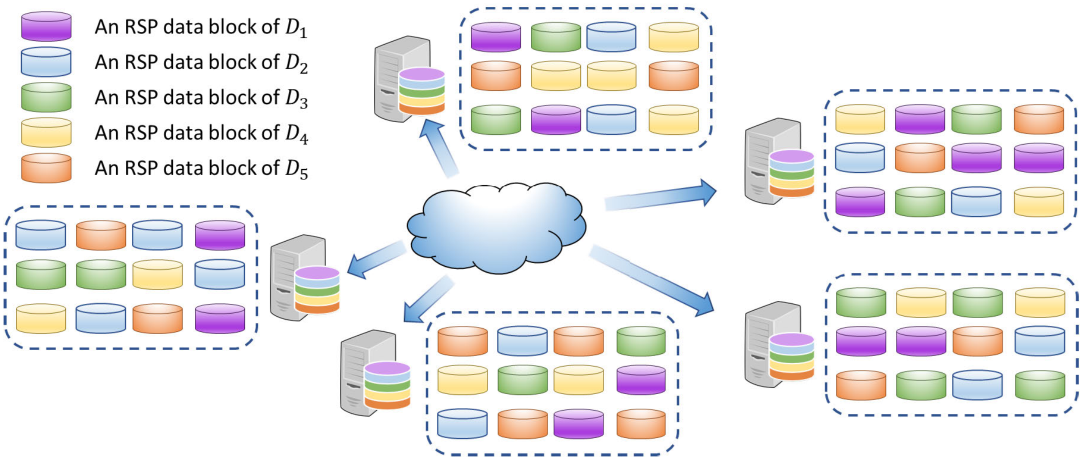
An RSP data block of D_1
An RSP data block of D_2
An RSP data block of D_3
An RSP data block of D_4
An RSP data block of D_5

FIGURE 5. An example of block-level data replication among 5 data centers shows that each data center has some RSP data blocks of each of the original five data files $\{D_1, D_2, D_3, D_4, D_5\}$.

In Strategy 2, we use BDMS to generate RSP data blocks for the local data files in each data center and store the replicated RSP data blocks from other data centers. Due to the replications of RSP data blocks, data analysis can be carried out on any data center without data block downloading as in Strategy 1. Taking the case in Fig. 5 as an example, assume a big data set $\mathfrak{D}$ is stored in five distributed files D_1, D_2, D_3, D_4, D_5 in five data centers. The RSP data blocks of these five files are replicated in all five data centers. To analyze $\mathfrak{D}$, we just choose one data center and randomly select a set of RSP data blocks for each file using block-level sampling function in BDMS and combine the five sets of RSP data blocks in one set of new RSP data blocks as a set of random samples of $\mathfrak{D}$. Then, we can use the new set of RSP data blocks to conduct approximate analysis for $\mathfrak{D}$ in a similar way as in Strategy 1. Besides, this strategy enables us to build an ensemble model from the local models of five data centers. For example, we learn one model in each data center using its own RSP data blocks, we can independently obtain five models f_1, f_2, f_3, f_4. Then, we can choose one data center as the central data center $DC_{central}$ and download other four models from other data centers to $DC_{central}$ as shown in Fig. 6. Finally, we can build an ensemble model $\mathcal{F}$ from five models f_1, f_2, f_3, f_4, f_5.

The steps of data analysis using Strategy 2 are presented as follows:

- **Step 1:** Generating RSP.
  The first step is to partition the data stored in each data center to a set of RSP data blocks, and it is similar to the first step in Strategy 1. Let $\mathcal{R}_i$ be a set of RSP data blocks. BDMS converts $\mathcal{D}_i$ to $\mathcal{R}_i$ in the ith data center as

$$\mathcal{R}_i = RSP(\mathcal{D}_i), \quad \mathcal{D}_i \in \mathfrak{D}$$

  where $i = \{1, 2, .., \mathfrak{u}\}$ and $RSP(.)$ is the conversion function to an RSP data model.
- **Step 2:** Block-level data replication.

Algorithm 3 Data Replication

```
Input:
    $\mathcal{D}$: stored data in the data center;
    $\mathcal{L}$: a set of data centers;
    $\gamma$: replication factor.
Method:
 1: $\mathcal{R} \leftarrow RSP(\mathcal{D})$
 2: for $R \in \mathcal{R}$ do
 3:     $l \leftarrow$ select random $\gamma$ data centers from $\mathcal{L}$
 4:     for $DC \in l$ do
 5:         write $R$ into $DC$
 6:     end for
 7: end for
```

After generating $\mathcal{R}_i$ in the ith data center, the data blocks are replicated into random γ $(0 < \gamma \leq \mathfrak{u})$ data centers where γ is a replication factor. After replication of RSP data blocks, the kth data center stores a collection of subsets of data blocks $\mathfrak{S}_k$ from $\mathfrak{u}$ data centers as

$$\mathfrak{S}_k = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{\mathfrak{u}}\}$$

where $\mathcal{S}_i$ is a set of data blocks from the RSP data model $\mathcal{R}_i$ in the ith data center

$$\mathcal{S}_i \subset \mathcal{R}_i, \quad i = \{1, 2, .., \mathfrak{u}\}$$

According to the RSP theory, $\mathcal{S}_i$ is a random sample of $\mathcal{D}_i$. Thus, $\mathfrak{S}_k$ is a random sample of $\mathfrak{D}$. Algorithm 3 describes the steps of performing data replication in each data center.

- **Step 3:** Data analysis.
  In the kth data center, we randomly select b data blocks from each subset $\mathcal{S}_i \in \mathfrak{S}_k$. Then, these subsets are merged into one set of b new RSP data blocks as

$$\bar{\mathfrak{S}}_k = \{\bar{S}_1, \bar{S}_2, \ldots, \bar{S}_b\}$$

Algorithm 4 Building a Base Learner

```
Input:
    $\mathfrak{S}$: a collection of subsets of the distributed data;
    $\mathfrak{B}$: the base learning algorithm;
    $b$: the sample size.
Output:
    $f$: a base learner model.
Method:
 1: $\mathcal{S}_{\mathfrak{S}} = \phi$
 2: for $j = 1$ to $b$ do
 3:     $S_{combined} \leftarrow$ create an empty RSP data block
 4:     for each $\mathcal{S} \in \mathfrak{S}$ do
 5:         $S_{selected} \leftarrow$ select the $j$th RSP data block from $\mathcal{S}$
 6:         $S_{combined} \leftarrow S_{combined} \cup S_{selected}$
 7:     end for
 8:     append $S_{combined}$ to $\mathcal{S}_{\mathfrak{S}}$
 9: end for
10: $f \leftarrow \mathfrak{B}(\mathcal{S}_{\mathfrak{S}})$
11: return $f$
```

Algorithm 5 Building an Ensemble Learner

```
Input:
    $\mathcal{L}$: a set of data centers;
    $\mathfrak{B}$: the base learning algorithm;
    $b$: the sample size.
Output:
    $\mathcal{F}$: an approximate ensemble model of $\mathfrak{D}$.
Method:
 1: $f_{set} = \phi$
 2: for $l \in \mathcal{L}$ do
 3:     $f \leftarrow$ request the base learner from data center $l$ using
        Algorithm 4($l.data$, $\mathfrak{B}$, $b$)
 4:     append $f$ to $f_{set}$
 5: end for
 6: $\mathcal{F} \leftarrow \Psi(f_{set})$
 7: return $\mathcal{F}$
```

where

$$\bar{S}_j = \bigcup_{i=1}^{\mathfrak{u}} S_j^i$$

which is a random sample data block of $\mathfrak{D}$, according to Theorem 1 in [17]. $\bar{\mathfrak{S}}_k$ is a set of random sample data blocks of $\mathfrak{D}$, each with an empirical distribution similar to the empirical distribution of $\mathfrak{D}$. Therefore, analyzing $\bar{\mathfrak{S}}_k$ gives an approximate result of $\mathfrak{D}$. Hence, in this step, a base learning algorithm $\mathfrak{B}$ is applied to $\bar{\mathfrak{S}}_k$ to build a base learner model f. The steps of building a base model are described in Algorithm 4.

- **Step 4:** Building an ensemble model.
  The $\mathfrak{u}$ base learner models are collected from the different data centers into $DC_{central}$ to build the ensemble model $\mathcal{F}$.

$$\mathcal{F} = \Psi(f_1, f_2, .., f_{\mathfrak{u}})$$

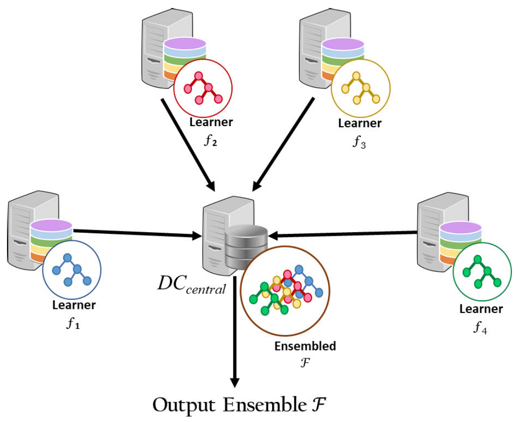

FIGURE 6. Process to build an ensemble model in Strategy 2. In each data center, a learning algorithm is applied to a subset of RSP data blocks, which is selected randomly to build a base learner. After that, the learned models are sent to the central data center to build the ensemble model.

where Ψ is the consensus function of the ensemble model. This step is illustrated in Fig. 6. Algorithm 5 is proposed to operate in $DC_{central}$. Its main function is to request the base models from different data centers first, and then build the ensemble model.

As shown in Step 2, *block-level data replication* enables each data center to have a random sample of the original distributed data $\mathfrak{D}$. The merged data blocks in $\bar{\mathfrak{S}}_k$ have statistical characteristics similar to the original data $\mathfrak{D}$. Therefore, processing $\bar{\mathfrak{S}}_k$ in any data center gives an approximate result as processing $\mathfrak{D}$. Accordingly, Strategy 2 provides us with the ability to estimate or build an approximate model of a vast distributed data on multiple data centers by two ways, 1) analyzing the data in any data center as shown in Step 3, or 2) building an ensemble model as shown in Step 4.

V. SIMULATION RESULTS

In this section, we use simulation results to show the performance of the proposed strategies in big data analysis across geo-distributed data centers. The experiments were conducted on a simulated environment consisting of one local data center located in Shenzhen, China and four remote Amazon Web Services (AWS) data centers in North America, Asia and Australia. In the simulation analysis, we separated the performance of data transfer from performance of data analysis in a data center, as the downloading performance and the analysis performance. Usually, the data downloading operation is only required once in an analysis task and its performance is determined by the network performance. We use the time of data block transfer from remote data centers to demonstrate that the proposed Strategy 1 is feasible in practice.

A. SIMULATION ENVIRONMENT

Our simulation environment consists of five data centers. We used Amazon Web Services (AWS) to set up EC2 micro instances in four AWS data centers in North Virginia (US),

TABLE 1. Characteristics of simulation data sets.

Case	DS name	Total size	N	M	Classes	Remarks
A	DC1	22.5 GB	25,000,000	100	125	The class labels are from 0 to 124
	DC2	22.5 GB	25,000,000	100	125	The class labels are from 125 to 249
	DC3	22.5 GB	25,000,000	100	125	The class labels are from 250 to 374
	DC4	22.5 GB	25,000,000	100	125	The class labels are from 375 to 499
B	DC1	22.5 GB	25,000,000	100	500	70% of the records are of classes 0 to 124
	DC2	22.5 GB	25,000,000	100	500	70% of the records are of classes 125 to 249
	DC3	22.5 GB	25,000,000	100	500	70% of the records are of classes 250 to 374
	DC4	22.5 GB	25,000,000	100	500	70% of the records are of classes 375 to 499
C	DC1	22.5 GB	25,000,000	100	500	These records are selected randomly from the big synthetic data set
	DC2	22.5 GB	25,000,000	100	500	These records are selected randomly from the big synthetic data set
	DC3	22.5 GB	25,000,000	100	500	These records are selected randomly from the big synthetic data set
	DC4	22.5 GB	25,000,000	100	500	These records are selected randomly from the big synthetic data set

Mumbai, Seoul and Sydney. These remote data centers were used to store distributed RSP data blocks and measure the downloading time of remote data files in different sizes.

In the local data center in Shenzhen University, we configured a small cluster with 5 nodes for the simulation, each having 12 cores (24 with Hyper-threading), 128 GB RAM, and 12.5 TB disk storage. The operating system is Ubuntu version 14.04.5. Apache Hadoop 2.6.0 and Apache Spark 2.3.0 are installed on this cluster. Scala version 2.11.12 is used for implementation of the algorithms. The local data center is used as the central data center to demonstrate the performance of big data analysis using the two proposed strategies.

B. DATA GENERATION

In the simulation, we generated one big data set with the form $Z_i = (X_i, Y_k)$, with the identical and independent distribution for $i = \{1, \ldots, N\}$, $\forall Y_k \in (Y_1, \ldots, Y_K) : X_{i,m} = N(\mu_{k,m}, \sigma_{k,m})$, where $m = \{1, \ldots, M\}$, $\mu_{k,m} \in U(0, 10)$ and $\sigma_{k,m} \in U(0, 10)$. We generated the data set with the following parameters: the number of features $(M) = 100$, the number of classes $(K) = 500$, and the number of records $(N) = 100, 000, 000$. The final generated data volume was 90 GB, and the data records are sorted on class labels.

We distributed the generated dataset in four remote data centers in three ways to simulate different situations.

- **Case A**: To simulate different class distributions in different data centers, we divided the original data into 4 different data sets, each having 125 classes. By this division, we ensure the class distribution in one data center is totally different from others. Then, four sets of RSP data blocks were generated from the four datasets using the functions in the BDMS libary RRPlib [19]. These RSP data blocks were uploaded to the four remote data centers.
- **Case B**: In this case, we simulate the unbalanced class distribution; i.e., some classes have more objects than other classes. We divided the original data into 4 data sets as follows. Dataset $DC1$ has 70% of classes $0-124$, and 30% other classes. Similarly, dataset $DC2$ has 70% of classes $125-249$, dataset $DC3$ has 70% of classes $250-374$, and dataset $DC4$ has 70% of classes $375-499$. In the same way as Case A, the four datasets were converted to RSP data blocks and uploaded to the four remote data centers.
- **Case C**: To simulate the balanced class distribution in each data center, we divided the original data into four data sets with equal proportions of 500 classes, converted the four data sets into RSP data blocks, and uploaded them to the four remote data centers.

TABLE 2. The time needed for downloading 1MB files from four AWS data centers to the local data center.

Region	Min (min)	Avg (min)	Max (min)
US East (N. Virginia)	0.34	0.88	2.55
Asia Pacific (Mumbai)	0.53	1.43	5.46
Asia Pacific (Seoul)	0.80	1.55	2.55
Asia Pacific (Sydney)	0.83	5.32	13.08

Table 1 shows the characteristics of the simulation data sets in the three cases.

The generated simulation data were distributed in five data centers in which four remote data centers were used to test the data transfer time and the local one was used for data analysis.

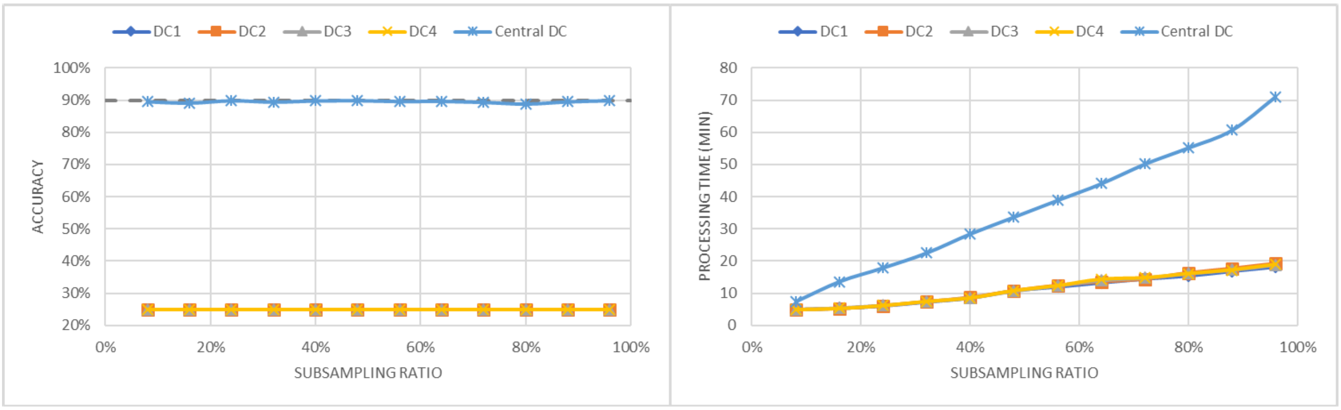

FIGURE 7. Classification results for the synthesized data sets distributed as the case A. The dashed line on the top is the accuracy of running the random forest algorithm on the synthesized data as a whole.

TABLE 3. The time needed for downloading 1GB files from four AWS data centers to the local data center.

Region	Min (min)	Avg (min)	Max (min)
US East (N. Virginia)	7.47	8.87	11.23
Asia Pacific (Mumbai)	7.47	14.33	36.44
Asia Pacific (Seoul)	10.69	15.55	22.93
Asia Pacific (Sydney)	43.01	52.94	67.92

C. PERFORMANCE OF DATA DOWNLOADING FROM REMOTE DATA CENTERS

In this simulation, we used two sizes of data block files of 1MB and 1GB in remote data centers. We downloaded data block files in the two sizes from the remote data centers to the local data center in Shenzhen University. To consider the dynamic situation in the network, the data block files of each size was repeatedly downloaded 100 times, the minimal, average and maximal downloading times from each remote center are reported in Table 2 and Table 3, respectively. We can see that the largest average downloading time is about 53 minutes for the 1GB data block file, which was from the remote data center in Sydney. Since this downloading operation is often carried out once in a data analysis task. This downloading performance for 1GB data file is acceptable in practice. The performance is much better from other remote data centers, where the speed of the global internet is much higher.

D. PERFORMANCE OF DATA ANALYSIS USING STRATEGY 1

In this section, we demonstrate the performances of data analysis across geo-distributed data centers based on Strategy 1 by building classification models from simulation data sets. We considered three data distribution cases in Section V-B. For each case, the performances of execution time and classification accuracy of models built from RSP data blocks of the remote data centers and the local data center were recorded. Random forest algorithm in Spark was used in building the classification models.

Fig. 7-a shows the classification accuracy of models with different percentage of data in each data center. The curves in the bottom show the accuracy of the models built separately from the data set in each remote data center where the simulation data is distributed in Case A. Because each model only covers 25% of the classes, the accuracy of the models is constantly 25%, which indicates the model cannot be applied to the data in other data centers. However, the top curve shows the accuracy of the model built from merged data blocks of all four remote data centers. We can see that the model achieved the accuracy of 90%, because the merged data blocks represent the random sample of the entire simulation data. The curves in Fig. 7-a also show that a small portion of data is enough to build a good model because the accuracy of the models do not increase much as the data increases. This implies that we do not need to download much data from remote data centers.

Fig. 8-a shows similar results of Case B. In Case B, the data set in each remote cluster contains all 500 classes but very unbalanced with 125 dominant classes of 70% and other 375 classes of only 30%. The accuracy of the models from individual remote data centers, as shown in the bottom curves of Fig. 8-a has a small increase to 35%, slightly better than the corresponding models in Fig. 7-a. However, considering the merged data blocks, the accuracy of the model in the top curve is same as the top curve in Fig. 7-a, because the training data sets are essentially same.

Fig. 9-a shows the results of Case C from data sets of four remote data centers with balanced classes. The accuracy of the models from individual remote data centers increased significantly up to 73%. However, this is a rare situation in realty because different data centers usually store data with different distributions. We can see that with the merged data blocks, the accuracy of the model is further increased to 90%, same as other cases. The reason is that the size of data blocks increased so the performance of individual trees is increased which results in a better model of random forest.

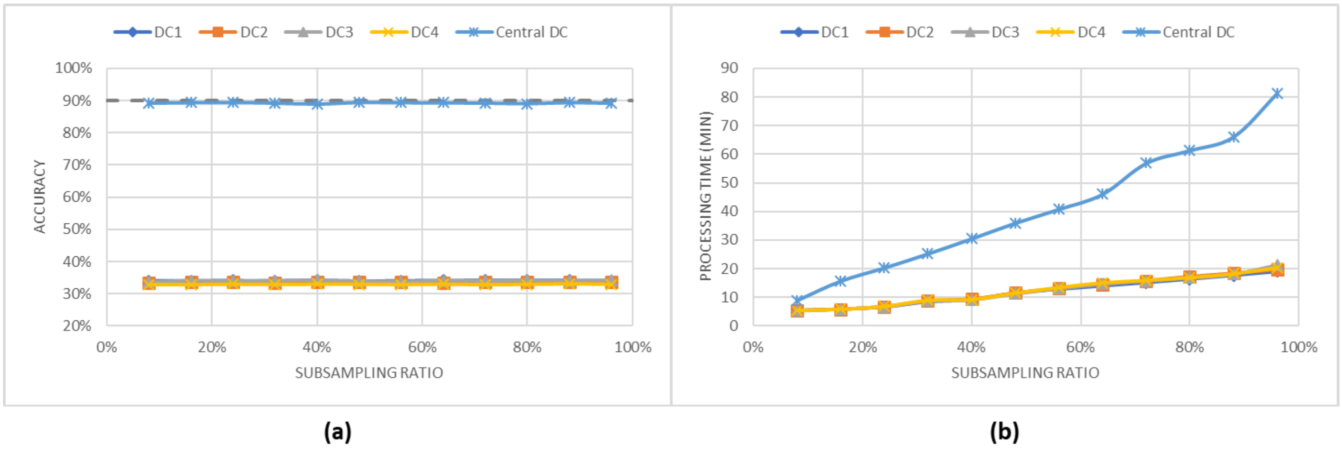

FIGURE 8. Classification results for the synthesized data sets distributed as the case B. The dashed line on the top is the accuracy of running the random forest algorithm on the synthesized data as a whole.

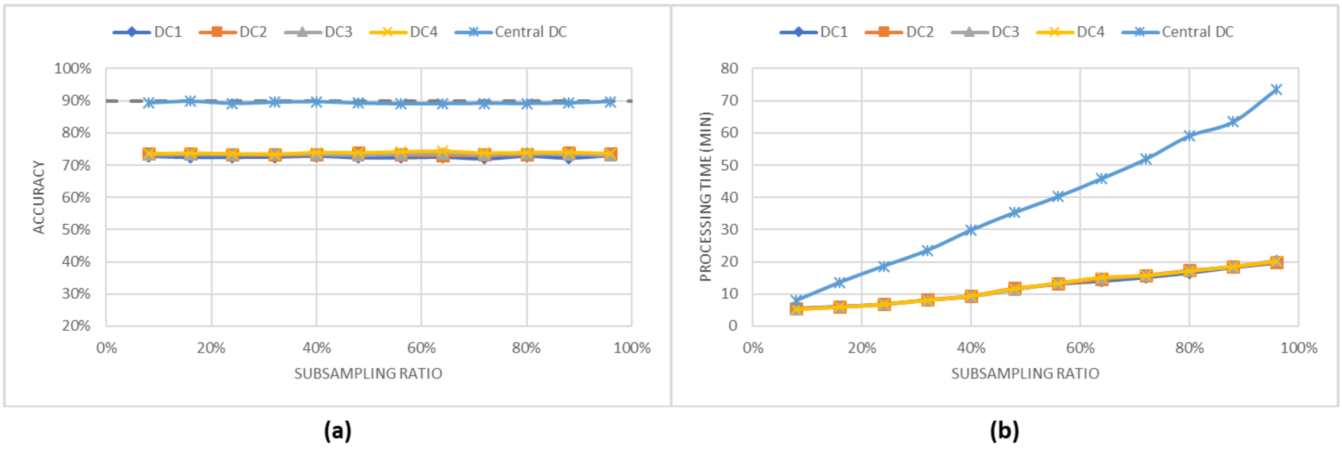

FIGURE 9. Classification results for the synthesized data sets distributed as the case C. The dashed line is the accuracy of running the random forest algorithm on the synthesized data as a whole.

Fig. 7-b, Fig. 8-b and Fig. 9-b show similar patterns of the computation performance of building models in three cases. We can observe that the execution time increases linearly. For instance, the execution times for processing 10% and 100% of the data in the central data center are about 10 min and 70 min, respectively. The difference between the processing times of the combined data on the central data center and other data centers is due to the data ratio. For example, the processing time for analyzing 50% of the data in each data center (which is equal to 2.3 GB) is around 11 min while the processing time for analyzing the same percentage on the central data center (which is equal to 9 GB) is 35 min, i.e., the increase of processing time is due to the increase in the data size.

E. PERFORMANCE OF DATA ANALYSIS USING STRATEGY 2

In this section, we demonstrate the performances of data analysis across geo-distributed data centers using Strategy 2. Similarly, the random forest algorithm was used in building the classification models from simulation data sets in three cases. The performance is demonstrated as the execution time and classification accuracy of models. Following the steps of Strategy 2, we conducted the experiments in the steps below:

(i) We used AWS to set up four remote data centers in N. Virginia, Mumbai, Seoul, and Sydney and upload the four data sets {DC1, DC2, DC3, DC4} to them. The configuration of these remote data centers was made same as the small cluster in the local data center in Shenzhen.

(ii) We replicated the RSP data blocks of each data center to other data centers using the proposed block-level data replication method and made four sets of new data blocks {newDC1, newDC2, newDC3, newDC4} in each data center.

(iii) For each new data set, we built a classification model using a random forest algorithm from some randomly selected data blocks.

(iv) We downloaded the four random forest models from the remote data centers to the local data center, as shown in Fig. 6, and used the voting method as a consensus function to ensemble these four models as the ensemble model for classification.

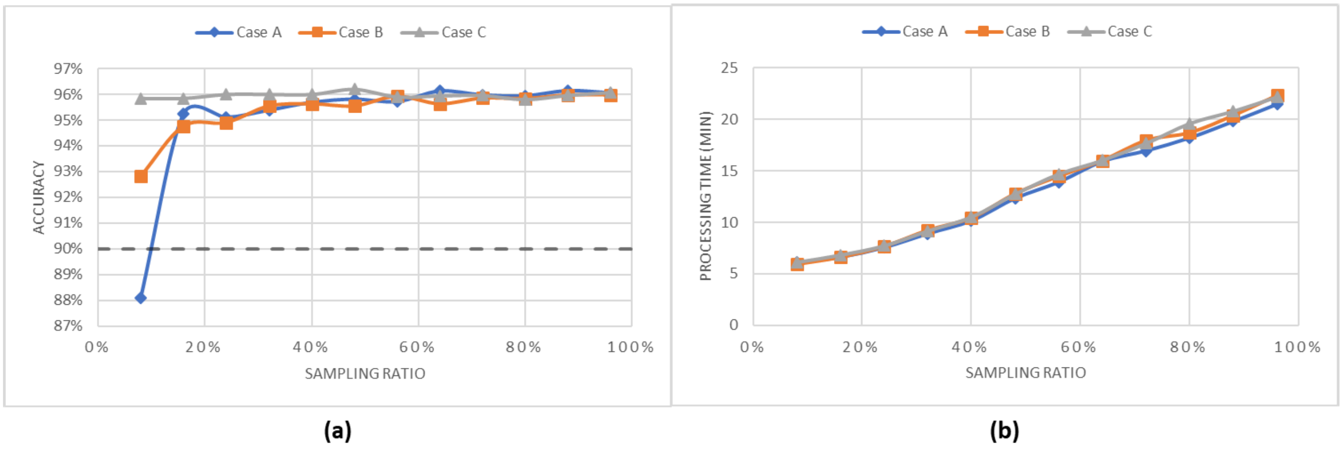

FIGURE 10. **Classification results for the synthesized data sets considering the ensemble strategy. The dashed line is the accuracy of processing random forest algorithm on the synthesized data as a whole.**

Fig. 10-a shows the accuracy of the ensemble models from three cases. When 20% of the simulation data was used in samples to build the ensemble models, the accuracies of the models are approximately between 95% and 96%. This result shows that the ensemble models performed better than the results from Strategy 1. The dashed line in Fig. 10-a is the result of the random forest model built with the entire simulation data. We can see the models with the merged data blocks in Strategy 1 achieved the same accuracy of the single model from the entire data.

Fig. 10-b shows the performance of execution time which does not include the downloading time. We can see in all three cases that the processing time for building models from 20% of the data is around 7 min and the processing time increases linearly as more data are used in building models. In building ensemble models, the time to download the remote random forest models to the central data center to build the ensemble model is called *latency* which is essential in such cases. Hence, to accomplish our simulation, we performed an experiment on Amazon Web Services (AWS) to simulate the latency, same way as discussed in Section V-A. In the simulation, the model size was approximately 2MB. According to Table 2, it took less than 11 min to transfer the model files from the different AWS remote data centers to the local data center.

VI. DISCUSSION

This paper addresses a new challenge of wide-area big data analytics. In this paper, we focus on approximate big data analysis across geo-distributed data centers. We propose two strategies for data analysis in two data distribution scenarios, with or without replications, as explained in Section IV.

Strategy 1 avails in case a decision-maker needs to see a glimpse into the data; therefore, downloading some RSP blocks from each data center is enough to get an approximate estimation. Moreover, assume that multiple operations are needed to be performed on the data. Instead of repeating these operations on the whole data by distributing jobs and tasks to different data centers, it is more efficient to have some RSP blocks downloaded first to a local data center and then analyzed. Also, this approach separates the data storage from data analysis.

On the other hand, managing data across different data centers using Strategy 2 leads to many advantages, including: (i) *The high availability and low latency*. Copying data to different geo-distributed data centers increases the response times for users who are geographically separated. (ii) *A disaster recovery plan* [34]. (iii) *Fast ensemble estimation*. In this strategy, the distributed data is firstly converted to the RSP data blocks, which are replicated to different data centers. This operation arranges the replicated data on each data center to be a random sample of the whole data, as shown in Section IV-C. The distributed random samples of the big data support efficient approximate analysis and enable distributed ensemble learning or estimation. Since the local models of remote data centers are transferred to the local data center, it is more efficient compared to transfer data in current methods.

Finally, we acknowledge that the limitation of this approach is the time cost and bandwidth consumption in data replication between geo-distributed data centers. However, we only replicate a subset of RSP data blocks in each data center and the data replication is carried out only once and can be done automatically during the time the network is not heavily loaded. In practice, this step may not generate a big cost for business. In future work, we will explore streaming data and scheduling data replication among multiple data centers.

VII. CONCLUSION

In this article, we have proposed two strategies to support the approximate big analysis of distributed data across multiple data centers. In both strategies, we store the data on each data center as a set of RSP data blocks. In the first strategy, some data blocks are required to download from the remote data centers to a central data center for approximate analysis of the big data as a whole. The main advantage of this strategy is to separate the storage level from the analysis level. In the second strategy, we consider data replication among different data centers. As a result of data replication, the data in each data center forms as a random sample of the whole distributed data. The experimental results show that a sample of the data on each data center is enough to be representative of the whole distributed data. In the future, we will continue to study the effect of the BDMS system to solve streaming data problems.

REFERENCES

[1] *For the Airline Industry, Big Data is Cleared for Take-Off*. Accessed: Dec. 13, 2018. [Online]. Available: http://fortune.com/2014/06/19/big-data-airline-industry/

[2] *Apache Hadoop*. Accessed: Dec. 13, 2018. [Online]. Available: https://hadoop.apache.org/

[3] M. Zaharia, "Apache spark: A unified engine for big data processing," *Commun. ACM*, vol. 59, no. 11, pp. 56–65, 2016.

[4] M. S. Mahmud, J. Z. Huang, S. Salloum, T. Z. Emara, and K. Sadatdiynov, "A survey of data partitioning and sampling methods to support big data analysis," *Big Data Mining Anal.*, vol. 3, no. 2, pp. 85–101, Jun. 2020.

[5] S. Ji and B. Li, "Wide area analytics for geographically distributed datacenters," *Tsinghua Sci. Technol.*, vol. 21, no. 2, pp. 125–135, Apr. 2016.

[6] S. Dolev, P. Florissi, E. Gudes, S. Sharma, and I. Singer, "A survey on geographically distributed big-data processing using MapReduce," *IEEE Trans. Big Data*, vol. 5, no. 1, pp. 60–80, Mar. 2019.

[7] A. Vulimiri, C. Curino, P. Brighten Godfrey, T. Jungblut, J. Padhye, and G. Varghese, "Global analytics in the face of bandwidth and regulatory constraints," in *Proc. 12th USENIX Conf. Networked Syst. Design Implement.*, Berkeley, CA, USA, 2015, pp. 323–336.

[8] A. Vulimiri, C. Curino, P. B. Godfrey, T. Jungblut, K. Karanasos, J. Padhye, and G. Varghese, "WANalytics: Geo-distributed analytics for a data intensive world," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, New York, NY, USA, 2015, pp. 1087–1092.

[9] K. Kloudas, M. Mamede, N. Preguiça, and R. Rodrigues, "Pixida: Optimizing data parallel jobs in wide-area data analytics," *Proc. VLDB Endowment*, vol. 9, no. 2, pp. 72–83, Oct. 2015.

[10] L. Chen, S. Liu, B. Li, and B. Li, "Scheduling jobs across geo-distributed datacenters with max-min fairness," *IEEE Trans. Netw. Sci. Eng.*, vol. 6, no. 3, pp. 488–500, Jul. 2019.

[11] H. Jiang, H. E, and M. Song, "Multi-prediction based scheduling for hybrid workloads in the cloud data center," *Cluster Comput.*, vol. 21, no. 3, pp. 1607–1622, Sep. 2018.

[12] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, "Volley: Automated data placement for geo-distributed cloud services," in *Proc. 7th USENIX Conf. Networked Syst. Design Implement.*, Berkeley, CA, USA, 2010, p. 2.

[13] Q. Pu, G. Ananthanarayanan, P. Bodik, S. Kandula, A. Akella, P. Bahl, and I. Stoica, "Low latency geo-distributed data analytics," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 421–434, Sep. 2015.

[14] S. Liu, H. Wang, and B. Li, "Optimizing shuffle in wide-area data analytics," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 560–571.

[15] S. Salloum, Y. He, J. Z. Huang, X. Zhang, T. Z. Emara, C. Wei, and H. He, "A random sample partition data model for big data analysis," 2017, *arXiv:1712.04146*. [Online]. Available: http://arxiv.org/abs/1712.04146

[16] S. Salloum, J. Z. Huang, and Y. He, "Random sample partition: A distributed data model for big data analysis," *IEEE Trans. Ind. Informat.*, vol. 15, no. 11, pp. 5846–5854, Nov. 2019.

[17] C. Wei, S. Salloum, Z. Tamer Emara, X. Zhang, J. Z. Huang, and Y. He, "A two-stage data processing algorithm to generate random sample partitions for big data analysis," in *Cloud Computing*. Cham, Switzerland: Springer, 2018, pp. 347–364.

[18] T. Z. Emara and J. Z. Huang, "A distributed data management system to support large-scale data analysis," *J. Syst. Softw.*, vol. 148, pp. 105–115, Feb. 2019.

[19] T. Z. Emara and J. Z. Huang, "RRPlib: A spark library for representing HDFS blocks as a set of random sample data blocks," *Sci. Comput. Program.*, vol. 184, Oct. 2019, Art. no. 102301.

[20] F. Nargesian, E. Zhu, J. Renée Miller, Q. Ken Pu, and C. Patricia Arocena, "Data Lake Management: Challenges and Opportunities," *Proc. VLDB Endowment*, vol. 12, no. 12, pp. 1986–1989, Aug. 2019.

[21] C.-C. Hung, L. Golubchik, and M. Yu, "Scheduling jobs across geo-distributed datacenters," in *Proc. 6th ACM Symp. Cloud Comput.*, New York, NY, USA, 2015, pp. 111–124.

[22] C.-C. Hung, G. Ananthanarayanan, L. Golubchik, M. Yu, and M. Zhang, "Wide-area analytics with multiple resources," in *Proc. 13th EuroSys Conf.*, New York, NY, USA, 2018, pp. 1–16.

[23] Z. Hu, B. Li, and J. Luo, "Flutter: Scheduling tasks closer to data across geo-distributed datacenters," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[24] D. Boru, D. Kliazovich, F. Granelli, P. Bouvry, and A. Y. Zomaya, "Energy-efficient data replication in cloud computing datacenters," *Cluster Comput.*, vol. 18, no. 1, pp. 385–402, Mar. 2015.

[25] N. K. Gill and S. Singh, "Dynamic cost-aware re-replication and rebalancing strategy in cloud system," in *Advances in Intelligent Systems and Computing*, vol. 328, S. C. Satapathy, B. N. Biswal, S. K. Udgata, and J. K. Mandal, Eds. Cham, Switzerland: Springer, 2015, pp. 39–47.

[26] J. Liu and H. Shen, "A popularity-aware cost-effective replication scheme for high data durability in cloud storage," in *Proc. IEEE Int. Conf. Big Data*, Dec. 2016, pp. 384–389.

[27] N. Mansouri and M. M. Javidi, "A new prefetching-aware data replication to decrease access latency in cloud environment," *J. Syst. Softw.*, vol. 144, pp. 197–215, Oct. 2018.

[28] P. Matri, M. S. Pérez, A. Costan, L. Bougé, and G. Antoniu, "Keeping up with storage: Decentralized, write-enabled dynamic geo-replication," *Future Gener. Comput. Syst.*, vol. 86, pp. 1093–1105, Sep. 2018.

[29] Y. Mansouri, A. N. Toosi, and R. Buyya, "Cost optimization for dynamic replication and migration of data in cloud data centers," *IEEE Trans. Cloud Comput.*, vol. 7, no. 3, pp. 705–718, Jul. 2019.

[30] S. Limam, R. Mokadem, and G. Belalem, "Data replication strategy with satisfaction of availability, performance and tenant budget requirements," *Cluster Comput.*, vol. 22, no. 4, pp. 1199–1210, Jan. 2019.

[31] G. DeCandia, "Dynamo: Amazon's highly available key-value store," *ACM SIGOPS Oper. Syst. Rev.*, vol. 41, no. 6, pp. 205–220, 2007.

[32] R. Sumbaly, J. Kreps, L. Gao, A. Feinberg, C. Soman, and S. Shah, "Serving large-scale batch computed data with project voldemort," in *Proc. 10th USENIX Conf. File Storage Technol.*, Berkeley, CA, USA, 2012, p. 18.

[33] E. B. Edwin, P. Umamaheswari, and M. R. Thanka, "An efficient and improved multi-objective optimized replication management with dynamic and cost aware strategies in cloud computing data center," *Cluster Comput.*, vol. 22, no. S5, pp. 11119–11128, Nov. 2017.

[34] T. Dunning and F. Ellen *Data Where You Want It*. Newton, MA, USA: O'Reilly Media, 2017.

TAMER Z. EMARA (Member, IEEE) received the B.S. degree from Tanta University, Egypt, in 2005, and the M.S. degree from Mansoura University, Egypt, in 2015. He is currently pursuing the Ph.D. degree with the Big Data Institute, Shenzhen University, China. He is currently a Lecturer with the Higher Institute of Engineering and Technology, Kafrelsheikh, Egypt. His main research interest includes big data management. He is a member of the ACM.

JOSHUA ZHEXUE HUANG was born in July 1959. He received the Ph.D. degree from the Royal Institute of Technology, Sweden, in June 1993. He is currently a Distinguished Professor with the College of Computer Science and Software Engineering, Shenzhen University. He is also the Director of the Big Data Institute and the Deputy Director of the National Engineering Laboratory for Big Data System Computing Technology. He has published over 200 research papers in conferences and journals. His main research interests include big data technology and its applications. He received the first PAKDD Most Influential Paper Award in 2006. He is known for his contributions to the development of a series of k-means type clustering algorithms in data mining, such as k-modes, fuzzy k-modes, k-prototypes, and w-k-means that are widely cited and used, and some of which have been included in commercial software. He has extensive industry expertise in business intelligence and data mining, and has been involved in numerous consulting projects in Australia, Hong Kong, Taiwan, and mainland China.