CLOUD BASED DATA ANALYSIS INTEGRATING R FOR SCALABLE DISTRIBUTED SYSTEM SUPPORTING REAL-TIME ANALYTICS

Rishi Jain[1], S.Venkatesan[2]

[1]*B.E final year - CSE Dept – Dayananda Sagar College of Engineering, Bangalore – 560078*

[2]*Professor, CSE Dept. – Dayananda Sagar College of Engineering-560078*

ABSTRACT

The need to perform complicated statistic analysis of big data by institutions of engineering, scientific research, health care, commerce, banking and computerresearch is immense. It is widely recognized that OLTP and OLAP queries have different data access patterns, processing needs and requirements. Therefore, the OLTP and OLAP queries are specifically handled by two different systems, and the data are periodically extracted from the OLTP system, transformed and loaded into the OLAP system for data analysis.

Learning “Data Analysis with R” not only adds to existing analytics knowledge and methodology, but also equips with exposure into latest analytics techniques including forecasting, social media analytics, text mining & so on.Enterprise with a cloud environment can encompass cost of hardware, upgrading software, maintenance or network configuration, thus it making it more economical.

Keywords: R, Data Analysis,Apache Hadoop,Cloud, OLAP, OLTP.

I. INTRODUCTION

The Cloud Based Big Data Analytics is a system to deal with big data to perform linear regression and similar predictive analysis with ease and prove to be very helpful for engineering research, business, health care, scientific research, banking & finance and machine learning where complicated statistical analysis need to be performed. Analysis of large data is very complicated for traditional analytic environment which can be done with ease in distributed environment without undermining the quality of the result.The commercial Bigdata Analytical tools like IBM BigInsight, Teradata & so on are available in the market. These tools are both commercially high in license cost and require additional investment for building & maintaining computing cluster. In Big enterprises volume of data to be analyzed keeps growing exponentially thereby escalating the demand for additional nodes in the cluster, space and overall cost. Such cluster setups demand heavy capital expenditures making it infeasible for SME (Small & Medium Enterprises) as well as is poses an overhead of maintenance, space, disaster recovery for bigger enterprises.

Database systems implemented for large scale data processing are typically classified into two categories: OLTP systems and OLAP systems. The data stored in OLTP systems are continuously exported to OLAP systems through Extract- Transform-Load (ETL) tools. In recent years, MapReduce framework has been widely used in implementing large scale OLAP systems because of its scalability, and these include Hive. Most of the present only focus on optimizing OLAP, and are oblivious to updates made to the OLTP data since the last loading. However, with the increasing need to support real-time analytics, the issue of freshness of the OLAP results has to be addressed, for the simple fact that more up-to-date analytical results would be more utilizing for time-critical decision making. The idea of supporting real-time OLAP (RTOLAP) has been investigated in traditional database systems. The most straightforward approach is to perform near real-time ETL by shortening the refresh interval of data stored in OLAP systems.

The RTOLAP is defined : a real-time OLAP (RTOLAP) query accesses, for each key, the recent value preceding the submission time of the query. Specifically, we propose and design a scalable distributed RTOLAP system called R-Store, in which the storage system supports multi-versioning, and each version is associated with a timestamp.To facilitate enhanced processing of RTOLAP queries, we periodically materialize the real-time data into a data cube and implement an IncrementalScan operation in HBase to avoid the shuffling of the entire HBase table to MapReduce during real-time querying. To the best of our knowledge, this is the first work that proposes a scalable RTOLAP distributed system based on MapReduce framework. In summary, the contributions of this paper are as follows:

1) A scalable distributed system frameworkcalled R-Store, for performing RTOLAP. R-Storeevaluates an OLAP query by transforming it into aMapReduce job, which is run on our modified HBase(in the remaining of this paper, we name it as HBase-R in order to differentiate it from HBase), to obtainthe real-time data.
2) An enhanced storage model for caching thedata cube result. The data cube is treated as historicaldata, while the data updated after the refresh time ofthe data cube are real-time data. We also propose amore enhanced scan operation in the storage model forobtaining the real-time data.
3) Integrating streaming MapReduce into the system,which maintains a real-time data cube in the reducers,and periodically materializes the data cube. This datacube update method is much swifter than the data cubere-computation method, and in turn theprocessing of RTOLAP since fewer real-time data arescanned during the query execution.
4) Design an algorithm to efficiently process theRTOLAP queries, which takes both the historical datacube and the real-time table as input. We also proposea cost module that directs the adaptive processing ofRTOLAP.
5) Perform an extensive experimental study on acluster with more than one hundred nodes, whichconfirms the effectiveness of the cost model, and theefficiency and scalability of R-Store.

II. RELATED WORK

Big data is a terminology that depicts the large chunk of data – both structured and unstructured – that inundates a business on a day-to-day basis. But it's not the amount of data that's vital. It's what organizations perform with the data that matters. Big data can be analyzed for better decisions and strategic business moves.

The proposal touches on a number of areas such as OLAP processing, distributed processing and data cube maintenance.

A. Real-Time Data Warehousing

The growing demand for fast business analysis coupledwith increasing use of stream data have generated great interestin real-time data warehousing. Some have proposed near real-time ETL, as a means to shorten the data warehouserefresh intervals. These works require fewer modifications tothe existing systems, but they cannot achieve 100% real-time. In C-store two separate storesare used to handle in-place updates. The updates are generally storedin a write-store (WS), while queries execute against the readstore(RS), and consolidated with the WS during execution.

B. Distributed Processing

MapReduce is a parallel data processing framework for large scale data processing. Its programming model consists of two user-defined functions, map and reduce, that operate on key/value pairs.There arefew researches on supporting both OLTP and OLAP in a single hybrid system.

While MapReduce provides an efficient and simple platform for scalable distributed processing, it is not efficient for supporting online and continuous stream processing. HStreaming and MapReduce Online are extensions made to the MapReduce framework that support stream processing.

C. Data Cube Maintenance

Data cube maintenance has been studied for a long time. The earliest works focused on efficient incremental view maintenance for data warehouses. However, as the count of dimension attributes increases, the cost of incrementally updatation data cube increases drastically. To improve the performance of data cube maintenance, instead of generating the delta value for all the cuboids during the update process, an method of refreshing multiple cuboids by the delta value of asingle cuboid has been proposed. Most of these algorithms were designed for a single node configuration and are not scalable to a distributed environment.

III. SYSTEM MODEL

Cloud based analysis constitutes four vital components

- R statistical software to perform statistical analysis

- Hadoop framework to distribute data and compute tasks in the cluster computing
- Rhadoop to link R and Hadoop
- Amazon EMR to deploy system to provide SaaS.

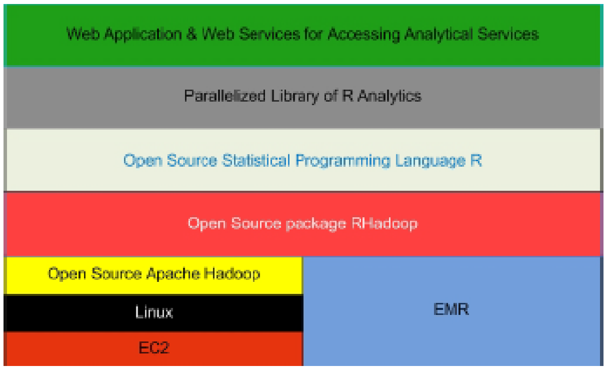

Figure 1. System Model of Cloud Based Analysis

A. Web Application & Web Services for Accessing Analytical Services:

CBA is designed in such a way that anyone with basic computer knowledge can utelise it convenintly. The features of web application are:

- Authenticity check
- Convenient
- Data upload and file management
- Select analytical feature to perform analysis
- View result etc.

B. Parallelized library of R Analytics:

Open source Apache Hadoop and Open source high level statistical programming language R are collaborated to create a parallelized library of R analytics.

C. Open Source Statistical Programming Language R:

R is a free software programming language. It is used by statisticians and data miners for statistical computing, graphics and several such applications.R is rich in various statistical analysis packages. There are multiple packages available in the number of 5922 in CRAN packages depository and the number is ever incremental.

D. Open Source package RHadoop:

RHadoop is an open source projectaimed at large scale data analysts to empower them to usethe horizontal scalability oh Hadoop using the R language.ravro, plymr, rmr, rhdfs and rhbase, the 5 R packages enableits users to manage and analyze massive quantities ofinformation using Hadoop.

- ravro – which is the R package thatallows the readingand writing of files in avro format, to R
- Plyrmr – is a more recent R package that makes R andHadoop perform in near perfect if not perfectharmony, in the analysis of higher lever plyr like data.
- Rmr - is a R package that came into being to allowusers to write map reduce programs in R since it ismore productive and far more easier
- Rhdfs - is a R package that gives administration ofHDFS files from within R. It uses Hadoop common togive access to map reduce file services
- Rhbase - is a R package that permits its users toget connect with hbase and corelate with hbase functions.

E. The Open Source Apache Hadoop:

Apache Hadoop is asimple to use and implement, dependable and accurate distributed computing framework. It prompts it's users to shift from single server to multiple connected machines each functioning as a separate unit for data storage and computing. Since Apache Hadoop software library that has several thousand computers over a cluster, accuracy of analysis can be assured, as each unit is so programmed to spot and rectify errors.Hadoop stores massive quantities of data over several systems in the cluster.

IV. ARCHITECTURE AND DESIGN

A. R-Store Architecture:

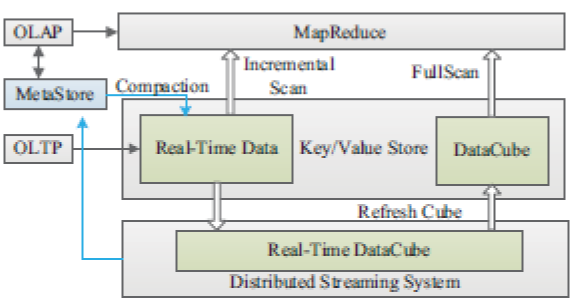

Fig. 2. Arcchitecture of R-Store

The system holds four components: a distributed key/value store, a streaming system for upholding the real-time data cube, a MapReduce system for efficient processing large scale OLAP queries, and a MetaStore for storing some global variables and configurations.

The OLTP processing queries are submitted to the key/value store, while the OLAP queries are simply processed by the MapReduce system. The simplest method of bolstering RTOLAP for MapReduce is to verify the whole real-time table and procure the latest version prior to the submission time of the OLAP query for every key/value pair (FullScan operation), as the input of the MapReduce job. The key/value supports multi-version concurrency control in case the OLTP queries and OLAP queries are deadlocked by each other. However, this technique is not that accurate becauseprocuring one version for each key/value pair is aexpensive operation in large scale distributed systems. In real time applications, such as social networks, the updates follow a Zipf distribution, and within a time interval, only a minimal portion of keys are updated in the table. Based on this, we effort to accelerate OLAP queries by materializing the real-time data and table towards a data cube. When an OLAP query is produced to the system, it initially connects to *MetaStore*to capture the timestamp of the query for consistency. The statistics preserved in *MetaStore*are also utilized to optimize the query based on our proposed cost model (Section V-C). Post the optimization fromcost model, the OLAP optimization query can be transformed to a MapReduce job which takes as input both the historical values in the data cube and the real-time calue in the key/value store. Efficient access to real-time data, the key/value store is created in a way to support incremental scan (Section III-B1). The real-time data is verified by the IncrementalScan, while the data cube is verified by the FullScan. The IncrementalScantechnique only shuffles the key/value pairs that are modified after the last building of the data cube, and thus is much swifter than FullScan because fewer data are shuffled.

The data cube is aswellretained in the distributed key/value store and is time to time refreshed based on the real-time table. The different versions of the key/value pairs prior the refresh time of the data cube are compressed in order to accelerate the scan time of the real-time data and table. Refreshing the data cube is crucial to the system because if the data cube is refreshed at a faster rate, more data are compacted by the compaction scheme, and fewer real-time data are accessed during the scan operation. In an extreme case where no updates are submitted, the MapReduce job only requiresto scan the data cube. To efficiently refresh the data cube,the updates applied to the key/value store are streamed to thestreaming s ystem, and a real-time data cube is maintainedin the local storage of the streaming system. The real-timedata cube is continuously along with a frequency materialized to the key/value store toreset the data cube. Based on our experimental outcomes, thismethod is much faster than the method of re-computing thedata cube, and the throughput of this method is sufficientlyhigh to process the update streams from the key/value store.

Once this refresh process is accomplished, the timestampof the latest data cube is delivered to *MetaStore*, and the compactionprocess is invoked to compact the real-time data.The*MetaStore*also stores other global information, includingthe submission time of each OLAP query, the frequency ofmaterializing the data cube, etc.

A.1. R-Store Implementations:

Implementations of HBase-R:

HBase is an open source distributed key/value store. A table preserved and put in HBase is partitioned to many*regions*, which are assigned to particular nodes, and each node runs a *region* server to keep care of*regions* and serve the transactions. Insidea particular region, the data of the same column family (a group of columns) are procured and kept in the same structure, as called *store*. A *store* has inbuilt memory structure, *memstore*, and multiple in-disk files, *storefiles*. When a latest version of data is about to be inserted into this *store*, it is first inserted into the *memstore*and appended to the write ahead logs. Once the zapacity of the *memstore*approaches its threshold, the data in the *memstore*are moved to a *storefile*. The *storefiles*are sorted in inverse chronological order.HBase only supports the FullScan operation, so IncrementalScan is designed and implemented in HBase-R.

1) *IncrementalScan:* For a *store* in a particular *region*, by entering the same key across the *storefiles*and *memstore*simultaneously, the IncrementalScan operation scans the keys in a predefined ascending order. For each key, the variation with the bigger timestamp is scanned prior. For all the statedvariations of a key, the algorithm verifies the timestamp of each version and returns the necessary two versions. If the key has only one version, which determines the operation on the key is an insertion, the IncrementalScan only brings back that version for the key.
2) *Compaction:* HBase's compaction process merges all the *storefiles*to one file and keeps only one version for each particular key. The high global compaction in HBase-R is akin to HBase's default, but with a verying triggering condition; local compaction only compacts the versions earlier than a certain timestamp. To keep a verification check that the compaction process does not deadlocks the scan processes, the compacted data are kept in disimilar files, rather than directly replacing the un-compacted data.
3) *Load Balancing:* HBase has its predefined*region* size which is depicted as 256MB. If quantification of the data for a particular *region* is greater than this size, it is on its own split to two sub-*regions*, whichare distributed to other nodes. In HBase's default setting, only a fixed number of versions for a key are stored.

Algorithm 1: Adaptive IncrementalScan

input: Timestamp T_{DC}, Timestamp T_Q, boolean[] DistinctKeys, int NumDistinctKeys

kvMap ← new HashMap<Key, Value>();
for KeyValue kv ∈ MemStore **do**
    if kvMap.contain(kv.key) **then**
        continue;
    else
        kvMap.put(kv.key, kv.value);
NumKeysNotInMemory ← NumDistinctKeys - kvMap.size();
if $CostOfRandom \times NumKeysNotInMemory < CostOfScan \times NumOfUpdatedKeyValues$ **then**
    for key updated but not in kvMap **do**
        kv ← randomRead(key);
        kvMap.put(kv.key, kv.value);
    for each kv before T_{DC} **do**
        if kvMap.exist(kv.key) **then**
            send kvMap(kv.key) and kv;
else
    delete kvMap;
    invoke the default `IncrementalScan`(T_{DC}, T_Q)

Real-Time Data Cube Maintenance:

R-Store brings inHStreaming for keeping care of the real-time data cube (other streaming MapReduce systems are used in R-Store). Every mapper of HStreaming is highly responsible for completing the updates by a range of keys. The map function of the data cube update algorithm is shown in Algorithm 2. When an update for a key is received, the medivial value for this key is retrieved from the local storage. To efficiently retrieve the old value, a clustered index is created for the key/values, and the frequently updated keys are respectively cached in memory. In actual, the updates are generally on a small range of keys, and the old value of the updates have a high probability to be retrieved from the cache. If the key is rudimentary, for each cuboid, one key/value pair is created and mixed to the reducers. The output key is the mix of the dimension attributes, and the map output value is a numeric value. If the key of the update resides in local storage and the upgraded key/value pair falls into the particular cell for a cuboid, one key/value pair is shuffled to the reducer, and the numerical value is equivocal to the described value change. Otherwise, two key/value pairs are created, one is the new value with a tag "+", and the other is the medieval value with a tag "-". The reduce function is called at a time in the interval as specified by the user,*wr*.

Algorithm 2: Map Function for Incremental Update

```
input: KeyValue kv
oldkv = retrieveFromLocal(kv.key);
if oldkv == null then
    for cuboid in data cube do
        CuboidK ← extractCuboidKey(cuboid,
        kv.value);
        CuboidV ← extractCuboidValue(kv.value);
        CuboidV.setTag("+");
        Emit(CuboidK, CuboidV);
    insertToLocal(kv);
else
    oldCuboidV ← extractCuboidValue(oldkv.value);
    oldCuboidV.setTag("-");
    newCuboidV ← extractCuboidValue(kv.value);
    newValue.setTag("+");
    for cuboid in data cube do
        oldCuboidK ← extractCuboidKey(cuboid,
        oldkv.value);
        newCuboidK ← extractCuboidKey(cuboid,
        kv.value);
        if oldCuboidK == newCuobidK then
            newCuboidV.set(computeChangeOfCell
            (oldCuboidV,newCuboidV));
            Emit(newCuboidK, newCuboidV);
        else
            Emit(oldCuboidK, oldCuboidV);
            Emit(newCuboidK, newCuboidV);
    updateToLocal(kv);
```

B. CBA Architecture

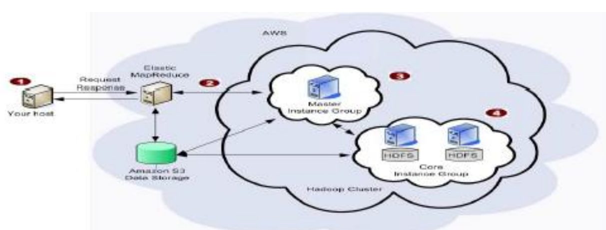

Fig. 3. Architecture of CBA

As is evident from figure 4, the architectural model of CBA permits a user to interact with Amazon S3[21] as well as hadoop cluster, using the web browser. The environment to perform file management within cloud storage is provided by Amazon S3, cloud storage with the help of which a user can upload files in hdfs or can copy a file to hdfs from Amazon S3 in order to perform mapreduce task in hadoop cluster for high performance. After selecting the required files from hdfs, researchers can opt for analysis whose results are achieved after being performed over several datasets in a cluster since computation occurs in distributed environment thereby optimizing the performance of the system. After completion of analysis, the result of analysis is displayed to user in browser and users can store the result of analysis in cloud storage Amazon S3 for future reference.

Amazon EMR allows businessmen, researchers, data analysts, and developers to process vast amounts of data with ease and at a far lesser cost. It uses a hosted Hadoop framework running on the web-scale infrastructure of EC2[22] and Amazon S3. Since CBA is cloud based, availability is one of the features of this system for users equipped with the internet anywhere around the world, all round the clock.

Amazon Elastic MapReduce (Amazon EMR) makes hadoop cluster easy to provision and manage in the AWS Cloud interface. Amazon EMR is available in two different distribution of hadoop, one is Amazon Distribution and next is the MapR Distribution of hadoop. MapR distribution of hadoop comes with additional hadoop application like spark, hive, and so on. MapR distribution also serves support for client and have enhanced many feature of hadoop to make ease-of-use as well as features. MapR distribution of hadoop has good presence in market and most of the leading enterprises like oracle, ibm and so on follow MapR distribution of hadoop. Inorder to perform predictive analytics in cloud platform on the top of MapR distribution of Hadoop Cluster CBA is put to use. Host or Client of CBA can access system after authorization and can perform predictive analysis of bigdata in cloud platform from anywhere with internet connectivity. The CBA is a user friendly system which is a browser based interface and anyone with a little knowledge of computer can interact and deal with the system.

V. CONCLUSION

CBA is a SaaS to analyzebigdata in cloud and is nowadays focused on linear regression and time series analysis. Computing framework distribution, increasing number of nodes in cluster by memory scaling and parallel processing in distributed environment makes it easy to deal with big data and performance thereby decreasing the limitations of processing of large scale data. We now neither have to pick a random sample and end up with an inaccurate result nor do we have to choose vertical scaling and face a dead end.

MapReduce is a parallel execution framework, which hasbeen widely adopted due to its scalability and suitability in a large scale distributed environment. However, most existing works only focus on optimizing the OLAP queries and assume that the data scanned by MapReduce are unchanged during the execution of a MapReduce job. Factually, the real-time outcomes from the latest updated data are more explorative for decision making. In this paper, we propose R-Store for supporting real-time OLAP on MapReduce. R-Store leverages stable technology (HBase and HStreaming) and extends them to achieve high performance and scalability. The storage system of R-Store adopts multi-version concurrency control to support real-time OLAP. The experimental results highlight that the system can bolster real-time OLAP queries highly efficiently analogous to the baseline methods. Though the performance of OLTP degrades slightly due to the competition for resources with OLAP, the response time and throughput remain good and acceptable.

VI. REFERENCES

1. http://hbase.apache.org/.
2. M. Athanassoulis, S. Chen, A. Ailamaki, P. B. Gibbons, and R. Stoica. "Masm: efficient online updates in data warehouses". In SIGMOD, pages865–876, 2011.
3. Y. Cao, C. Chen, F. Guo, D. Jiang, Y. Lin, B. C. Ooi, H. T. Vo, S. Wu, and Q. Xu. Es2: "A cloud data storage system for supporting both oltpand olap". ICDE, pages 291–302, 2011.
4. S. Ceri and J. Widom. "Deriving production rules for incremental viewmaintenance". In VLDB, pages 577–589, 1991.
5. Roger S. Barga, JaliyaEkanayake, Wei Lu, "Project Daytona: Data Analytics as a Cloud Service", IEEE 28th International Conferenceon Data Engineering, 2012.
6. Nikolay Laptev, Kai Zeng, Carlo Zaniolo., "Very Fast Estimation for Result and Accuracy for Big Data Analytics: the EARL System",IEEE's, ICDE Conference 2013.
7. Duggan, M., Ellison, N.B., Lampe, C., Lenhart, A,.and Madden, M."Social Media Update 2014," Pew Research Center, Jan. 2015.
8. Thomas H. Davenport ,JillDyché. "Big Data in Big Companies," May2013.
9. H. V. Jagadish, Johannes Gehrke, AlexandrosLabrinidis, YannisPapakonstantinou, Jignesh M. Patel, Raghu Ramakrishnan, Cyrus Shahabi, "Big data and its technical challenges", ACM, vol. 57, Jul.2014.
10. A. Kemper, T. Neumann, F. F. Informatik, T. U. Mnchen, and DGarching.Hyper: "A hybrid oltp&olap main memory database systembased on virtual memory snapshots". In *In ICDE*, 2011.