SURVEY ON DISTRIBUTED DATA STRATEGIES TO SUPPORT LARGE-SCALE DATA ANALYSIS ACROSS GEO-DISTRIBUTED DATA CENTERS

Niranjan Kokate*1, Shailesh Bendale*2

*1,2Department Of Computer Engineering, NBN Technical Campus, Pune, India.

ABSTRACT

The storage of big data in a single data center is no longer possible when the volume of data rises fast. As a result, businesses have devised two scenarios for storing their large data across several data centers. The company's large data is scattered across numerous data centers in the first scenario, with no data replication. Data is kept in numerous data centers in the second scenario, but vital data is replicated in these data centers to ensure data safety and availability. Analyzing massive data dispersed across several data centers. In this we study two data distribution algorithms from our referenced base paper [1] to assist big data analysis across geo-distributed data.

Keywords: Distributed Computing, Random Sample Partition, Wide Area Analysis, Big Data.

I. INTRODUCTION

As today's world shifts towards data driven or data centric lifestyle, day to day basis applications generating more vast and complex data. So it's like solar system of our planet, as sun is in center and all planets rotate around it. So, all industries are depending upon data. So, it's must for industries to carry out complex operations to gain insights for data. As big companies that serves overseas countries or vast geographical areas need robust and scalable system to carry out rigorous operations with maintaining seamless experience for clients. So industries use cloud data centers span across many countries. So, for analysis of this big data in multiple data centers its need to do it on whole data for delivering global reports to decision makers and recommending features for targeted clients. While taking this scenario into consideration of doing analytical job over span data across multiple data centers as whole, without hampering efficiency and effectiveness of analytical job is big concern for current big data technologies. The most prominent framework for big data processing and analysis in parallel and distributed fashion are Apache Hadoop and Spark. These frameworks, on the other hand, designed to process and analyze data locally within same DC, necessitating the cloning of all data to a single DC prior to performing a locally distributed computation. Many distributed execution solutions have been proposed with the aim of interpreting big data across multiple network infrastructure. The strategy of distributing analytic jobs or tasks among geo-distributed DC’s and then quantifying the intermediate results for further processing is acknowledged as distributed execution. To facilitate distributed large data analysis, the random sample partition (RSP) data model was recently recommended. A massive data file is represented in the RSP data model as a succession of non-overlapping data blocks called RSP data blocks, including one that comprises a random sampling of the whole big data. To synthesize RSP data blocks from other HDFS data files, a two-stage data processing approach was developed. To administer RSP data blocks on a computing cluster, a large data management system (BDMS) was described in. RSP data block compilation and RSP block-level sampling were implemented in BDMS through using RRPlib Spark package.

In this review paper we focused on issue of interpreting massive data across various data centers are the goal of this study. In practices, enterprises have devised two scenarios for storing massive data in lots of data centers: without replication or with replication. In our reviewed paper two data distribution strategies to assist big data analysis across geo-distributed DCs, based on above two scenarios and the RSP data model.

In first approach, Big Data Management System (BDMS was developed in [2] to administer RSP data blocks on various computing cluster) is installed in each DC. If big data set is span across multiple DCs each containing subset of big data, so RSP data blocks of each subset of big data is stored on local BDMS as RSP data model. In analyzing phase of this big data, subset of RSP data blocks distributed across multiple DCs are acquired and merged together in a single data center forming new set of RSP data blocks which is depiction of random sample of whole big data.
Fig. 1. To store data on several data centers, use Strategy 1: choose a set of RSP data blocks at arbitrary from every data center, then download the specified RSP data blocks to the central data center for analysis.

In second approach it updates the distribution of RSP data model. In short, same strategy of first approach is applied but subset of RSP data model stored on each DCs is replicated different data centers to avail benefits of data availability and safety. As the RSP data blocks are replicated in each other data centers so an analytical job is scheduled on a single data center as it contains subset of RSP data blocks which itself is subset of whole big data. Since collection of samples and analysis of RSP data blocks carried out within single data center, no data communication requirement is essential for analysis purpose.

Fig. 2. Each data center holds some RSP data blocks of each of the original five data files D1, D2, D3, D4, D5, as shown in figure an example of replication scheme among five data centers.

II. BACKGROUND

In recent years, research into analyzing data across geographically dispersed data centers has grown in prominence. The current initiatives’ main goal is to deploy data analytic jobs efficiently across numerous data centers. The emergence of social media has made sharing and accessing information faster and smoother. Social media is being used to promote consciousness, especially in critical situations. When the Amazon rainforests saw severe clusters of smoldering flames, the first tweet about the fires was posted on August 6, 2019, two weeks before the disaster was reported on cable news. The reason for giving this example is that in these critical moments red alerts and alarming for disaster management teams will overcome the crisis. As the volume and velocity of big data vary among huge amounts so gaining insights, analyzing big data and reporting such severe condition is highly valuable. So main aim or problem is to analyze effectively and efficiently this huge amount of data. Many enterprise solutions have been proposed. For example [3]-[7] authors in this developed various task scheduler for overcoming volume of data traffic. Volley [8] and Iridium [9] focused on data placement issues through this it able to reduce job completion time by wisely distributing data based on
users or client’s requests across various geographically distributed DCs. The authors in [10] enhanced the shuffle stages by proactively transmitting files from mapper to reducer tasks during the shuffle phase to maintain higher locality. Despite their significant implications, due to the level of abstraction essential to address the problem, these tactics may not contribute to better performance. Existing works, for example, presumed that intermediate data dimensions were known ahead of schedule, regardless of the fact that this is highly unlikely to happen in exercise. Moreover, different task assignment algorithms lead to conflicting flow patterns throughout data centers, which, in fact, results in different job completion timelines. Secondly, across multiple data centers, the statistical distribution of data, as well as the quantities of data, varies over time. The available capacities of data centers also vary. Non-analytical jobs, for example, can overburden one or more data centers, resulting in resource scarcity, delayed system response, and poor performance. Finally, network bandwidth fluctuation is a major factor because data centers with low bandwidth would become a bottleneck if tasks are evenly distributed among all data centers. Many studies have concentrated on data replication in cloud systems. Boru et al. [11] proposed a data replication mechanism for cloud data centers. In both intra- and inter-data centers, the proposed framework optimizes energy consumption due to network bandwidth and communication latency. Using the notion of the knapsack problem, Gill and Singh [12] devised a method to minimize the replication cost. When the expense of replication surpasses the user budget, this technique retransfers the replicas to a lower-cost data center. Matri et al. [13] proposed a write-enabled dynamic replication technique that takes advantage of decentralized storage systems like Dynamo [14] and Voldemort [15]. They also proposed an approximated object locating mechanism, which would allow clients to tentatively find the closest data replica.

Mansouri et al. [16] offered a way to decrease the offline cost of data migration and dynamic replication across cloud data centers. Limam et al. [17] developed a dynamic replication technique to improve the availability and satisfaction of tenant data while accounting for tenant budget and provider profit. The researchers used a cost model to figure out how many replicas are needed to ensure excellent data availability. According to the cost model, the replica can only be made before the minimum number of replicas is reached or when the required reaction time is not exceeded. Furthermore, the authors concentrated exclusively on read-only data replication. In summary, the preceding report concentrated on approaches for simplifying the production of data copies in order to cut costs and time. In the paper reviewed by us introduce a different data replication approach to facilitate large data analysis in this work. The new technique attempts to present a novel solution on data replication, network bandwidth, and robustly doing the analytical jobs.

III. LITERATURE REVIEW

A. Random Sample Partition

The Random Sample Partition (RSP) distributed data model is presented to ease block-level sampling and support huge data analysis. The vision is to help it possible to use the scattered data blocks of a huge data collection as random samples in approximation big data analysis. In this architecture, a large data set is represented as a series of tiny discrete random sample data blocks known as RSP blocks. The probability density function of each RSP block is identical to that of the entire data set. As a result, an RSP block is just like a record-level sample from all the other data. To construct an RSP from an HDFS file, a two-stage data partitioning algorithm is devised. In exercise, an RSP is generated on a computer cluster offline and only once. Block-level samples became just as effective as record-level samples using the RSP model, yet sampling RSP blocks is efficient. The experimental findings reveal that sample statistics and distributions from RSP blocks are equal to those from record-level samples, but much better than those from standard HDFS blocks. The amount of time required to obtain several random samples from massive amounts of data is halved.

The RSP paradigm has substantial consequences on the efficiency of big data processing due to the statistical benefits of RSP blocks. It enables the RSP-based approach for approximate big data analysis, a different feature for analyzing big data on tiny computer clusters. Leveraging block-level samples from an RSP, this technique employs a step-by-step approach to create approximate findings. The RSP is first used to choose a block-level sample. The number of chosen blocks is governed on the resources that are available. Second, each selected RSP block is subjected to a sequence of events that executes in simultaneously. Third, the output of these blocks are
pooled to provide a rough figure of the whole data. The three processes can be repeated to progressively optimize the outcome.

On Hadoop cluster platforms, big data files are frequently stored as HDFS files. To create an RSP from an HDFS file, as the partition operation on massive data for computer clusters, a two-stage data partitioning mechanism is devised. The two stages of the data partitioning method are as follows [18]:

- **Stage 1:** Divide D into $P$ non-overlapping subsets $\{D_1, D_2, \ldots, D_P\}$ of equal size (e.g., $P$ is the number of HDFS blocks in $D$). 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 $\{D_{1,1}', D_{1,2}', \ldots, D_{1,K}'; D_{2,1}', D_{2,2}', \ldots, D_{2,K}'\}, \ldots, \{D_{P,1}', D_{P,2}', \ldots, D_{P,K}'\}$.

- **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 $D$ (see Theorem 2 in [18].)

The created RSP is saved as an HDFS-RSP file.

**Fig. 3.** Time performance for generating RSPs (the left bar) from synthesized datasets with 100 features on a small Spark cluster of 5 nodes. The two-stage method was run on 10 different sizes of data from 100GB ($P = K = 1000$ blocks) to 1TB ($P = K = 10,000$ 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.

**B. Big Data Management System**

The random sample partition (RSP) data model [19] was recently introduced as the foundation for the large data management system (BDMS) [2]. The goal of this system’s design was to store massive amounts of data as distributed RSP data blocks in the RSP data model, and perhaps to facilitate block-level sampling of RSP data blocks for approximation big data analysis on computer clusters. RSP data block production using random samples data splitting, data blocks organization, and data blocks sampling are only a few of the features available in this system for managing RSP data blocks. Block-based sampling for different data analysis operations carried out on computer clusters may be effectively and efficiently supported by operating and maintaining huge data sets as RSP data blocks. This system’s enormous effect may be expanded to simulate large data analysis throughout several data centers.

**Fig. 4.** High Level Architecture Of BDMS

The block manager, block registry, and BDMS client are the three fundamental units of the BDMS architecture, as depicted in Fig. 4. On the dedicated links of a cluster, the HDFS system can be utilized as the baseline distributed file capacity for collecting RSP data blocks as data block files. The block manager is in charge of...
transforming large volumes of data into RSP data chunks and organizing the file blocks' logical structure. The block registry is a data repository that comprises a statistical overview as well as all metadata on RSP data blocks. The BDMS client is a library that allows you to connect the BDMS to other software.

**C. Pixida**

PIXIDA is a data analytics framework scheduler that is intended to minimize the volume of traffic that must travel links that are more liable to congestion, particularly ones between DCs. Jobs submitted to PIXIDA have input from several DCs and require that their output be kept at a certain DC, which may or may not be the same as the ones holding the input.

![Figure 5](image)

**Fig. 5.** For a three-partition input, the scenario for Spark with its associated operator- and task-level job graphs ((a) and (b) correspondingly).

PIXIDA does this by converting the traffic minimization issue into a graph partitioning question, in which the job's task-level graph (e.g., Figure 5b) is subdivided into partitions. The jobs that are launched in the same DC are divided into these divisions. To create such a graph, PIXIDA requires the following information: a) the job's task-level graph, b) the places of the input partitions and output, and c) the quantity of each task's output.

The graph is known at compile time, i.e., the coordinates of the input partitions can be determined from the underpinning storage system, such as HDFS, and the output destination is specified by the user once the task is submitted for processing. Thus, rely on statistics acquired by a Tracer phase for the third piece of data. It would use the Tracer to estimate the magnitude of each task's output.

**D. RRPLib**

RRPLib primarily consists of three parts: data generator, RRP, and massive-RRP. The data generator component’s job is to create artificial datasets for classification and regression. The RRP component, as well as the massive-RRP component, is an implementation example for the round-random partitioner in [20]. The simulated data is used to compare the results of various machine learning methods when applied to large datasets. The data generator is primarily meant to produce synthetic datasets for different algorithms for this purpose. It specifically look at the two most common learning problems, classification and regression.

RRP [20] proposes partitioning a large file into non-overlapping blocks, with each block representing a random sample of the whole file. The massive-RRP component is given as an alternate method if the data surpasses the restricted resources, making the RRP algorithm difficult to implement. The primary concept is to partition the data into several groups of blocks, allowing the RRP method to be applied within a constrained resource range. It presume that the blocks are saved in distinct files in the massive-RRP implementation, and that the huge file is the folder that includes these blocks' files.
IV. DISCUSSION

The base paper which reviewed, explored a foundation of a new big data analytics conundrum. It focuses on comparative big data analysis across geographically distributed data centers in this study. In two data distribution situations, with or without replication, it suggests two data analysis methodologies. If a decision-maker requires a quick look at the data, Strategy 1 is available; downloading a few RSP blocks from each data center is enough to gain a rough estimate. Furthermore, consider that the data will require various operations to be done on it. It is more economical to have selected RSP blocks downloaded first to a local data center and then analyzed, than repeating these processes on the entire data by dispersing jobs and tasks to multiple data centers. Furthermore, this method isolates data storage from data analysis.

On the other hand, managing data across different data centers using Strategy 2 leads to many advantages, including: (a) The high availability and low latency. For customers who are geographically distant, replicating data across various geo-distributed data centers raises response times. (b) A disaster recovery plan. (c) 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 process sets up every data center's data redundancies to be a random selection of the entire data. The large data's distributed random samples allow for rapid approximation analysis and distributed ensemble learning or estimation. It is more efficient than existing techniques of data transfer since the local models of faraway data centers are sent to the local data center. Finally, it accepts that the time and bandwidth utilization of data replication across geo-distributed data centers is a constraint of our technique. However, in each data center, it only duplicates a sample of RSP data blocks, and the data replication is done just once and may be automatically generated when the network is not overburdened.

V. CONCLUSION

We study two ways in this review paper to facilitate approximation large analysis of dispersed data across numerous data centers. On each data center, it stores the data as a collection of RSP data blocks in both techniques. In the first technique, some data blocks must be downloaded from remote data centers to a central data center in order to do a rough analysis of the large data as a whole. The key benefit of this method is that it separates the storage and analysis levels. We study data replication throughout different data centers in the second alternative. The data at every data center forms a random sample of the entire dispersed data in the event of data replication.

VI. REFERENCES


