Processing NoSQL Datasets with Hadoop-Streaming

[1]Miss. Varsha S Dahatonde, [2]Prof. Ashish Kumar

[1]PG Student, [2]Professor, [1,2]Department Computer, G H Raisoni COEM Chas, Maharashtra, India.

[1]vaishudahatonde@gmail.com, [2]ashish.kumar@raisoni.net

Abstract: **Unstructured data generated from industries, scientific datasets and governmental data creating a problem for storage. This kind of data requires high processing, more time for execution but it is not provided by traditional datasets. To process unstructured data i.e. NoSQL in this paper we have used Hadoop streaming as it is used for processing non-java executable. To provide fast processing to NoSQL dataset like Cassandra or MongoDB and here we are going to use hadoop streaming tool. Also going to analyze the comparison for processing both datasets. As to observe whether MongoDB and Cassandra supports different features for same database or not. Here through comparison at the end are able to decide which system should be used for small and large database processing.**

Keywords* — NoSQL, *Non-java executable, Unstructured, Streaming

I. INTRODUCTION

SQL is easy to process as it is structured language and in the relational database form also predefined fields are available for accessing it. On other hand unstructured data can't be accessable through any predefined fields as it doesn't support relational database. The main problem while handling the unstructured database is about the storage and processing of the data requires high computational resources. NoSQL database is coded in district programming languages and available as open source software. Objective of this paper is to handle the unstructured data using widely used NoSQL database system hadoop streaming for Cassandra and MongoDB datasets [1]. The existing work uses Map Reduce pipeline that is adopted by Hadoop streaming and MARISSA. For evaluation of data the pipeline have three stages: Data preparation, Data Transformation and Data Processing [1].

In earlier days for resolving and storing the data , separate databases were required. Now-a-days , Nosql provides resolution to the matter of information isolation by victimization datasets, it share the constant context however not the constant structure or format, to be collected along. It allows the information not solely to be hold on within the same tables however to late on be analyzed jointly. once non-uniform information grows to massive sizes but, a distributed approach to research unstructured information has to be thought-about.

Various sources generate massive information (big data). Internet, Web, on-line however the input file is sometimes massive and also the computations have to be compelled to be distributed across thousands of machines so as to complete in an exceedingly affordable quantity of our time . The issues of a way to set the computation, distribute the data, and handle failures conspire to obscure the initial straightforward computation with massive amounts of advanced code to trot out these issues.

As a reaction to the current quality , process pipeline permitting not solely native Hadoop MapReduce programs written in Java to create use of the NoSQL storage systems, however conjointly any non- Java feasible used for processing. Such a pipeline is helpful in applications that need a web instruction execution platform.

NoSQL information stores provide not solely the potential of storing massive, loosely structured information that may later be analyzed and deep-mined as an entire, however they conjointly provide the flexibility for queries to be applied on such information. This can be particularly useful once real time answers area unit required on solely slices of the hold on information. Despite the presence of this valuable instruction execution potential from NoSQL stores, there's a necessity for a software pipeline permitting "Big Data" to faucet NoSQL information stores as sources of input. There's conjointly a necessity for a software pipeline permitting MapReduce heritage programs written in C, C++, and non-Java executables to use "Big Data" technologies.

II. Related Work

Existing system comprises a process pipeline permitting not solely native Hadoop MapReduce programs written in Java to form use of the NoSQL storage systems, however conjointly any non-Java feasible used for processing. Such a pipeline is beneficial in applications that need associate degree If you want to submit your file with one column electronically, please do the following: offline instruction execution platform. they'd use Apache prophetess in their analysis, a well known NoSQL answer, and mix it with each Apache Hadoop and a Map scale back answer of their own MARISSA [3]. They show on an item-by-item basis once it's helpful to method the info directly from NoSQL stores and also the performance impact of initial downloading it to a MapReduce frame-work.

E. Dede have planned 2 completely different approaches, one operating with the distributed prophetess cluster [3] on to perform MapReduce operations and also the alternative mercantilism the dataset from the info servers to the beer system for more process.

E. Dede have given the analysis that to do progressive transition in both fields like scientific and industrial datasets has been the big task behind the developer and researcher in the NoSQL data model. To process Loosely structured data is a challenge for early data store systems, and while working with such database i.e unstructured are very expensive and time consuming. As the quantity of unstructured data grows, so does the demand for a processing pipeline that is capable of seamlessly combining the NoSQL storage model and a ”Big Data” processing platform such as MapReduce. Although, MapReduce is the paradigm of choice for data-intensive computing, Java-based frameworks such as Hadoop require users to write MapReduce code in Java. Hadoop Streaming, on the other hand, allows users to define non-Java executables as map and reduce operations.[1]

MapReduce is a basically programming model developed by Google for processing and generating large data sets in distributed environments[2]. Many real-world tasks can be implemented by two functions, map and reduce. MapReduce plays a key role in Cloud Computing, since it decreases the complexity of the distributed programming and is easy to be developed on large clusters of common machines. Hadoop, an open-source project, is used to implement Google MapReduce architecture. It is wildly used by many applications such as FaceBook, Yahoo, Twitter, and so on. However, it is difficult to decouple an application into functions of map and reduce for common users. In this paper, they have develop a web-based graphic user interface for ordinary users to utilize MapReduce without the real programming. Users only have to know how to specify their tasks in target-value-action tuples. Real examples are provided for demonstration.[2].

In the last decade, the increased use and growth of social media, unconventional web technologies, and mobile applications, have all encouraged development of a new breed of database models. NoSQL data stores target the unstructured data, which by nature is dynamic and a key focus area for ”Big Data” research. New generation data can prove costly and unpractical to administer with SQL databases due to lack of structure, high scalability, and elasticity needs. NoSQL data stores such as MongoDB and Cassandra provide a desirable platform for fast and efficient data queries. This leads to increased importance in areas such as cloud applications, e-commerce, social media, bioinformatics, and materials science. In an effort to combine the querying capabilities of conventional database systems and the processing power of the MapReduce model, this paper presents a thorough evaluation of the Cassandra NoSQL database when used in conjunction with the Hadoop MapReduce engine.[3]

MapReduce has since its inception been steadily gaining ground in variousscientific disciplines ranging from space exploration to protein folding. The model poses a challenge for a wide range of current and legacy scientific applications for addressing their Big Data challenges. For example: MapRe-duce’s best known implementation, Apache Hadoop, only offers native support for Java applications. While Hadoop streaming supports applications compiled in a variety of languages such as C, C++, Python and FORTRAN, streaming has shown to be a less efficient MapReduce alternative in terms of performance, and effectiveness. Additionally, Hadoop streaming offers lesser options than its native counterpart, and as such offers less flexibility along with a limited array of features for scientific software.[4]

Many analysis works [14-15] gift results involving the performance of a prophetess info system for large knowledge volumes. during this paper, we've set to guage the performance of prophetess NoSQL info system specifically for genomic knowledge.

III. PROBLEM STATEMENT

The Mapreduce framework has created complicated large-scale processing simple and economical. Despite this, MapReduce is meant for instruction execution of huge volumes of knowledge, and it's not appropriate for recent demands like time period and on-line process. MapReduce is inherently designed for prime turnout instruction execution of massive knowledge that take many hours and even days, whereas recent demands square measure additional focused on jobs and queries that ought to end in seconds or at the most, minutes. [1,2] The info we elect depends on many factors, the sort of knowledge we have a tendency to store, the categories of queries we have a tendency to do, our operations needs and infrastructure, and also the tools out there for it.

In the bank there takes place a huge transaction every month if they want a data of particular user on particular day it is not possible for them to analyze such a huge data with relational database or if the bank application is an non java application then it is not an easy task to convert non java application to java based application. Thus our proposed system will allow not only processing Hadoop MapReduce programs written in java to make use of NoSQL storage system but also any non java executable used for data processing.

IV. IMPLEMENTATION DETAILS

A. System Summary

MongoDB and prophetess have a typical goal high measurability however they take terribly completely different study approaches to induce there. during this section well cowl every of take issueentlthe various} aspects of the 2 architectures and highlight however the 2 databases differ.

B. Module Description

The planned system comprises following elements :

1. Data Preparation

Data Preparation is that the step of downloading the info from prophetess and Mongodb servers to the corresponding filing systems - HDFS for Hadoop Streaming and also the shared file system for MARISSA. For each of those frameworks this step is initiated in parallel. prophetess permits mercantilism the records of a dataset in JSON formatted files [9]. Exploitation this feature, every node downloads the info from the native prophetess server to the filing system. In our experimental setup, every node that's running a prophetess server is additionally a employee node for the MapReduce framework in use

2. Data Transformation (MR1)

Cassandra permits users to export datasets as JSON formatted files. As our assumption is that the MapReduce applications to be run ar inheritance applications that ar either not possible or impractical to be changed and therefore the input file must be born-again into a format that's expected by these target executables. For this reason, our software system pipeline includes a MapReduce stage, wherever JSON information will be reworked into alternative formats. during this part every input record is processed to be born-again to a different format and keep in inter mediator output files. This step doesn't involve any information or process dependencies between nodes and so may be a nice appropriate the MapReduce.

3. Data Processing (MR2)

This is the ultimate step of the MapReduce Streaming pipeline. we tend to run the non-java executables, over the output of MR1

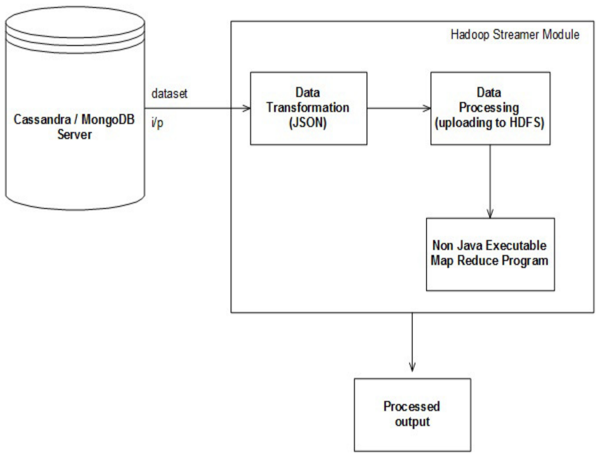

Cassandra / MongoDB Server
dataset
i/p
Hadoop Streamer Module
Data Transformation (JSON)
Data Processing (uploading to HDFS)
Non Java Executable Map Reduce Program
Processed output

Fig 1. Planned system design

C. MATHEMATICAL MODULE

The planned system model will be explained by following parameters:

1. Practicability Study Consider the system " s" as

- S={Ta,A,Op,Su,Fa}
- Ta={T1,T2,T3,…….Tn} , No. of System User.
- A= {a1}, each user having only one account and contains user ID.
- Op={Op1,Op2,Op3,….Op7}, Operations performed by User System

- Su-success {connection establish, sending or retrieving data successfully
- Fa-failure {not a valid user, incorrect password}
- MATHEMATICAL REPRESENTATION

  N
  Σ Ai=1 NOTE: 1=true , 0=false

  i=1
  Then user is already exists.
  Else
  Add new account in it.
  Ai contain user id.
- To login in Account :

  N
  Σ Ai =1 then Σ Ai = Σ Ud =1 i=1
  i=1 i=1

(verifies parole in User Details in line with the User authority ID gift in Account)

If and given that account info and user details ar match i.e account info that contain User id if match with user Id and parole of user authority details then Login Success

Else
login failed

- User Authority Operations:

  n
  ΣOpi =1 then Operation Success
  i=1

perform Link Details for check Hadoop Streaming approach, Data Streaming approach and Perform Analysis.

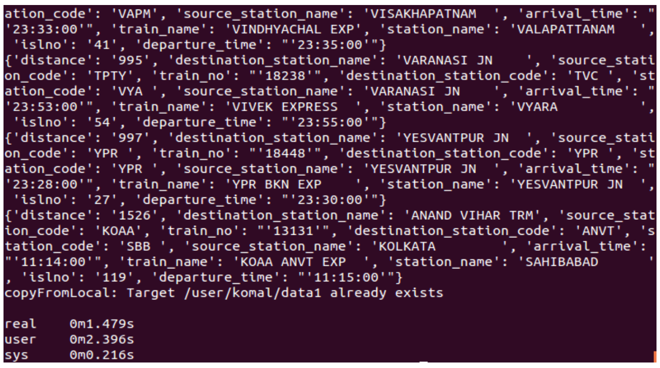
Else
Operation Failed
show result "not any approach found".

V. RESULT

In this section same database were provided for the both Cassandra and MongoDB for processing using hadoop streaming.

In Fig 2. Dataset is processed through Cassandra where performance is calculated on the basis of 3 parameters i.e. Real time, User time and system time for the processing database. Same for MongoDB is performed and result were shown in fig 3.

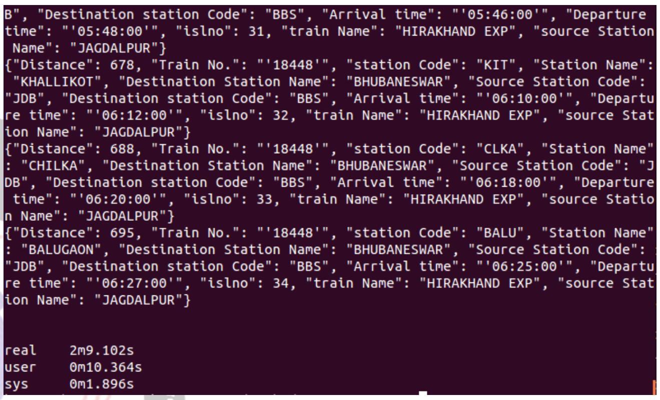
As Hadoop is very relevant tool in processing data which are huge in size When scope of project is finalized Hadoop gives priority. The Existing System mainly focus only on processing the dataset which are downloaded from the Cassandra server, where user has only one choice to process the dataset as well as user is mandate to know querying in Cassandra.



```
ation_code': 'VAPM', 'source_station_name': 'VISAKHAPATNAM  ', 'arrival_time': "
'23:33:00'", 'train_name': 'VINDHYACHAL EXP', 'station_name': 'VALAPATTANAM   ',
 'islno': '41', 'departure_time': "'23:35:00'"}
{'distance': '995', 'destination_station_name': 'VARANASI JN    ', 'source_stati
on_code': 'TPTY', 'train_no': "'18238'", 'destination_station_code': 'TVC ', 'st
ation_code': 'VYA ', 'source_station_name': 'VARANASI JN    ', 'arrival_time': "
'23:53:00'", 'train_name': 'VIVEK EXPRESS  ', 'station_name': 'VYARA          ',
 'islno': '54', 'departure_time': "'23:55:00'"}
{'distance': '997', 'destination_station_name': 'YESVANTPUR JN  ', 'source_stati
on_code': 'YPR ', 'train_no': "'18448'", 'destination_station_code': 'YPR ', 'st
ation_code': 'YPR ', 'source_station_name': 'YESVANTPUR JN  ', 'arrival_time': "
'23:28:00'", 'train_name': 'YPR BKN EXP    ', 'station_name': 'YESVANTPUR JN  ',
 'islno': '27', 'departure_time': "'23:30:00'"}
{'distance': '1526', 'destination_station_name': 'ANAND VIHAR TRM', 'source_stat
ion_code': 'KOAA', 'train_no': "'13131'", 'destination_station_code': 'ANVT', 's
tation_code': 'SBB ', 'source_station_name': 'KOLKATA        ', 'arrival_time':
"'11:14:00'", 'train_name': 'KOAA ANVT EXP  ', 'station_name': 'SAHIBABAD      '
, 'islno': '119', 'departure_time': "'11:15:00'"}
copyFromLocal: Target /user/komal/data1 already exists

real    0m1.479s
user    0m2.396s
sys     0m0.216s
```

Fig 2. Performance of Cassandra dataset



```
B", "Destination station Code": "BBS", "Arrival time": "'05:46:00'", "Departure
time": "'05:48:00'", "islno": 31, "train Name": "HIRAKHAND EXP", "source Station
 Name": "JAGDALPUR"}
{"Distance": 678, "Train No.": "'18448'", "station Code": "KIT", "Station Name":
 "KHALLIKOT", "Destination Station Name": "BHUBANESWAR", "Source Station Code":
"JDB", "Destination station Code": "BBS", "Arrival time": "'06:10:00'", "Departu
re time": "'06:12:00'", "islno": 32, "train Name": "HIRAKHAND EXP", "source Stat
ion Name": "JAGDALPUR"}
{"Distance": 688, "Train No.": "'18448'", "station Code": "CLKA", "Station Name"
: "CHILKA", "Destination Station Name": "BHUBANESWAR", "Source Station Code": "J
DB", "Destination station Code": "BBS", "Arrival time": "'06:18:00'", "Departure
 time": "'06:20:00'", "islno": 33, "train Name": "HIRAKHAND EXP", "source Statio
n Name": "JAGDALPUR"}
{"Distance": 695, "Train No.": "'18448'", "station Code": "BALU", "Station Name"
: "BALUGAON", "Destination Station Name": "BHUBANESWAR", "Source Station Code":
"JDB", "Destination station Code": "BBS", "Arrival time": "'06:25:00'", "Departu
re time": "'06:27:00'", "islno": 34, "train Name": "HIRAKHAND EXP", "source Stat
ion Name": "JAGDALPUR"}


real    2m9.102s
user    0m10.364s
sys     0m1.896s
```

Fig 3. performance of mongodb dataset

After the analyzing we found that user should get an option to select the database of his own choice and perform operation manually without any understanding of the query language Hence we prefer Hadoop streaming which uses Hadoop framework that supports distributed computing specifically MapReduce algorithm. Because of this we can process large amount of data in efficientway and in less amount of time.

VI. CONCLUSION

Processing pipeline permitting not solely native Hadoop MapReduce programs written in java to the NoSQL storage systems, however conjointly any non-Java practicable used for processing NoSQL databases or new tests victimization prophetess with totally different hardware configurations seeking enhancements in performance. Considering the performance of prophetess and MongoDB info will certainly facilitate within the process of unstructured information. additional it's attainable to stipulate new approaches in studies of process the unstructured information. Through process prophetess and MongoDB in single system can facilitate USA to execute totally different queries on non-java executables.

REFERENCES

[1] Agneeswaran, V. *Big Data Analytics Beyond Hadoop: Real-Time Applications with Storm, Spark, and More Hadoop Alternatives*; Pearson FT Press: Upper Saddle River, NJ, USA, 2014.

[2] Kambatla, K.; Kollias, G.; Kumar, V.; Grama, A. *Trends in big data analytics*. J. Parallel Distrib. Comput. 2014, 74, 2561-2573.

[3] E. Dede, Z. Fadika, J. Hartog, M. Govindaraju, L. Ramakrishnan, D. Gunter, and R. Canon. Marissa: *Mapreduce implementation for streaming science appli-cations*. In E-Science (e-Science), 2012 IEEE 8th International Conference on,pages 1-8, 2012.

[4] A. Lakshman and P. Malik. *Cassandra: structured storage system on a p2p Network* . In Proceedings of the 28th ACM symposium on Principles of distributed computing, PODC 09, pages 55, New York, NY, USA, 2009. ACM.

[5] Apache Hadoop. http://hadoop.apache.org .

[6] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. *The hadoop distributed file system*. In Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on, pages 1 10, May 2010.

[7] E. Dede, Z. Fadika, J. Hartog, M. Govindaraju, L. Ramakrishnan, D. Gunter, and R. Canon. Marissa: *Mapreduce implementation for streaming science appli-cations*. In E-Science (e-Science), 2012 IEEE 8th International Conference on,pages 18, 2012.

[8] National Energy Research Scientific Computing Center. http://www.nersc.gov.

[9] L. Ramakrishnan, P. T. Zbiegel, S. Campbell, R. Bradshaw, R. S. Canon, S.Coghlan, I. Sakrejda, N. Desai, T. Declerck, and A. Liu. Magellan: experiences 32 from a science cloud. In Proceedings of the 2nd international workshop on Scientific cloud computing, ScienceCloud 11, pages 4958, New York, NY, USA, 2011. ACM.

[10] Fadika, Zacharia and Govindaraju, Madhusudhan and Canon, Shane and Ra-makrishnan,Lavanya. *Evaluting hadoop for data-intensive scienti_c operations*. IEEE Cloud Computing, 2012.

[11]Cassandra wiki , operations. http://wiki.apache.org/cassandra/ Operations.

[12] Elif Dede, Bedri Sendir, Pinar Kuzlu, Jessica Hartog, Madhusudhan Govindaraju: "*An Evaluation of Cassandra for Hadoop*", Grid and Cloud Computing Re-search Laboratory SUNY Binghamton, New York, USA , 2013 IEEE Sixth International Conference on Cloud Computing.

[13] Z. Fadika, M. Govindaraju, R. Canon, and L. Ramakrishnan. Evaluating *Hadoop for Data-Intensive Scienti_c Operations. In Cloud Computing (CLOUD)*, 2012 IEEE 5th International Conference on, pages 67-74. IEEE, 2012.

[14] Z. Ye and S. Li, "*A request skew aware heterogeneous distributed storage system based on Cassandra*," in Proceedings of the International Conference on Computer and Management (CAMAN '11), pp. 1-5, May 2011.

[15] G. Wang and J. Tang, "*The NoSQL principles and basic application of cassandra mode*l," in Proceedings of the International Conference on Computer Science and Service System (CSSS '12), pp. 1332-1335, August 2012

[16] B. G. Tudorica and C. Bucur, "*A comparison between several NoSQL databases with comments and notes*," in Proceedings of the 10th RoEduNet International Conference on Networking in Education and Research (RoEduNet '11), pp. 1-5, June 2011.

[17] Y. Li and S. Manoharan, "*A performance comparison of SQL and NoSQL databases*," in Proceedings of the 14th IEEE Paci_c Rim Conference on Communications, Computers, and Signal Processing (PACRIM '13), pp. 15-19, August 2013.

[18] A. Silberstein, B. F. Cooper, U. Srivastava, E. Vee, R. Yerneni, and R. Ramakrishnan. *E_cient bulk insertion into a distributed ordered table*. In Proceedingsof the 2008 ACM SIGMOD international conference on Management of data, pages 765778. ACM, 2008.

[19] R. Sumbaly, J. Kreps, L. Gao, A. Feinberg, C. Soman, and S. Shah. *Serving large-scale batch computed data with project voldemort*. In Proceedings of the 10th USENIX conference on File and Storage Technologies, pages 1818. USENIX Association, 2012.

[20] J. Sun and Q. Jin. *Scalable rdf store based on hbase and mapreduce*. In Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on, volume 1, pages V1633 V1636, aug. 2010.

[21] R. Taylor. *An overview of the hadoop/mapreduce/hbase framework and its current applications in bioinformatics*. BMC bioinformatics, 11(Suppl12):S1, 2010.

[22] A. Thusoo, J. Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, S. Antony, H.Liu, and R. Murthy. *Hive-a petabyte scale data warehouse using hadoop*. In Data Engineering (ICDE), 2010 IEEE 26th International Conference on, pages 9961005. IEEE, 2010.