ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 11, May 2014

Big Data Analytics: An Approach using Hadoop Distributed File System

P Beaulah Soundarabai[1], Aravindh S[1], Thriveni J[3], K.R. Venugopal[3] and L.M. Patnaik[4]
[1] Department of Computer Science, Christ University, Hosur Main Road, Bangalore, India,
[2] Department of Computer Science and Engineering, University Visvesvaraya College of Engineering, Bangalore University, Bangalore, India.
[3] Honorary Professor, Indian Institute of Science, Bangalore, India

Abstract — Today's world is driven by Growth and Innovation for a better future. All of which are based on analysis and harnessing of tons of data, typically known as Big Data. The tasks involved for achieving results at such a scale can be challenging and painfully slow. This paper works towards an approach for effectively solving a large and computationally intensive problem by leveraging the capabilities of Hadoop and Hbase. Here we demonstrate how to reduce and distribute the large problem across multiple nodes as small chunks and later aggregate the results obtained for the various chunks to arrive at the desired result. This method of distributing and aggregating is called map-reduce, wherein dividing and distributing the problem to different nodes is called Map task and aggregation of results is called Reduce task. As part of implementation we will be analyzing population census data.

Keywords: Hadoop; HDFS; BigData; NoSQL; MapReduce; pig; HIVE.

I. INTRODUCTION

In information technology, big data is a collection of data over a period of time, and can exist in both structured and unstructured form. The data could be anything from pictures, videos and posts from a social media site to climate information collected by weather sensors. This data is so large and complex that it is a challenge to extract, clean, transform, store, analyze and harness useful information. This exercise falls beyond the scope of a RDBMS or traditional data processing applications, simply due to the existence of tons of bytes of data. Yet, it is imperative that we work with such Big Data to derive correlations and other information to aide development, growth, break-through and innovation in various fields. Almost every domain is reliant on information that can be harnessed from Big Data to help move towards progress. [1 to 5] . Big Data can be typically characterized by the three V's –

- Volume – Exponentially growing data.
- Variety – Data comes in all shapes, sizes and forms.
- Velocity – Rate of data creation and the rate at which data is analyzed and harnessed for useful information.

The paper proposes an implementation of Big Data analytics using Hadoop and Hbase. It is important to note that the tests conducted are infrastructure dependent and may vary based on the underlying hardware and network.

Motivation: Big data analysis with the help of famous tools like informatics, vertical etc., help us to get the details of the entire data, get an insight and to make decisions based on the results derived. Most of the tools are not freely available and they require training and maintenance from the experts on the particular tools. We wanted to bring out the benefits of Hadoop freamework which is an open source tool and show how it reduces the implementation time and the cost with respect to the centralized system.

Contribution: In this paper, we have proposed a Big data analysis with HDFS Framework of Hadoop that works towards providing better performance in terms of time, cost and user friendliness. The main objective of our proposed model is to apply business logic on a huge volume of data. We have run eight different queries on 2GB data size of audio conferencing call files and estimated the runtime of the queries. We have applied filtering of the selected attributes and on the queries were made to run on the projected structure.

Organization: The rest of the paper is organized as follows: Section II shows the related literature of other researchers; Background of Hadoop system is present in Section III. Problem Definition and the Methodology are available in Section IV; Section V describes the implementation and the results; Section VI details the Performance Evaluation; Conclusion of the work is presented in Section VII.

II. LITERATURE SURVEY

S Sathya [6] et al., have discussed the problems related the heterogeneous database integration by providing the implementation of Virtual Database Technology (VDB) through Hadoop's MapReduce. They have built a virtual database engine to get the high performance data integration for large volume of heterogeneous data. Zheyuan Liu [7] have presented a detailed study of the resource consumption profiles for MapReduce of Hadoop with respect to CPU and memory by altering the configuration parameters Input output buffers related to input output buffer.

Yanfeng Zhang [8] have focused their research on computations which involve lots of interations on massive data sets. They have evaluated their PrIter that supports the prioritization to converge the iterative results and through which they have achieved better speed over the Hadoop

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 11, May 2014

iterative algorithms. Yongwei Wu [9] et al., have proposed a systematic and practical performance analysis framework, with respect to architecture and design models for defining the structure and behavior of typical master/slave DFSs. They have evaluated the performance of DFA both qualitatively and quantitatively and they also the Hadoop Distributed File System (HDFS).

The performance of MapReduce is checked in three different virtual machines and different number of virtual machines is done by Yang Yang [10] et al., and they have discussed that this performance test in turn would help the researchers to design adaptive scheduling algorithms through which better performance could be achieved with more virtual machines. Ronald C Taylor [11] has presented an overview of Hadoop framework along with HDFS, HBase and MapReduce concepts and the applications which are being benefited in the field of bioinformatics. Skewed workload problem in MapReduce is addressed by Venkata Swamy Martha [12] et al., by hierarchical MapReduce which splits the heavy tasks into children tasks and they are again given to MapReduce as a new job recursively till the job becomes fit to get executed.

The detailed study on the merits and demerits of in-storage processing on current Solid-State Disk (SSD) architectures is presented by Yangwook Kang [13] et al., and the workflow manager developed with the name Nova and deployed at Yahoo and gets excuted in Hadoop clustered framework. Two models named flat platform performance model and workflow performance model have been proposed and tested by Zhuoyao Zhang et al., which increase the accuracy, effectiveness and performance of MapReduce frame work of Hadoop[14] [15]. Various performance increasing models with workload sharing and management, configuration redesigning have been proposed by researchers in various scenarios. [16-20].

III. BACKGROUND

A. *MapReduce*

Map and Reduce are the two major features of Hadoop's MapReduce model. It generally processes the datasets which are of huge models by creating the multiple dataset and applying the execution on all of these clusters parallely.

At the core of its technique MapReduce typically follows the Divide and Conquer policy, in that Map and Reduce are two disparate and distinct jobs. In a distributed environment, the primary node divides the input into smaller workloads and distributes it to all the secondary nodes. The secondary nodes may in-turn divide and distribute the workload further if possible. The secondary nodes work on the assigned workload and return the results back to the primary node. This process is known as the Map task. The results from secondary nodes are aggregated and processed to reach the desired result by the primary node. This process is known as the Reduce task. However it is important to note that for a effective MapReduce processing each of these Map tasks should not be interdependent, allowing parallelism.

B. *Apache Hadoop*

Hadoop is a Java based open source software being developed by a global community of developers and is licensed under the Apache V2 license. It is pioneered for data intensive distributed applications. The fundamental concept implemented in Hadoop is MapReduce which is the key algorithm used for distribution activities. Hadoop can be visualized as three major components – Kernel, MapReduce Engine and Hadoop Distributed File System (HDFS).

Hadoop is widely popular and has been implemented across major organizations to handle large distributed computing applications. Although the implementation of Hadoop is scalable, fault-tolerant and flexible, it is however governed by all the limitations of MapReduce. Hence, it is crucial to evaluate if the implementation of Hadoop is the answer to the business need at hand.

C. *HDFS*

Hadoop Distributed File System Consists of two main nodes namely DataNode and NameNode. The DataNodes contains the original file which were split into many parts and kept across the entire cluster and the NameNode contains the namespace tree and the mapping of the namespace tree blocks into its actual DataNode. For the availability and fault tolerance reasons each DataNode gets replicated into number of times and the NameNode maintains the details of the replicated DataNodes. Each DataNode is capable of executing multiple application queries in parallel.

D. *NoSQL*

NoSQL is a classification of Database Management Systems and is characterized by its non-usage of SQL as a means for querying the database. NoSQL thrives on the fact that RDBMS may not be suitable for all the situations, especially when the data available is voluminous and does not particularly display any sort of relationships. It helps build a fault tolerant and distributed architecture where the data is partitioned and stored across different machines, citing performance and space limitations. However NoSQL databases do not guarantee preservation of the ACID (Atomicity, Consistency, Isolation, and Durable) properties.

E. *HBase*

HBase is a Java based open source, NoSQL, distributed database. It is a column oriented database system with an underlying implementation of Hadoop Distributed File System (HDFS). An HBase database comprises of a set of tables which contains rows and columns, similar to a traditional database. Each HBase column represents an attribute of an object and a collection of such columns are known as column families.

A requirement for a distributed, random, scalable big data store could make use of HBase's full potential. It is flexible in that it allows the schema of the tables to be modified at any point in time, thereby making it very apt to be used in an environment of constantly changing requirements. Being

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 11, May 2014

a distributed database means that HBase implements the concepts of a primary master node and secondary nodes in a cluster or grid.

IV. PROBLEM DEFINITION AND METHODOLOGY

A. *Problem Definition*

Given an audio conference call dataset of 2GB size which is been taken from the open source, we wanted to evaluate the dataset with the following eight different aspects.

1. Finding the Average number of participants for each conference call. (Conference_Id, CustomerID, CustomerName, average (ActualNumOfParticipants))

2. Customerwise total amount (CustomerID, CustomerName, sum (ActualCallDuration * $PricePerMin))

3. Total time of all calls. (Per day, Per Week, Per month High frequency period (ActualCallDuration))

4. Total amount for all calls. ((Actual Call Duration * $PricePerMin) - (RetailBridgingCharges + RetailOtherCharges + RetailPromotionCharges + RetailTaxAmount) - WholesaleBridgingCharges + WholesaleOtherCharges + WholesaleOtherCharges + WholeSaleTaxAmount))

5. Maximum number of participants. (ActualNumOFParticipants, conference_id)

6. Number of conferences by customer (Group customerId, sum (conference_id)).

7. Total customer Ids in a year.

8. Calls which only have minimum (2) participants.

Query 1 is mainly used to find the average available number of participants for each of the conference call. Any conference call usually begins with minimum of three participants and other participants would join and leave in between. A record is created for every call (minimum call duration) for each of the participant. So, there are lots of record for each participant for a single conference_id. With the help of the conference_id, we can find, how many participants were there on an average for a particular conference call.

The second query helps the service providers to find out what is the business of each customer to them and based on this result they can come up with some marketing offers for the targeted customers.

Third query finds out the total time of all calls with respect to a single day, week, and month and even during the busy hours of a day.

Query 4 is to show that various mathematical calculations are also possible in Hadoop with less execution time. We have done a sample calculation of addition and subtraction to find the total amount for all the calls which need not be the same. Based on the requirement, the query can be tailored. The fifth query helps us to find out how many people participated in a particular conference totally.

Next query groups all the customer records of a particular customer_id and then finds out how many different conferences he / she has participated in. This may be done for a day or a week or month also.

To find the total number of customers in a particular year, we have used the 7th query. To exactly find out how many calls have been made between any two customers, we have used the last query. This is also mainly used in marketing. To further decide a business, we can find how many customers make calls frequently to other person in one – one call and he can be targeted for any business offers.

B. *Algorithm*

We have considered the following nine steps to perform the - queries in Hadoop – Pig framework model.

1. Take all Call Detail Records (CDRs) of size 2GB.
2. Consolidate all the CDRs into a single file.
3. Load the file into HDFS. (We have already setup the HDFS).
4. Load the file into memory (PIG)
5. Apply Transformations and Projection logic on the loaded data.
6. Save the projection results back on HDFS as a different file.
7. Create a table in HIVE with the projected structure.
8. Load the pig output file into this table.
9. Run the business queries on this table.

In the first three steps, we have taken all the CDRs and fed them into a single file with name *audconCdrs.* There are more than 80 attributes available in each CDR and this single consolidated file is stored in the HDFS. The command for the consolidation of all CDRs is given below.

(Loading all files into 1 consolidated file in HDFS.)
audconCdrs= load '/pig-works/cdrs/audcon /AXPREMIERE20112060033.csv' using PigStorage(',') as (InvoiceID:long, InvoiceDate:chararray, ReservationID:long, ReservationDate:chararray, CustomerID:long, CustomerName:chararray, CustomerAccountID:long, AccountOwner:chararray, conference_nm:chararray, moderator_nm:chararray, ScheduledTime:chararray, ScheduledLength:int, ScheduledNumOfParticipants:int, billing_cd_1:chararray, billing_cd_2:chararray, billing_cd_3:chararray, billing_cd_4:chararray, billing_cd_5:chararray, billing_cd_6:chararray, billing_cd_type_1:chararray, billing_cd_type_2:chararray, billing_cd_type_3:chararray, billing_cd_type_4:chararray, billing_cd_type_5:chararray, billing_cd_type_6:chararray, TimeZone:chararray, currency:chararray, filler_1:chararray, scheduler_phone:chararray, SchedulerName:chararray, SchedulerAddress1:chararray,

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 11, May 2014

SchedulerAddress2:chararray,
SchedulerAddress3:chararray,
SchedulerAddress4:chararray, SchedulerCity:chararray,
SchedulerStateCountry:chararray,
SchedulerZipCode:chararray, BillingAddress1:chararray,
BillingAddress2:chararray, BillingAddress3:chararray,
BillingAddress4:chararray, BillingCity:chararray,
BillingStateCountry:chararray, BillingZipCode:chararray,
PlatformID:int, PlatformName:chararray,
GenProductID:chararray, product_name:chararray,
CallStart:chararray, CallTime:int, participant_nm:chararray,
calledno:long, UsageType:chararray,
ActualNumOFParticipants:int,
ActualCallDuration:chararray, Origin:chararray,
ParticipantDuration:chararray, RetailBridgingCharges:float,
Filler10:chararray, RetailOtherCharges:float,
RetailPromotionCharges:chararray,
WholesaleBridgingCharges:int, Filler10_dup:chararray,
WholesaleOtherCharges:int,
WholesalePromotionCharges:int, room_no:chararray,
RetailTaxAmount:float, WholeSaleTaxAmount:float,
conference_id:long, seqnum:long, vendor_id:int);

The third step is to load the file (audconCdrs) into HDFS and in the fourth step; we have loaded the file into the memory of Pig. The fifth step is to apply the transformation and projection logic on the loaded data which filters only the needed fields by discarding the others from the 80+ attributes. Many fields were eliminated and there are few new fields have been added with some mathematical calculations. *autconProj* is the name of the new file name which has the filtered fields of the old file *audconCdrs*. The code for this action is as follows:

audconProj= foreach audconCdrs generate InvoiceID,
InvoiceDate, CustomerID, CustomerName,
AccountOwner, TimeZone, currency, BillingCity,
BillingStateCountry, product_name, CallStart, CallTime,
UsageType, ActualNumOfParticipants, ActualCallDuration,
RetailBridgingCharges, WholesaleBridgingCharges,
WholeSaleTaxAmount, RetailTaxAmount,
RetailBridgingCharges * RetailTaxAmount,
WholesaleBridgingCharges * WholeSaleTaxAmount;

For creating the table in Hive, we have used the following code and the table name is *audcon*.
create table audcon (InvoiceID int, InvoiceDate string,
CustomerID int, CustomerName string, AccountOwner
string, TimeZone string, currency string, BillingCity string,
BillingStateCountry string, product_name string, CallStart
string, CallTime int, UsageType string,
ActualNumOFParticipants int, ActualCallDuration string,
RetailBridgingCharges float, WholesaleBridgingCharges
float, WholeSaleTaxAmount float, RetailTaxAmount float,
RetailCallCharge float, WholeSaleCallCharge float) ROW
FORMAT DELIMITED FIELDS TERMINATED BY ',';

As the final step, the eight business queries were run on this particular table and we have checked the execution time for each of the query independently.

The huge volume of real time data cannot be analyzed by the centralized system and Hadoop framework offers MapReduce model for processing such large datasets. It is a framework for providing a distributed hierarchical solution for the complicated problems with a huge volume of data. We have extracted only the required variables from the CDRs' micro data for our analysis.

V. IMPLEMENTATION AND RESULTS

A. *Simulation Background*

The setup used for the above implementation includes a machine installed with Ubuntu, HBase database and Java version 6. The above given variables are then fed into the HBase database tables in different column families. Each column family consisting of four to five relative columns. The data size chosen for implementation is of 2 GB.

B. *Implementation*

To feed the data in the database we used a Mapper which reads the data from the extracted data set, creates different column families and inserts the data in the HBase table. Later the data is read using HBase scan functionality. The read data is analyzed using a reducer by dividing the records in smaller chunks and later aggregating the same.

In our implementation we have considered the case of analyzing the eight special queries. The result for the application is as given below. The execution was done on two different machines with Hadoop framework as well as on a standalone centralized system.

Table 1 shows the execution time comparison of the eight various queries processed in Hadoop framework and on the centralized system. For 2 GB size of data, the centralized system takes almost double the amount of time which was taken by Hadoop framework. Hadoop System requires minimum of 2 to 3 minutes overhead time to setup the system and then the original processing starts. When we have a big size of data, it takes less time to do the processing as the setup is required only once for each problem and the processing time for the centralized system would go exponentially with respect to the time taken by Hadoop system.

The various advantages of Hadoop framework are,

- Scalability – Hadoop stores a huge volume of data (in tera bytes) in the entire distributed sytem in many machines and if the size of the data grows in future, the data and processing can happen by adding few more machines as it is a distributed approach.

- Speed – The map or reduce jobs get executed in each copy of the data chunks on various machines at a time, it has the high advantage of reducing the network utilization. Since the processing of data sets is happening in parallel, the overall processing time reduces drastically.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 11, May 2014

- Economy – Big data analytic tools like informatics needs a huge cost for the software wherein Hadoop is an open source framework by Apache and it is freely available. So, it does not require any cost to get the software model and also does not require much economy or time for the training.

- Reliability and Fault Tolerance – While storing the data across the machines in the distributed system, Hadoop also replicates them into minimum of 3 copies which provides high availability of data and it achieves fault tolerance and load balancing. It also moves the data occasionally from machine to machine and thus attains location transparency.

- Flexibility – Hadoop can work with any type of data and it is compatible with various sources of data also. It is capable of bringing data from multiple sources and can work on them to derive intelligence out of them.

Table 1. Queries' Execution Time Comparison

Query No.	Execution Time (in minutes)	
	Hadoop - Pig - Hive	Centralized System
1	5.07	9.58
2	4.55	9.53
3	5.14	10.1
4	4.47	9.52
5	5.02	9.55
6	5.19	10.14
7	4.5	9.47
8	4.12	8.2

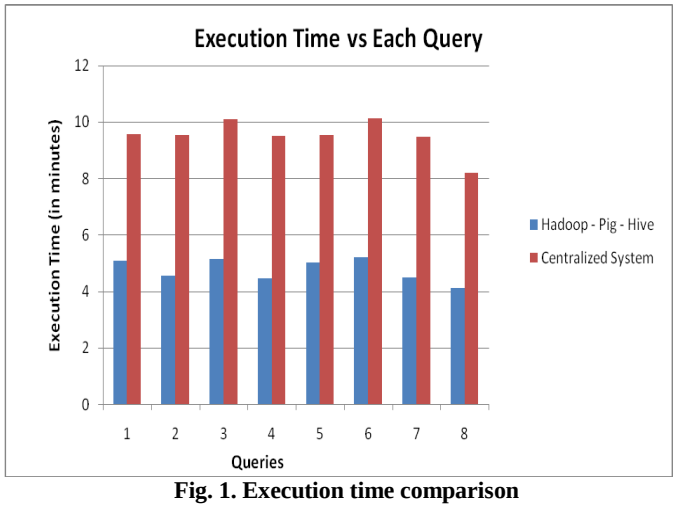

Fig. 1. Execution time comparison

Fig. 1 shows the execution time comparison of our eight queries with respect to Hadoop framework with 2 systems and a centralized system. The Hadoop HDFS has shown a better performance and the implementation time is almost half the time of the centralized system. This is due to the distribution of data and the parallel processing of them.

VI. CONCLUSIONS

Hadoop framework gives flexibility to the programmer to choose the feature that he is comfortable to code with. Since writing a lot many lines of code in Java would be too difficult to work on the relational database model and so we have chosen the Hive and the Pig models to work on the datasets and Pig has the advantage of creating MapReduce code for us.

We have executed business logic on a 2GB dataset and could achieve a better performance when we have executed it in Hadoop and on a centralized system. In future we will execute the same execution on different virtual machines and would analyze the behavior of the execution.

REFERENCES

[1] Hadoop - Apache Software Foundation project home page. [http://hadoop.apache.org/].

[2] Lam C, Warren J: Hadoop in Action. Manning Publications; 2010.

[3] Venner J: Pro Hadoop. New York: A Press; 2009.

[4] White T: Hadoop: The Definitive Guide. Sebastopol: O'Reilly Media; 2009.

[5] Chunk Lam, Hadoop In Action, Manning Publications Co., 2011.

[6] S.Sathya, M.Victor Jose, Application of Hadoop MapReduce Technique to Virtual Database System Design, Proceedings of ICETECT, 2011.

[7] Zheyuan Liu, Dejun Mu, Analysis of Resource Usage Profile for MapReduce Applications Using Hadoop on Cloud, 2012 / IEEE / 978-1-4673-0788-8.

[8] Yanfeng Zhang, Qixin Gao, Lixin Gao, and Cuirong Wang, PrIter: A Distributed Framework for Prioritizing Iterative Computations, IEEE Transactions on Parallel And Distributed Systems, Vol. 24, No. 9, September 2013.

[9] Yongwei Wu, Feng Ye, Kang Chen, and Weimin Zheng, , Modeling of Distributed File Systems for Practical Performance Analysis IEEE Transactions on Parallel And Distributed Systems, Vol. 25, No. 1, January 2014.

[10] Yang Yang, Xiang Long, Xiaoqiang Dou, Chengjian Wen, Impacts of Virtualization Technologies on Hadoop, Third International Conference on Intelligent System Design and Engineering Applications, 2013

[11] An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics, Ronald C Taylor, The 11th Annual Bioinformatics Open Source Conference (BOSC) 2010.

[12] VenkataSwamy Martha, Weizhong Zhao, Xiaowei Xu, h-MapReduce: A Framework forWorkload Balancing in MapReduce, IEEE 27th International Conference on Advanced Information Networking and Applications, 2013.

[13] Yangwook Kang, Yang-suk Kee, Ethan L. Miller, Chanik Park, Enabling Cost-effective Data Processing with Smart

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 11, May 2014

SSD, 29th IEEE Symposium on Massive Storage Systems and Technologies (MSST 13).

[14] Christopher Olston and Xiaodan Wang, Nova: Continuous Pig/Hadoop Workflows, SIGMOD'11, June 12–16, 2011.

[15] Zhuoyao Zhang, Ludmila Cherkasova and Boon Thau Loo, Getting More for Less in Optimized MapReduce Workflows, IFIP, 2013.

[16] H. Herodotou, F. Dong, and S. Babu. No one (cluster) size fits all: Automatic cluster sizing for data-intensive analytics. In Proc. of ACM Symposium on Cloud Computing, 2011.

[17] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. Cetin, and S. Babu. Starfish: A Self-tuning System for Big Data Analytics. In Proc. of 5th Conf. on Innovative Data Systems Research (CIDR), 2011.

[18] F. Tian and K. Chen. Towards Optimal Resource Provisioning for Running MapReduce Programs in Public Clouds. In Proc. of IEEE Conference on Cloud Computing (CLOUD 2011).

[19] Verma, L. Cherkasova, and R. H. Campbell. ARIA: Automatic Resource Inference and Allocation for MapReduce Environments. Proc. of the 8th ACM International Conference on Autonomic Computing (ICAC), 2011.

[20] Z. Zhang, L. Cherkasova, A. Verma, and B. T. Loo. Automated Profiling and Resource Management of Pig Programs for Meeting Service Level Objectives. In Proc. of IEEE/ACM Intl. Conference on Autonomic Computing (ICAC), 2012.