Random Sample Partition: A Distributed Data Model for Big Data Analysis

Salman Salloum, Joshua Zhexue Huang, and Yulin He

***Abstract*—With the ever-increasing volume of data, alternative strategies are required to divide big data into statistically consistent data blocks that can be used directly as representative samples of the entire data set in big data analysis. In this paper, we propose the Random Sample Partition (RSP) distributed data model to represent a big data set as a set of disjoint data blocks, called RSP blocks. Each RSP block has a probability distribution similar to that of the entire data set. RSP blocks can be used to estimate the statistical properties of the data and build predictive models without computing the entire data set. We demonstrate the implications of the RSP model on sampling from big data and introduce a new RSP-based method for approximate big data analysis which can be applied to different scenarios in the industry. This method significantly reduces the computational burden of big data and increases the productivity of data scientists.**

***Index Terms*—Approximate computing, big data analysis, cluster computing, data partitioning, random sampling.**

I. Introduction

A. Motivation

As DATA volume in different industrial areas goes beyond the petabyte scale, big data analysis is becoming a challenging problem to data scientists in companies with small computing clusters. Although the divide-and-conquer paradigm is employed to scale iterative data analysis and mining algorithms to big data on computing clusters [1]–[5], the scalability of these algorithms is limited to the available resources. A common remedy to this problem is approximate computing [6], [7], where samples of data are used to get approximate results at lower costs [8]–[15]. However, sampling on computing clusters becomes inefficient with the increasing volume of distributed data. This is prohibitive if multiple random samples are required in statistical analysis and diagnostics [16]–[18].

Manuscript received October 19, 2018; revised January 17, 2019; accepted April 2, 2019. Date of publication May 6, 2019; date of current version November 5, 2019. This was supported by the National Natural Science Foundation of China under Grant 61473194. Paper no. TII-18-2736. *(Corresponding author: Joshua Zhexue Huang.)*

The authors are with the National Engineering Laboratory for Big Data System Computing Technology, Shenzhen University, Shenzhen 518060, Guangdong, China, and also with the Big Data Institute, College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, Guangdong, China (e-mail: ssalloum@szu.edu.cn; zx.huang@szu.edu.cn; yulinhe@szu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2019.2912723

In fact, the mainstream cluster computing frameworks and engines (e.g., Apache Hadoop,[1] Apache Spark,[2] and Microsoft R Server[3]) implement a shared-nothing architecture[4] where each node is independent in terms of both data and resources. The MapReduce computing model [5] is adapted to distribute data and computation. Hadoop Distributed File System (HDFS) [19] organizes and replicates the data as small distributed data blocks. In this architecture, Record-Level Sampling (RLS) from an HDFS file becomes time-consuming because selecting records with equal probability requires scanning the entire data. Block-Level Sampling (BLS) can be more efficient, but the results from block-level samples may not be as good as those from record-level samples [20]–[23]. The problem occurs because HDFS doesn't consider the statistical properties when distributing big data on computing clusters. Consequently, using block samples in big data analysis can produce biased or even statistically incorrect results. Since data partitioning has a crucial impact on the performance of iterative algorithms [12], [24]–[26], we propose a new idea to make HDFS blocks as ready-to-use random samples of the entire data set to improve the quality of BLS and the performance of approximate big data analysis.

B. Our Contributions

In this paper, we propose the Random Sample Partition (RSP) distributed data model to facilitate BLS and support big data analysis. Our objective is to enable the distributed data blocks of a big data set to be used directly as random samples in approximate big data analysis. In this model, a big data set is represented as a set of small disjoint random sample data blocks, called RSP blocks. The probability distribution in each RSP block is similar to that in the entire data set. Thus, an RSP block is equivalent to a record-level sample from the entire data. A *two-stage data partitioning method* is developed to generate an RSP from an HDFS file. In practice, an RSP is generated offline, and only once, on a computing cluster. With the RSP model, block-level samples become as good as record-level samples, but sampling RSP blocks is efficient. The experimental results have shown that the sample statistics and distributions from RSP blocks are equivalent to those from record-level samples, but significantly better than those from normal HDFS blocks.

[1] [Online]. Available: http://hadoop.apache.org/
[2] [Online]. Available: https://spark.apache.org/
[3] [Online]. Available: https://www.microsoft.com/en-us/cloud-platform/r-server
[4] [Online]. Available: https://www.oreilly.com/learning/processing-data-in-hadoop

The time to take multiple random samples from big data is reduced to seconds.

Given the statistical advantages of RSP blocks, the RSP model has significant implications on the efficiency of big data analysis. It enables a new method to analyze big data on small computing clusters, called the *RSP-based method for approximate big data analysis*. This method employs a stepwise process to obtain approximate results using block-level samples from an RSP. First, a block-level sample is selected from the RSP. The number of selected blocks is set according to the available resources. Second, a sequential algorithm is applied in parallel to each selected RSP block. Third, the outputs of these blocks are combined to produce an approximate result for the entire data. The three steps can be repeated to improve the result incrementally. To test this method, we conducted experiments, using three real and three synthetic data sets up to 1TB, on a small computing cluster with only 640GB of memory. The results show that a few RSP blocks are sufficient to obtain estimates and models which are equivalent to those computed from the entire data set. This method can also be extended to different scenarios where the entire data can't be computed, e.g., cross-data centers and evolving data sets.

The main contributions of this paper are as follows.

1) We propose RSP, a distributed data model that preserves the statistical properties of the data set in each of its distributed data blocks.
2) We empirically show the computational and statistical advantages of BLS from an RSP.
3) We introduce the RSP-based method for approximate big data analysis, demonstrate its performance on a small computing cluster, and discuss extensions to cross-data centers and evolving data sets.

The remainder of this paper is organized as follows. We start with a summary of related work in Section II. In Section III, we propose the RSP model, including the formal definitions and a partitioning method. After that, we define BLS from an RSP and demonstrate its statistical and computational advantages in Section IV. Then, we introduce the RSP-based method for approximate big data analysis and discuss its performance and extensions in Section V. Section VI concludes this paper.

II. Related Work

In this section, we first review cluster computing frameworks and show how the RSP model complements them. Then, we discuss the differences between the RSP-based method and other approaches in approximate big data analysis.

A. Cluster Computing Frameworks

Cluster computing frameworks with a shared-nothing architecture have been adopted to scale iterative algorithms to big data [27]. To eliminate the cost of reading/writing intermediate data blocks after each iteration in Hadoop MapReduce, Apache Spark introduces an in-memory computing model using the Resilient Distributed Datasets (RDDs) [28]. Microsoft R Server introduces the Parallel External Memory Algorithms (PEMA) for statistical analysis with the on-disk eXternal Data Frame (XDF) format to address the in-memory limitation in R. High-level libraries of data analysis and machine learning algorithms were developed for applications such as Mahout,[5] MLlib,[6] and MicrosoftML.[7] To enable data scientists to work with distributed data and algorithms using their preferred languages, e.g., R and Python, some libraries were also developed in these languages such as RHadoop,[8] RHIPE,[9] PySpark,[10] SparkR,[11] sparklyer,[12] and RevoScaleR. It is also possible to apply sequential algorithms in parallel to individual data blocks to reveal deeper insights as in RHIPE [29]. This is a key functionality to scale existing sequential algorithms to big data on a computing cluster without developing parallel implementations of these algorithms. However, the exploration and combination of the block-level results becomes a challenge to data scientists due to the large number of HDFS blocks in a big data set and the inconsistency of data distributions in these blocks.

The RSP-based method solves this problem by making the distributed data blocks as random samples of the entire data set, using block-level samples to get approximate results from a few data blocks, and supporting the incremental update of the result according to the available resources. Nevertheless, the RSP model is structurally similar to existing data models in HDFS, RDD, and XDF. Thus, the new model doesn't preclude applying existing parallelized algorithms to the entire data in the mainstream cluster computing frameworks. The RSP model can be implemented as extensions to existing frameworks and libraries, and to enable approximate big data analysis on small clusters with less memory.

B. Approximate Big Data Analysis

The RSP-based method has two main differences from existing frameworks for approximate big data analysis. *First, the entire data is stored as ready-to-use disjoint random sample data blocks. The RSP generation is an offline operation.* This is different from ApproxHadoop [8] that uses online multistage sampling from HDFS blocks. HDFS blocks may not be random samples and online sampling is expensive in HDFS due to the communication, memory, and I/O costs. *Second, the RSP-based method targets at offline data analysis workloads where data scientists use a variety of algorithms in exploratory data analysis, statistical estimation, and predictive modeling to infer global statistical properties and patterns from large data volumes.* This is different from distributed approximate query processing engines such as BlinkDB [10] that uses offline stratified sampling on frequently occurring columns and uniform sampling to support ad-hoc queries. Our method is also different from streaming data analysis frameworks such as IncApprox [9] that uses online stratified sampling to produce an incre-mentally updated approximate output from streaming

[5][Online]. Available: https://mahout.apache.org/
[6][Online]. Available: https://spark.apache.org/mllib
[7][Online]. Available: https://docs.microsoft.com/en-us/machine-learning-server
[8][Online]. Available: https://github.com/RevolutionAnalytics/RHadoop
[9][Online]. Available: http://deltarho.org/docs-datadr/index.html
[10][Online]. Available: spark.apache.org/docs/2.2.0/api/python/pyspark.html
[11][Online]. Available: https://spark.apache.org/docs/latest/sparkr.html
[12][Online]. Available: https://spark.rstudio.com

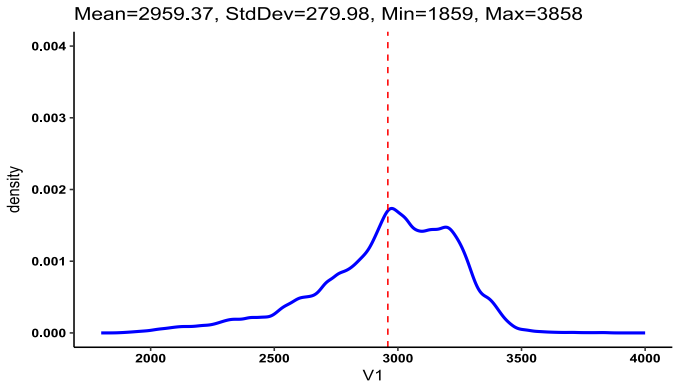


Fig. 1. Distribution of V1 in Covertype data.

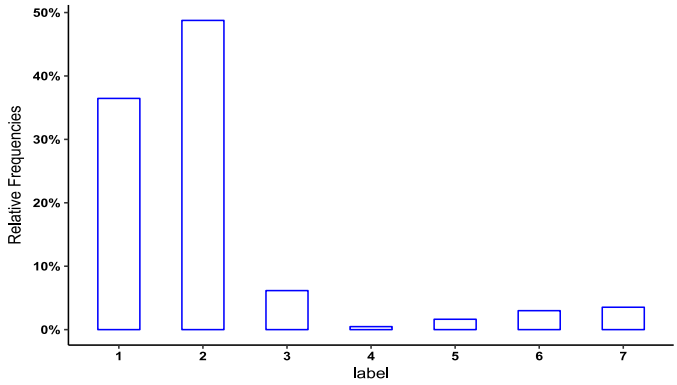


Fig. 2. Frequency distribution of classes in Covertype data.

data. On the other hand, RSP blocks can be used directly as subsamples in statistical estimation procedures such as the Bag of Little Bootstraps (BLB) [16].

III. Random Sample Partition of Big Data

In this section, we introduce the formal definitions of the RSP distributed data model and a method to generate an RSP from an HDFS file.

A. Definitions

Assume that $\mathbb{D}$ is a multivariate data set of N records and M features where N is very large so $\mathbb{D}$ cannot be analyzed on a single machine. With the RSP model, $\mathbb{D}$ is divided into K small disjoint random sample data blocks in advance on a computing cluster. We first define a random sample of $\mathbb{D}$ as follows:

Definition 1: Random Sample: Let D be a subset of $\mathbb{D}$ containing n records chosen from $\mathbb{D}$ using a random process. D is a random sample of $\mathbb{D}$ if

$$E[F(x)] = \mathbb{F}(x)$$

where x is a random variable represented by a feature in $\mathbb{D}$. $F(x)$ and $\mathbb{F}(x)$ denote the sample distribution functions (s.d.f.) of x in D and $\mathbb{D}$, respectively. $E[F(x)]$ denotes the expectation of $F(x)$.

In cluster computing frameworks, $\mathbb{D}$ is divided into small disjoint HDFS blocks, but these blocks don't have similar statistical properties as $\mathbb{D}$. To show the difference of distributions in HDFS blocks, we created a partition of the Covertype[13] data set by sequentially cutting the data file into 12 blocks (similar to the process to import a big data file into HDFS). Fig. 1 shows the distribution of feature V1 of the entire data set, where V1 is a numeric feature (Mean = 2959.36 and Std. Dev = 279.98) representing the elevation (in meters) of the forest. Fig. 2 shows the frequency distributions of seven classes of the entire data set. We randomly selected four blocks and plotted their distributions of feature V1 and classes, respectively, as shown in Figs. 3 and 4. We can see that the distributions of the same feature in four

[13][Online]. Available: https://archive.ics.uci.edu/ml/datasets/Covertype

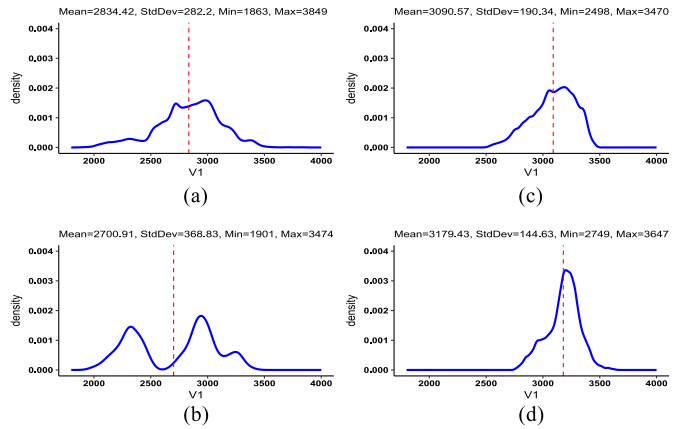


Fig. 3. Distributions of V1 in four blocks from the HDFS file of Covertype data. The vertical dashed line is the mean value. (a) V1 in Block 1. (b) V1 in Block 6. (c) V1 in Block 9. (d) V1 in Block 11.

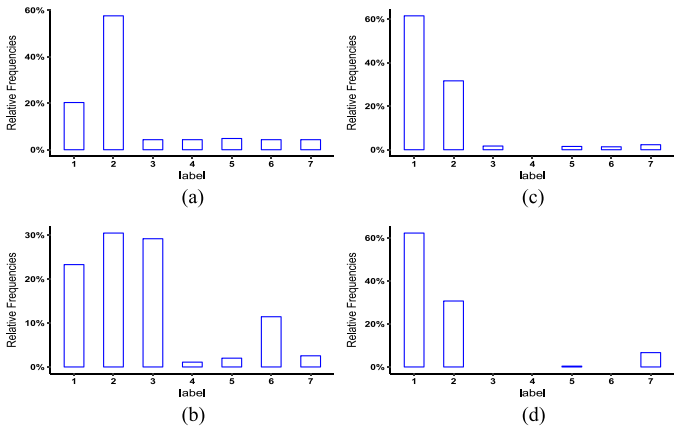


Fig. 4. Frequency distributions of classes in four blocks from the HDFS file of Covertype data. (a) Classes in Block 1. (b) Classes in Block 6. (c) Classes in Block 9. (d) Classes in Block 11.

blocks are different. They are also different from the distributions of the entire data set. According to Definition 1, these blocks are not random samples of the entire data set.

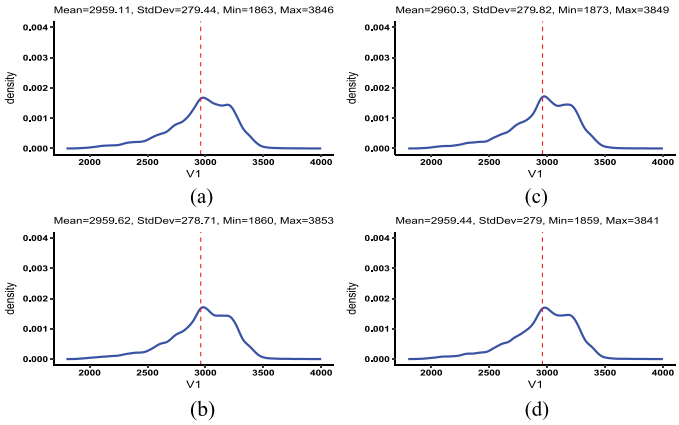


Fig. 5. Distributions of V1 in four RSP blocks from a random sample partition of Covertype data. The vertical dashed line is the mean value. (a) V1 in RSP Block 1. (b) V1 in RSP Block 6. (c) V1 in RSP Block 9. (d) V1 in RSP Block 11.

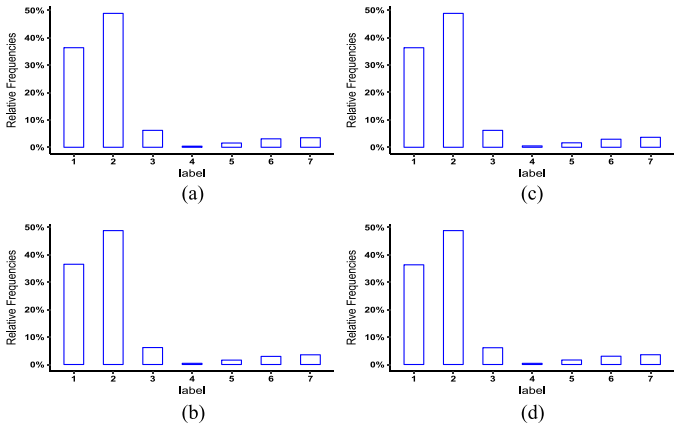


Fig. 6. Frequency distributions of classes in four RSP blocks from a random sample partition of Covertype data. (a) Classes in RSP Block 1. (b) Classes in RSP Block 6. (c) Classes in RSP Block 9. (d) Classes in RSP Block 11.

Mathematically, the set of HDFS blocks of $\mathbb{D}$ represents a partition[14] of $\mathbb{D}$, but HDFS blocks are not necessarily random samples of $\mathbb{D}$. In the new data model, we represent $\mathbb{D}$ as a *random sample partition* defined as follows:

Definition 2: Random Sample Partition: Assume $\mathbb{F}(x)$ is the sample distribution function of a random variable x in $\mathbb{D}$. Let $\mathbb{T}$ be a partition operation which divides $\mathbb{D}$ into a set of subsets $\mathrm{T} = \{D_1, D_2, \ldots, D_K\}$, 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 $i, j \in \{1, 2, \ldots, K\}$ and $i \neq j$;
3) $E[F_k(x)] = \mathbb{F}(x)$ for $k \in \{1, 2, \ldots, K\}$ where $F_k(x)$ denotes the sample distribution function of x in D_k and $E[F_k(x)]$ denotes its expectation.

Accordingly, each D_k is called an RSP block of $\mathbb{D}$, and $\mathbb{T}$ is called an RSP operation on $\mathbb{D}$. We restate Corollary 1 in [30] as Theorem 1 below to ensure that $\mathbb{D}$ can be divided into a set of RSP blocks.

Theorem 1: For a big data set $\mathbb{D} = \{\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_{K \times n}\}$ of N records where $N = K \times n$, randomly choose a permutation of the sequence $\{1, 2, \ldots, K \times n\}$, and denote it as $\tau = \{\tau_1, \tau_2, \ldots, \tau_{K \times n}\}$. For each $k = 1, \ldots, K$, set, $D_k = \{\mathrm{x}_{\tau_{n \times (k-1)+1}}, \mathrm{x}_{\tau_{n \times (k-1)+2}}, \ldots, \mathrm{x}_{\tau_{n \times k}}\}$, then each D_k is an RSP block of $\mathbb{D}$.

Using Theorem 1, we created an RSP from the Covertype data set by randomizing the order of its records (i.e., making the records independently and identically distributed or i.i.d), and sequentially dividing these randomized records into 12 RSP blocks. We randomly selected four RSP blocks and plotted the distributions of the same feature V1 and classes. The plots are shown in Figs. 5 and 6. From these figures, we can see that the distributions of the same feature in the four blocks are similar to each other. They are also similar to the distribution of the entire data set as shown in Figs. 1 and 2. These RSP blocks represent random samples of the entire data set.

[14][Online]. Available: https://en.wikipedia.org/wiki/Partition_of_a_set

B. RSP Generation

Big data files are often stored as HDFS files on Hadoop cluster platforms. To generate an RSP from an HDFS file, a two-stage data partitioning method is developed as the partition operation $\mathbb{T}$ on big data $\mathbb{D}$ for computing clusters. The two stages of the data partitioning method are as follows [30].

1) *Stage 1:* Divide $\mathbb{D}$ into P nonoverlapping subsets $\{D_1, D_2, \ldots, D_P\}$ of equal size (e.g., P is the number of HDFS blocks in $\mathbb{D}$). Randomize each subset D_p into i.i.d. as D'_p and cut it into an RSP of $\{D'_{p,1}, D'_{p,2}, \ldots, D'_{p,K}\}$ independently to generate P RSPs of $\{D'_{1,1}, D'_{1,2}, \ldots, D'_{1,K}\}, \{D'_{2,1}, D'_{2,2}, \ldots, D'_{2,K}\}, \ldots, \{D'_{P,1}, D'_{P,2}, \ldots, D'_{P,K}\}$.
2) *Stage 2:* From each RSP $\{D'_{p,1}, D'_{p,2}, \ldots, D'_{p,K}\}$ for $1 \leq p \leq P$, select its corresponding RSP block $D'_{p,k}$ for $1 \leq k \leq K$ to generate a new data block by merging the set of $\{D'_{1,k}, D'_{2,k}, \ldots, D'_{P,k}\}$ into data block D''_k. Repeat this merging operation K times to generate a new partition $\{D''_1, D''_2, \ldots, D''_K\}$, which is an RSP of $\mathbb{D}$ (See Theorem 1 in [30].) The generated RSP is stored as an HDFS file, called an HDFS-RSP file.

We tested this partitioning method on a small Spark cluster of five nodes (each node has 24 cores, 128 GB RAM, and 12.5 TB disk storage). Fig. 7 shows that the partitioning time increases almost linearly with the increase in data size. In this example, the number of records in each block is $n = 100\,000$. These results also show that generating an RSP from a big data set saves significant computational times of the RLS as we discuss in the following section.

IV. Sampling With RSP Blocks

After an HDFS file is converted into an HDFS-RSP file, sampling a set of random samples for approximate analysis becomes directly sampling a set of RSP block files. Since each RSP block file is a random sample of the entire data set, the expensive RLS process is no longer necessary. In this section, we define

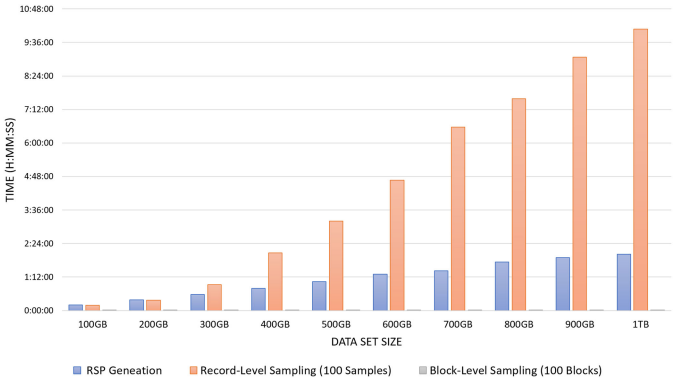


Fig. 7. Time performance for generating RSPs (left bar) from synthesized datasets with 100 features on a small Spark cluster of five nodes. The two-stage method was run on ten different sizes of data from 100 GB ($P = K = 1000$ blocks) to 1 TB ($P = K = 10\,000$ blocks). For comparison, the sampling time of 100 record-level samples (the middle bar) and the time for selecting 100 blocks (the right tiny bar) are shown for the same data sets.

BLS from an RSP and discuss its computational and statistical advantages in comparison with RLS and BLS from a normal HDFS file.

A. Block-Level Sampling From an RSP

In practice, an RSP model T from a data set $\mathbb{D}$ is created only once. Then, BLS is used to randomly select RSP blocks from T. A block-level sample of an RSP is defined as follows:

Definition 3: Block-Level Sample from an RSP: Let $\mathrm{S} = \{D_1, D_2, \ldots, D_g\}$ be a subset of T where $g < K$ and K is the number of RSP blocks in T. S is a block-level sample of T if $D_1, D_2, \ldots, D_g$ are randomly selected from T without replacement and with equal probability.

On a computing cluster, an RSP T is saved as an HDFS-RSP file with metadata $\mathrm{T}_{\mathrm{metadata}}$ storing RSP block information including the size and location. The BLS process uses $\mathrm{T}_{\mathrm{metadata}}$ to randomly select RSP blocks and is independent of the function f that is chosen to process the selected RSP blocks. The RSP blocks are selected without replacement, i.e., without repeating a block either in the same sample or in other samples for the same f. For each f, a selection table T_{f} of key-value pairs is created. The *key* is the block's identifier and the *value* is a binary scalar that represents whether the block was selected before for f (0: *NonSelected* and 1: *Selected*). This table is used to randomly select the identifiers of g RSP blocks from those blocks that were not selected before for f. Then, the locations of these blocks are obtained from $\mathrm{T}_{\mathrm{metadata}}$. This process can be applied to select RSP blocks depending on the availability of nodes in the cluster. Consequently, each selected RSP block can be computed locally on its node to avoid data movement.

Selecting the distributed RSP blocks directly as random samples saves a lot of sampling time, especially when many samples are required. For instance, it takes 10–15 s in average to select 100 RSP blocks from 100 GB data ($K = 1000$) and load them locally using Apache Spark or Microsoft R Server. As shown in Fig. 7, this time doesn't vary with bigger data sizes (the right tiny bar) because BLS depends only on the metadata. On the other hand, to get a record-level sample from an HDFS file, all the data should be loaded for selection of records with equal probability. In Fig. 7, we show the sampling time of 100 record-level samples without replacement (the middle bar). The number of records in each sample is also $n = 100\,000$. For small sizes, the sampling time of 100 record-level samples is almost the same as the time of RSP generation. However, the RLS time increases dramatically as the data becomes bigger. These results were produced with Apache Spark by repeating the *sample* transformation function, followed by the *count* action,[15] for 100 times on the RDD of each data set. With Microsoft R Server, this is even more expensive (e.g., drawing 100 record-level samples from 100 GB on the same cluster required more than 1 h). In case that record-level samples need to be written in HDFS for later analysis, the RLS time will be further increased dramatically. This shows the computational advantage of generating an RSP from $\mathbb{D}$ and using RSP blocks as random samples.

TABLE I
SAMPLE MEAN AND STANDARD DEVIATION OF V50 IN THREE CASES: 10 HDFS BLOCKS, 10 RSP BLOCKS AND 10 RECORD-LEVEL SAMPLES (RLS)

Sample Mean			Sample Standard Deviation		
HDFS	RSP	RLS	HDFS	RSP	RLS
0.965	4.223	4.217	0.000	6.251	6.276
0.978	4.248	4.227	2.993	6.251	6.276
2.005	4.248	4.228	3.000	6.252	6.289
4.020	4.253	4.228	3.985	6.257	6.292
4.987	4.265	4.242	4.995	6.261	6.295
5.000	4.271	4.245	4.997	6.261	6.300
5.001	4.272	4.249	6.016	6.262	6.306
6.980	4.273	4.254	6.017	6.270	6.309
7.006	4.276	4.266	6.982	6.273	6.319
7.984	4.276	4.277	8.018	6.279	6.323

B. Sample Statistics and Sampling Distributions

To demonstrate the quality of RSP blocks in comparison with the quality of HDFS blocks and record-level samples, we estimated the sample distribution of the sample mean and standard deviation from multiple samples or blocks of the data. Taking 100 GB synthetic data ($N = 100\,000\,000$, $K = 1000$, and $n = 100\,000$) as an example, the sample distribution was estimated in three cases: 100 randomly selected HDFS blocks, 100 randomly selected RSP blocks, and 100 record-level samples. The number of records n in each RSP block, HDFS block, and record-level sample is 100 000. We found that the estimates from RSP blocks and record-level samples are equivalent and close to the true values. In both cases, we got approximately the same variance, range, and standard error of the sample estimates. On the other hand, we found that the estimates from HDFS blocks have wider range, higher variance, and standard error. This shows that the samples of RSP blocks are equivalent to record-level samples, while the samples of HDFS blocks are not consistent because they are not, in general, random samples. Table I shows the values of the sample mean and standard deviation of feature V50 (Mean = 4.24, Std. Dev = 6.29,

[15] In Apache Spark, transformations on an RDD are only executed when an action is called.

TABLE II
SAMPLE MEAN AND STANDARD DEVIATION OF V50 USING THE BLB METHOD IN THREE CASES: 100 HDFS BLOCKS, 100 RSP BLOCKS, AND 100 RECORD-LEVEL SAMPLES (RLS)

Stat	Average Confidence Interval: 0.05%, 0.95%		
	HDFS	RSP	RLS
Mean	0.046, 8.998	4.041, 4.861	4.008, 4.896
StdDev	0.002, 8.937	5.701, 6.401	5.700, 6.427

Min $= -44.14$, Max $= 55.21$) in 10 HDFS blocks, 10 RSP blocks, and 10 record-level samples.

RSP blocks can also be used as subsamples to conduct the BLB analysis [16]. We applied this method to 100 HDFS blocks, 100 RSP blocks, and 100 record-level samples. We used 100 resamples from each block or sample to estimate the mean and standard deviation. The quality of the estimate was assessed with confidence intervals of [0.05%, 0.95%]. The results in Table II show that RSP blocks and record-level samples produce equivalent narrow intervals while HDFS blocks lead to a different wide interval. The RSP model results in good quality of estimates, and saves time, memory, and storage. Consequently, this model enables approximate big data analysis from a few RSP blocks as we discuss in the following section.

V. BIG DATA ANALYSIS AND MANAGEMENT WITH RSP BLOCKS

Given that a big data set $\mathbb{D}$ is stored as an HDFS-RSP file T on a computing cluster, the analysis of $\mathbb{D}$ will not be limited to the available memory. The RSP model enables a big data set to be analyzed on small computing clusters. In this section, we describe the RSP-based method for approximate big data analysis and show experimental results of real and synthetic data on a small computing cluster. After that, we discuss extensions to manage and analyze big data in two different scenarios using the RSP model.

A. Approximate Big Data Analysis With RSP Blocks

The RSP-based method uses block-level samples from an RSP of $\mathbb{D}$ to explore its statistical properties and build predictive models using existing sequential implementations of data analysis and mining algorithms. Only a few RSP blocks are selected and processed on their nodes without either loading the entire data set in memory or conducting expensive online RLS operations. In this method, we follow a stepwise ensemble process to obtain an approximate result from multiple batches of RSP blocks as shown in Fig. 8. To analyze $\mathbb{D}$ using a function f, this process works as follows:

1) First, a block-level sample of g RSP blocks is selected from an RSP, T, without replacement;
2) Second, the selected RSP blocks are processed in parallel as g independent computational tasks by applying a sequential implementation of f locally to each RSP block;
3) Third, the individual outputs of the RSP blocks are collected by the master node and combined to produce an approximate result for $\mathbb{D}$.

For exploratory data analysis where f is an estimate function, e.g., mean or variance, an ensemble estimate is calculated by averaging the base estimates computed from the selected RSP blocks. For predictive modeling where f is an algorithm to build a model, e.g., decision tree, the predictions from the base models are used to obtain the ensemble result for a new data point (i.e., the majority class in a classification task or the average response in a regression task). To improve the accuracy, these three steps are repeated until a satisfactory result is obtained or all RSP blocks are used up.

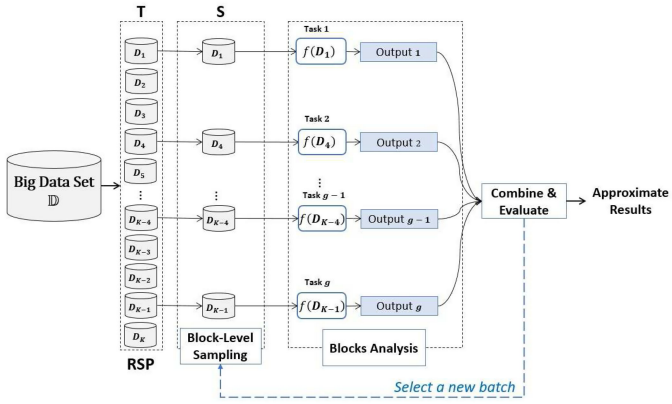


Fig. 8. The RSP-based method for approximate big data analysis.

TABLE III
REAL AND SYNTHETIC DATA SETS

Name - Task	N	M	K	n
Covertype - Classification (7 classes)	581 012	54	12	50 000
HIGGS - Classification (2 classes)	11 000 000	28	100	110 000
Taxi - Regression	125 000 000	26	250	500 000
DS1 - Regression	100 000 000	100	1000	100 000
DS2 - Classification (100 classes)	100 000 000	100	1000	100 000
DS3 - Classification (100 classes)	1000 000 000	100	10 000	100 000

An RSP of K RSP blocks with n records in each block was created from each data set.

B. Performance Evaluation

We developed a new distributed data-parallel framework based on the RSP method. We tested the prototype of this framework using a small cluster with Apache Spark 1.6 and Microsoft R Server 9.1. In this prototype, a data processing and analysis function f uses an existing R function that runs on g randomly selected RSP blocks in a parallel and distributed fashion. Each batch of g RSP blocks is executed as a single Spark job with only one stage of g Spark tasks. Three real data sets and three synthetic data sets were used in the test as listed in Table III. Covertype is used to predict the forest cover type from cartographic variables. HIGGS[16] is used to distinguish a signal process which produces Higgs bosons from the background process without Higgs bosons. Taxi, a preprocessed data set of New York Taxi[17] data in 2015, is used to predict the tip amount. DS1 is a synthetic data set for linear regression with 100 features.

[16][Online]. Available: https://archive.ics.uci.edu/ml/datasets/HIGGS
[17][Online]. Available: http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml

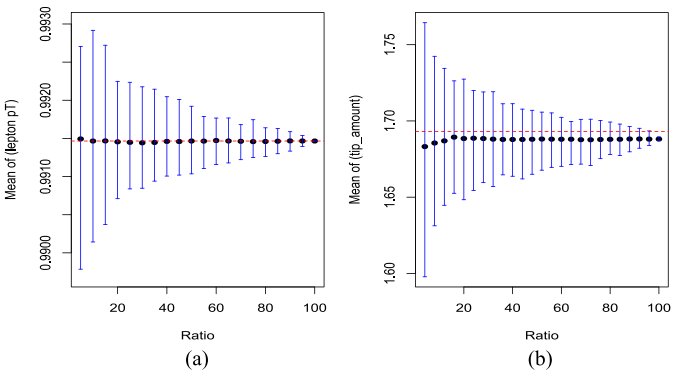


Fig. 9. RSP-based estimation of the mean of (a) the *lepton pT* feature in HIGGS data with $g = 5$ and (b) the *tip_amount* feature in Taxi data with $g = 10$. Each point represents the estimated value after each batch of g RSP blocks with the error range from 100 runs of the process. The dashed line represents the true value calculated from the entire data set.

DS2 and DS3 are two synthetic data sets for classification with 100 features and 100 classes.

First, an RSP was created from each data set using the two-stage data partitioning method with n records in each RSP block. n was chosen for each data set according to both the complexity of the data and the storage size of an HDFS block on our cluster. Then, the stepwise ensemble process was run to explore and analyze each data set using existing sequential R functions for data summary, classification, and regression. In this experiment, we used three R packages: *RevoScaleR*[18] containing parallel implementations of data analysis and mining algorithms, *rpart*[19] which is a sequential implementation of the CART decision tree algorithm, and *stats* package containing functions for fitting regression models.

The experimental results of these data sets are summarized as follows.

1) *Summary statistics:* Fig. 9(a) and (b) shows the results of applying the stepwise process, repeated for 100 times, to estimate the mean of *lepton pT* feature (Mean = 0.99, Std. Dev = 0.56, Min = 0.27, and Max = 12.09) in HIGGS, and the mean of *tip_amount* feature (Mean = 1.69, Std. Dev = 2.31, Min = 0, and Max = 40) in Taxi, respectively. This shows the change of the estimated values after each batch of RSP blocks until all RSP blocks are used up. We can observe that the error range becomes small and insignificant after few batches. We applied the same process to other data sets and found that a few RSP blocks are sufficient to obtain summary statistics that are close to the true values.
2) *Regression:* In Table IV, we compare single regression models (using the *rxLinMod* function in *RevoScaleR)* with RSP-based regression models (each base model was built using the *lm* function in *stats* package). The model performance of both single models and RSP-based ensemble models was measured in root mean square error (RMSE) and R-Squared. We can see that, with the RSP-based method, a small percentage of the data (12% of Taxi data and 2.4% of DS1) is enough to get equivalent regressors, and in the meantime, significant computation time is saved. For instance, instead of waiting about 6 min to get the regression model from the entire DS1 data, an equivalent regression model can be obtained in less than 1 min using only 24 RSP blocks.
3) *Classification*: Table V is a comparison between single classification models built from the entire data sets (using the *rxDTree* function in *RevoScaleR*) and RSP-based ensemble classifiers built from a few RSP blocks (each base model was built from one RSP block using the *rpart* function in *rpart* package). The performance of both single models and RSP-based ensemble models was measured in the overall accuracy and Kappa (Table V). Similar to the summary statistics and regression tasks, we can also see that a few RSP blocks are enough to produce classifiers which are equivalent to those built from the entire data, but in a significantly less time. For instance, while it takes more than 1 h to get a decision tree from the entire DS2 on the same cluster, an equivalent model can be obtained in less than 5 min using only 96 RSP blocks. Moreover, the memory of our cluster is not enough to hold the entire DS3 data (about 1 TB) and the number of blocks in DS3 (i.e., 10 000 blocks) is far beyond the available executors or cores. In such case, it is impractical or impossible to load and analyze the entire data set because the job either takes a very long time or fails due to the memory limit. The solution of this problem is straightforward with RSP blocks. We applied the RSP-based method and found that less than 192 RSP blocks (i.e., 3.84% of the data) were sufficient to obtain a model with 0.839 average accuracy.

C. Extensions to Cross-Data Centers and Evolving Data Sets

It is common to store industrial data in several data centers according to the geographical location of the source data. The same data may be stored in separate data centers. If the data is analyzed as a whole, it may need to integrate all parts of data from different data centers in one data center before the whole data can be analyzed. However, the RSP-based method can be directly used to analyze data in multiple data centers. First, the corresponding RSP of the data is created in each data center. If RSP blocks from different data centers have the same probability distributions, the analysis process can be applied locally in each data center to produce local results. Then, local results from different data centers are combined to produce the final results for the entire data. If RSP blocks from different data centers have different probability distributions, the samples of RSP blocks from all data centers are selected and downloaded to one cluster. The RSP blocks from different data centers are combined to form a set of RSP blocks for the whole data set. Then, the combined blocks are used to produce approximate results for the entire data stored in different centers.

For example, A is a big data set divided into three separate files (A_1, A_2, A_3) that are stored in three data centers (C_1, C_2, C_3),

[18][Online]. Available: https://docs.microsoft.com/en-us/machine-learning-server/r-reference/revoscaler/revoscaler

[19][Online]. Available: https://cran.r-project.org/web/packages/rpart/index.html

TABLE IV
COMPARING SINGLE REGRESSION MODELS AND RSP-BASED ENSEMBLE REGRESSION MODELS

Data	Single Model		RSP-based ensemble model				
	RMSE	R-Squared	RMSE		R-Squared		Ratio
			Mean±StdDev	Min, Max	Mean±StdDev	Min, Max	
Taxi	1.19	0.612	1.189± 0.0008	1.188, 1.191	0.614± 0.012	0.584, 0.635	12% (30 RSP blocks)
DS1	17.298	0.749	10.009± 0	10.009, 10.009	0.754± 0.005	0.744, 0.771	2.4% (24 RSP blocks)

TABLE V
COMPARING SINGLE CLASSIFICATION MODELS AND RSP-BASED ENSEMBLE CLASSIFICATION MODELS

Data	Single Model		RSP-based ensemble model				
	Accuracy	Kappa	Accuracy		Kappa		Ratio
			Mean±StdDev	Min, Max	Mean±StdDev	Min, Max	
Covertype	0.768	0.621	0.753± 0.0018	0.749, 0.759	0.592± 0.0029	0.586, 0.603	27.27% (3 RSP blocks)
HIGGS	0.705	0.408	0.713± 0.0012	0.710, 0.715	0.423± 0.0025	0.417, 0.429	10% (10 RSP blocks)
DS2	0.902	0.894	0.891± 0.0143	0.863, 0.919	0.890± 0.0144	0.861, 0.918	9.6% (96 RSP blocks)
DS3	-	-	0.839± 0.0357	0.785, 0.867	0.828± 0.0363	0.780, 0.851	3.84% (192 RSP blocks)

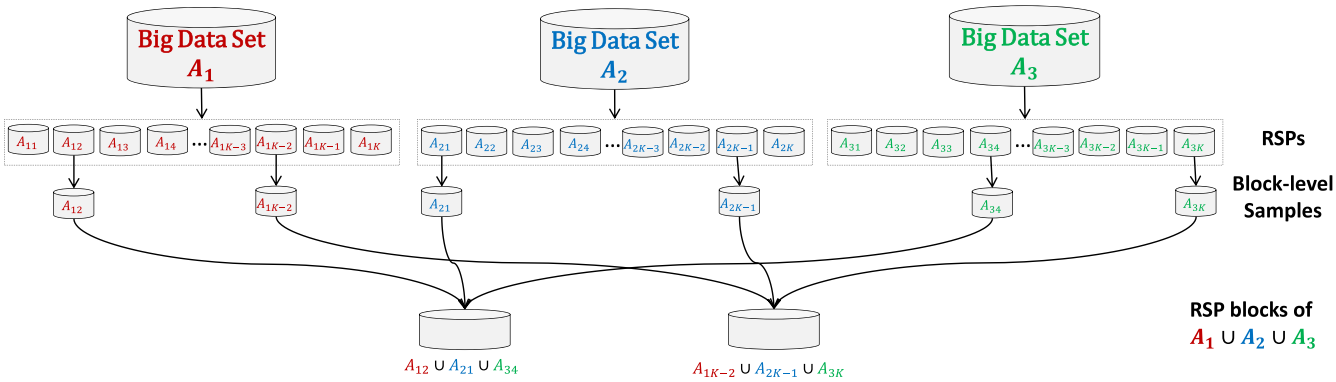


Fig. 10. Merging samples of RSP blocks from different data centers to produce RSP blocks of the entire data.

respectively. Fig. 10 illustrates the basic steps to produce representative RSP blocks of the entire data in A. An RSP is generated from each data subset A_i on its data center C_i for $1 \leq i \leq 3$. This operation is done only once in each data center. To analyze A, the same number of RSP blocks is randomly selected from each data center. The selected blocks can be downloaded into a local computing cluster. Then, the three sets of RSP blocks are merged into one set of RSP blocks for A where each new RSP block is formed by merging three RSP blocks, one from each data center. For instance, $g = 2$ RSP blocks are selected from each data center. The new blocks, i.e., $A_{12} \cup A_{21} \cup A_{34}$ and $A_{1K-2} \cup A_{2K-1} \cup A_{3K}$ in the figure, are RSP blocks of A (see Theorem 1 in [30]), and can be used to estimate the statistical properties of A or build ensemble models for A. In this case, the stepwise ensemble process can run on the local computing cluster which merges samples of RSP blocks from other data centers, stores the merged blocks on its local nodes, and continues the analysis process as before.

If a big data set $\mathbb{D}$ is an evolving data set, an RSP is created from the collected data in a time window. If RSP blocks in subsequent time windows have similar probability distributions, these blocks together represent an RSP of the entire data in the considered windows. If data have different probability distributions, RSP blocks from these windows are combined to represent an RSP of the entire data in these windows. In fact, to analyze data in different windows, we may not need to combine all RSP blocks from all windows. In a way similar to the cross-data center case, BLS is used to select RSP blocks from each window and merge them to produce RSP blocks of the entire data in all considered windows.

VI. CONCLUSION

In this paper, we proposed the RSP distributed data model to enable the distributed data blocks of a big data set to be used directly as random samples for approximate big data analysis. We showed that block-level samples from an RSP can be used efficiently and effectively for different data analysis tasks. We also demonstrated how a few RSP blocks are sufficient to obtain approximate estimates and models from big data using existing sequential implementations of data analysis and mining algorithms. The RSP data model enabled an efficient method to scale computing clusters to analyze and manage bigger data sets in different scenarios. Some of our future works will include extending the RSP-based method to other tasks (e.g., data cleaning and clustering) and testing it in different scenarios (e.g., multiple data centers and evolving data sets).

ACKNOWLEDGMENT

The authors sincerely thank the editors and anonymous reviewers whose valuable comments helped improve this paper significantly.

REFERENCES

[1] N. Lazar, "The big picture: Divide and combine to conquer big data," *CHANCE*, vol. 31, no. 1, pp. 57–59, 2018.

[2] J. Y. L. Lee, J. J. Brown, and L. M. Ryan, "Sufficiency revisited: Rethinking statistical algorithms in the big data era," *Amer. Statistician*, vol. 71, no. 3, pp. 202–208, 2017.

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

[4] S. Salloum, R. Dautov, X. Chen, P. X. Peng, and J. Z. Huang, "Big data analytics on apache spark," *Int. J. Data Sci. Analytics*, vol. 1, no. 3, pp. 145–164, 2016.

[5] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[6] R. Nair, "Big data needs approximate computing: Technical perspective," *Commun. ACM*, vol. 58, no. 1, pp. 104–104, Dec. 2014.

[7] S. Mittal, "A survey of techniques for approximate computing," *ACM Comput. Surv.*, vol. 48, no. 4, pp. 62:1–62:33, Mar. 2016.

[8] Í. Goiri, R. Bianchini, S. Nagarakatte, and T. D. Nguyen, "Approxhadoop: Bringing approximations to MapReduce frameworks," in *Proc. 20th Int. Conf. Architectural Support Program. Languages Operating Syst*. ACM, 2015, pp. 383–397.

[9] D. R. Krishnan, D. L. Quoc, P. Bhatotia, C. Fetzer, and R. Rodrigues, "Incapprox: A data analytics system for incremental approximate computing," in *Proc. 25th Int. Conf. World Wide Web*, ser. WWW '16. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2016, pp. 1133–1144.

[10] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica, "BlinkDB: Queries with bounded errors and bounded response times on very large data," in *Proc. 8th ACM Eur. Conf. Comput. Syst*. ACM, 2013, pp. 29–42.

[11] S. Kandula *et al.*, "Quickr: Lazily approximating complex adhoc queries in bigdata clusters," in *Proc. Int. Conf. Manage. Data*, ser. SIGMOD '16. New York, NY, USA: ACM, 2016, pp. 631–646.

[12] S. Salloum, J. Z. Huang, and Y. He, "Empirical analysis of asymptotic ensemble learning for big data," in *Proc. 3rd IEEE/ACM Int. Conf. Big Data Comput. Appl. Technologies*, ser. BDCAT '16. New York, NY, USA: ACM, 2016, pp. 8–17.

[13] J. Wang, S. Krishnan, M. J. Franklin, K. Goldberg, T. Kraska, and T. Milo, "A sample-and-clean framework for fast and accurate query processing on dirty data," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, ser. SIGMOD '14. New York, NY, USA: ACM, 2014, pp. 469–480.

[14] B. C. Kwon, J. Verma, P. J. Haas, and Ç. Demiralp, "Sampling for scalable visual analytics," *IEEE Comput. Graph. Appl.*, vol. 37, no. 1, pp. 100–108, Jan./Feb. 2017.

[15] J. J. Faraway and N. H. Augustin, "When small data beats big data," *Statist. Probability Lett.*, vol. 136, pp. 142–145, 2018.

[16] A. Kleiner, A. Talwalkar, P. Sarkar, and M. I. Jordan, "A scalable bootstrap for massive data," *J. Roy. Statistical Soc.: Series B (Statistical Methodology)*, vol. 76, no. 4, pp. 795–816.

[17] S. Agarwal *et al.*, "Knowing when you're wrong: Building fast and reliable approximate query processing systems," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, ser. SIGMOD'14. New York, NY, USA: ACM, 2014, pp. 481–492.

[18] L. Rokach, "Ensemble-based classifiers," *Artif. Intell. Rev.*, vol. 33, no. 1, pp. 1–39, 2010.

[19] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Proc. IEEE 26th Symp. Mass Storage Syst. Technologies (MSST)*, May 2010, pp. 1–10.

[20] X. Ci and X. Meng, *An Efficient Block Sampling Strategy for Online Aggregation in the Cloud*. Cham, Switzerland: Springer International Publishing, 2015.

[21] S. Chaudhuri, G. Das, and U. Srivastava, "Effective use of block-level sampling in statistics estimation," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, ser. SIGMOD '04. New York, NY, USA: ACM, 2004, pp. 287–298.

[22] V. Kalavri, V. Brundza, and V. Vlassov, "Block sampling: Efficient accurate online aggregation in MapReduce," in *Proc. IEEE 5th Int. Conf. Cloud Comput. Technol. Sci.*, vol. 1, Dec. 2013, pp. 250–257.

[23] Y. Wang, Y. Zhong, Q. Ma, and G. Yang, "Distributed and parallel construction method for equi-width histogram in cloud database," *Multiagent Grid Syst.*, vol. 13, no. 3, pp. 311–329, Sep. 2017.

[24] M. Vojnovic, F. Xu, and J. Zhou, "Sampling based range partition methods for big data analytics," Tech. Rep. MSR-TR-2012-18, February 2012.

[25] S. Phansalkar and S. Ahirrao, "Survey of data partitioning algorithms for big data stores," in *Proc. 4th Int. Conf. Parallel Distrib. Grid Comput. (PDGC)*, Dec. 2016, pp. 163–168.

[26] Q. Ke, V. Prabhakaran, Y. Xie, Y. Yu, J. Wu, and J. Yang, "Optimizing data partitioning for data-parallel computing," in *Proc. 13th USENIX Conf. Hot Topics Operating Syst.*, ser. HotOS'13. Berkeley, CA, USA: USENIX Association, 2011, p. 13.

[27] S. Landset, T. M. Khoshgoftaar, A. N. Richter, and T. Hasanin, "A survey of open source tools for machine learning with big data in the hadoop ecosystem," *J. Big Data*, vol. 2, no. 1, p. 24, Nov. 2015.

[28] M. Zaharia, M. Chowdhury, T. Das, and A. Dave, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," *NSDI'12 Proc. 9th USENIX Conf. Netw. Syst. Des. Implementation*, Apr. 2012, p. 2.

[29] S. Guha *et al.*, "Large complex data: divide and recombine (d&r) with RHIPE," *Stat*, vol. 1, no. 1, pp. 53–67, 2012.

[30] C. Wei, S. Salloum, T. Z. 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 – CLOUD 2018. Lecture Notes in Computer Science*, M. Luo and L.-J. Zhang, Eds., vol 10967, Cham: Springer International Publishing, 2018, pp. 347–364.

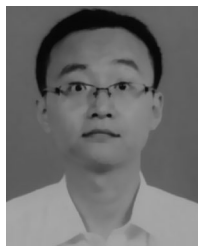
Salman Salloum was born in 1984. He received the master's degree in software engineering and information systems from Damascus University, Damascus, Syria, in 2013. He is currently working toward the Ph.D. degree in information and communication engineering with Big Data Institute, Shenzhen University, Shenzhen, China.

His current research is focused on distributed data-parallel computing and approximate computing for big data analysis.

Joshua Zhexue Huang was born in 1959. He received the Ph.D. degree in spatial databases from The Royal Institute of Technology, Stockholm, Sweden, in 1993.

He is currently a Distinguished Professor of Big Data with the College of Computer Science & Software Engineering, Shenzhen University, Shenzhen, China. He is also the Director of Big Data Institute, Shenzhen, China, and the Deputy Director of National Engineering Laboratory for Big Data System Computing Technology, Shenzhen, China. He has published over 200 research papers in conferences and journals. His main research interests include big data technology and applications.

Prof. Huang received the first Pacific Asia Conference on Knowledge Discovery and Data Mining (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 China.



Yulin He was born in 1982. He received the Ph.D. degree from Hebei University, Hebei, China, in 2014.

From 2013 to 2014, he has served as a Research Assistant with the Department of Computing, Hong Kong Polytechnic University, Hong Kong. From 2014 to 2017, he worked as a Postdoctoral Fellow in the College of Computer Science & Software Engineering, Shenzhen University, Shenzhen, Guangdong, China. He is currently an Assistant Professor of Big Data Technology and Application with Big Data Institute of Shenzhen University. His main research interests include neural network, imbalanced learning, probability density estimation for big data, and model generalization improvement based on artificial big data generation.