Performance Analysis of NoSQL Databases with Hadoop Integration

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science and Engineering

Submitted By
Ayush
(Roll No. 801232005)

Under the supervision of:
Dr. Seema Bawa
Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

July 2014

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "*Performance Analysis of NoSQL Databases with Hadoop Integration*", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Computer Science and Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. Seema Bawa* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

(Ayush)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

14/07/14

(Dr. Seema Bawa)
Professor,
Computer Science and Engineering Department,
Thapar University, Patiala

Countersigned by

(Dr. Deepak Garg)
Head
Computer Science and Engineering Department
Thapar University
Patiala

14/7

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

i

Acknowledgement

I would like to express my sincere and deep gratitude to my guide *Dr. Seema Bawa*, Professor in Computer Science & Engineering Department, for the guidance, support and encouragement. She provided me all resource and guidance throughout thesis work.

I am also thankful to *Dr. Deepak Garg*, Head of Computer Science & Engineering Department and *Dr. Ashutosh Mishra*, P.G. Coordinator, for the motivation and inspiration that inspired me for the thesis work.

I would also like to thank my parents and friends for their inspiration and ever encouraging moral support, which went a long way in successful completion of my thesis.

Above all, I would like to thank the almighty God for his blessings and for driving me with faith, hope and courage in the thinnest of the times.

Ayush

Abstract

According to the recent study by IBM in 2012 “the 90% of the internet data is produced in last 2 years and every day 2.5 exabytes of data is being created”. All of these was reason for the development of new databases called NoSQL. NoSQL databases have been designed to cater the requirements of handling the very large data also called “Big Data”. Unlike relational databases which are inefficient to process the big data the NoSQL databases have been designed to address the specific needs of current big data era like storing unstructured data, scalability and read/write efficiency. Relational databases have fixed schema due to which unstructured information cannot be stored in it and internet is exploding with such kind of information which comes from numerous sources. Since the data comes from variety of sources so there storage format cannot be controlled which has given rise to unstructured storage and to handle all these task through relational database has become inefficient. Along with the fulfilling storage requirements NoSQL databases have made their entry in real time analytics. Real time analytics is now being used in e-commerce, social networking websites to predict the customer needs and make business decisions accordingly.

In this thesis different types of NoSQL databases are explained and integration of NoSQL databases “MongoDB” and “Cassandra” with analytics tool “Hadoop” is done. Hadoop is not a database but it is a framework to handle large amount of data by distributing the data among different cluster nodes and applies parallel processing on them. We have evaluated both integrated technologies on different parameters like read and write, scalability and fault tolerance.

Table of Content

List of Figures

List of Tables

`

Chapter 1

Introduction

This chapter introduces NoSQL databases and their need in bigdata era, NoSQL databases types and their applicability to specific area of usage, Hadoop framework with processes included in it and organization of the thesis.

Due to the exponential growth in internet data it is now divided into four dimensions also called 4V model of bigdata [1]. The model is made considering the volume, velocity, variety and veracity of the data. As internet is flooding with the information from social networking, news feeds, scientific data and sensors data relational databases are unable to handle this vast amount of unstructured and semi-structured data. The information generated by the web is not in the structured format [2] and cannot be formed into the two dimensional structure which is followed by the relational databases so NoSQL [3] databases were designed to handle this ambiguity. The NoSQL databases are schema free unlike relational databases which have pre requirement to form the schema before storing the data in database instead NoSQL databases can store unstructured data making them appropriate tool for storage of big data. Various NoSQL databases are introduced in the market with each having its own specialty such as MongoDB [4], Cassandra [5], Redis [6] and Neo4j [7].

Another major problem with relational databases is that of its inefficiency to handle the scalability. Since relational databases concept came in when internet was in its initial stages so there was no issues for scaling the data that's why relational databases were not designed considering the need of scalability though scalability can be applied in relational databases but its performance is not as per the requirement of current scenario. Due to inefficiency of the relational databases "Facebook" had to use Memcache [8] to scale the data of user across the web. Both NoSQL databases MongoDB and Cassandra discussed in this paper fulfil the requirement of storing the unstructured information, scaling of the data and both databases are ranked top most databases in their respective databases types.

`

Now a days NoSQL databases are integrated with the analytics tool so that they can apply the analytics on the data with zero or minimal latency and further data can be used in making the managerial decision making. Instead for opting for the warehousing option where data is stored and later the analysis on that data is done, the companies are now opting for the real time analytics [22] so that they can know there customer well before the competitor companies.

Since the data is distributed all across the web so special distributed processing tool is required which can work in distributed environment and can also be integrated with the new NoSQL databases which are storing all the data in them so we have used Hadoop [9] as the tool and made its integration with the NoSQL databases because right now Hadoop is considered best fault tolerant tool and it is a parallel processing tool also which can run the map-reduce program in parallel and that too in inexpensive way because it does not require heavily configured clusters to work upon and clusters can be both homogeneous and heterogeneous configured.

1.1 Databases

A database stores the data in organized way such that it can retrieve the required data swiftly. Database is similar to the electronic file system in which you can perform all operation in quick and efficient way.

Earlier databases example can be of library catalogue which contains the three fields of reference number of book, book name and book location. The number of lines in the catalogue will be records and the collection of them all is called file. Database management systems are used to extract the information from the database. These database helps you to insert, delete and search for the data. There are different categories of databases available in the market depending on the applicability which runs on single to multi node system for example MySql and Oracle etc.

1.2 NoSQL Databases

A NoSql database models the data other than the tabular relations which is used in relational database. NoSql databases are often highly optimized for simple retrieval and

`

appending operations in different areas of application, whereas a relational database management system (RDBMS) is intended as a general purpose data store. Following are the reasons to use NoSQL databases.

Very Large data: The databases are becoming too large to fit into a single database table on a single machine.

Volume of Data was rapidly increasing: The number of web pages was increasing and for the tune up of systems the database had to scale up immediately to handle a petabytes of data.

Data on web is not consistent: Not having the proper defined fields

1.3 NoSQL Database Types

NoSQL databases are categorized in mainly four categories key-value pair, column oriented, document oriented and graph databases. Each type of database is used for specific purpose.

1.3.1 Key-Value Pair Databases

Data is stored in unstructured format consisting of a key and the values. Each key is unique and is used to retrieve the values associated with it. These databases can be visualized as relational databases having multiple rows and only two field's key and value. The value can be text, images and files etc. which are stored in schema less form making it unstructured data store. Key value databases are highly suitable for applications where data is continuously growing. Key value databases are considered best for the scalability but they lack in processing complex queries on the data [8]. For example: Redis [6] and Voldemort [19].

Its characteristics are:

i. Its schema is flexible and does not require to design schema at first and addition, deletion of fields is very convenient;
ii. It can range queries for key.
iii. High scalability: a single point of failure does not affect the whole cluster.

`

1	Name: ABC

(a)

2	Name: XYZ	EmailId: xyz@gmail.com

(b)

3	Name: XYZ	Last Name: UVW	EmailId:xyz@abc.com

(c)

Figure 1.1 Structure of data storage in Key-Value pair Database. In this Figure data storage of users in social networking site is shown. In part (a) only user name is stored (b) user name and email address is stored (c) user name, last name, EmailId is stored.

Figure1.1 shows that if same information had to be stored in the RDBMS tables then we had to define the number of columns during schema creation and if any column is left empty by user then still that space will be left unused (Null and empty cells in RDBMS also require space) leading to the wastage of memory but the in key value pair only space utilized is counted and no extra space is wasted even if user do not fill all its information.

1.3.2 Document Oriented Databases

Document oriented database is one of the most popular ways of storing data, in this each record and database is considered as a "document" and a "collection" respectively. In a document database, such as MongoDB, everything related to a database object is encapsulated together.

Table 1.1 Mapping of Relational and Document oriented databases

Relational Database	**Document-Oriented database**
Database	Database
Table	Collection
Record	Documents

Table 1.2 shows that both RDBMS and document oriented databases have concept of "database" but the "Table" of RDBMS is called "Collection" in document oriented database. The collection is not equivalent to tabular format as it is in RDBMS but it is

`

schema less means it can store “documents “in them. Usually single line in RDBMS is considered as record and it contain the all the information in fields. Documents are the records in document oriented database but instead of pre specified number of fields a documents can contain any number of fields and that information is also stored in Key-Value pair bases.

Storing data in this way has the following advantages:

i. Documents are independent units which helps in making performance much better and distribute data to nearest locations.
ii. Application logic is easier to write. You don’t have to translate between objects in your application and SQL queries, you can just turn the object model directly into a document.
iii. Unstructured data can be stored easily, since a document contains the keys and values application logic. In addition, cost of migration is less since the database does not need to know its information schema in advance. For example: MongoDB and CouchDB [20].

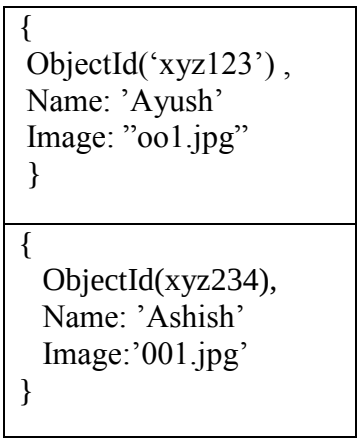


Figure 1.2 Structure of data storage in Document oriented database. In the Figure the structure of information storage in MongoDB is shown.

As shown in Figure1.2 MongoDB stores the data in JSON [21] (JavaScript Object Notation) format which is open, human and machine-readable standard that facilitates data interchange, and along with XML is the main format for data interchange used on the web. JSON format is used by twitter and facebook to store data.

`

1.3.3 Graph Oriented Databases

It contains nodes and edges relating nodes to represent relationship among them. Nodes in the graph also give information contained in them. Similarly, edges in the graph may also have their own properties i.e. directed or undirected. Relationships are identified by their names and can be traversed in both the directions. Comparing with Entity-Relational Model, a node corresponds to an entity, property of a node to an attribute and relationship between entities to relationship between nodes. In tabular databases if we want data from more than two tables then join operation is required. Join slows down the operation of database because it degrades the performance while joining two tables and also results in increased table size. Which leads to easier adaptation to schema evolution and ability to capture ad-hoc relationships. Social networking relational database stores data in structured format or we can say that in row and column fashion because it is predefined and have fixed schema, it is good for some applications which require complex queries which involves join, nested queries like bank application, transaction. But when we talk about dynamic data like weather report generation, sensor report where data are generated in tera or peta byte and change frequently, for such kind of application relational database has proved fail. We are in need of completely different database which also support basic features of relational database like sorting and searching and also have additional features which are not included in RDBMS. Graph databases are used where relation between users is as important as data [23]. For example: Neo4j.

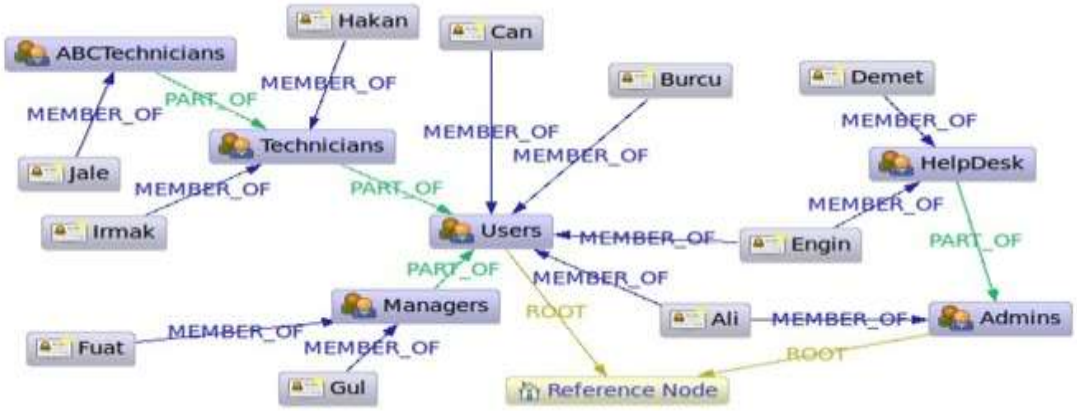


Figure 1.3 Graph database storage scenario [24]

`

The Figure1.3 contains the nodes which contains the information about the user and edges shows the friendship relation of one node with other.

1.3.4 Column Oriented Databases

It is developed to solve the problem of three areas which are data having many number of properties, sparse type of data and rapid changes in schema of the database required. Similar to relational databases where rows are the main storage unit, in column-oriented databases column are main storage unit. This change in storage design results in better performance for scaling of the data. These column oriented databases is also being used in the field of analytics because analytics is applied to very few specific columns and also benefitted because the data type of column and value range is fixed so analytics can be applied to specific type of data. In case of row oriented databases multiple column has to be handled which can be of different data type so analytics becomes difficult. Most of the column oriented databases are also compatible with map-reduce framework, which speeds up processing of large amount of data by distributing the problem on large number of systems. For Example Cassandra.

Figure 1.4 Conversion of tabular data into row oriented and column oriented data format [25]

`

In Figure1.4 transformation from tabular format to row oriented and column oriented database is done. In column oriented databases transposition is done so that whenever whole column data is required for scalability, analytical purpose that particular column can be transferred. Row oriented are more beneficial when multiple columns of row are required at same time.

1.4 Hadoop Framework

The Hadoop is a framework that allows for the distributed and parallel processing of large datasets across computer clusters. It is developed to scale up from single machines to hundreds of machines, which are offering computation and storage of the data. Rather than relying on the separate hardware to deliver high availability, the library itself is designed to detect fault tolerance so delivering a highly available service which may be prone to failures can be possible.

Data intensive states about the bigdata and distributed applications are the applications that works on network which require communication and coordination with each other by passing messages and parallel processing means distributing and processing of the data between the computer cluster.

Hence Hadoop works on a distributed and parallel environment because of map-reduce and is developed to store, handle and process large amount of data set (in petabytes, exabytes and more). Hadoop stores bigdata, but it is not a database. It's a framework that handles large amount of data for processing. Hadoop was developed from the research work by google that is Google file system and google's map-reduce.

1.4.1 Processes in Hadoop .

The system on which apache Hadoop runs means running a set of daemons or resident programs. These process have specific roles assigned to them which can run on one or more node in the system. The processes are:

NameNode: It act as the master of all nodes in the system it directs the slaves' nodes or datanodes to perform the processing of data. It is the index keeper of all the chunks which are allocated to the datanodes. Usually in large clusters the server hosting the

`

namenode does not perform any processing and are just kept to direct all the other datanodes. There is a negative aspect to the importance of namenode because if one namenode fails then the whole system will be halted and cannot proceed because there can be only one namenode in the system.

DataNode: Most of the machine in a cluster will have the datanode process running in them and the data chunks from the HDFS is assigned to them for computations. When you want to perform the computation the file from HDFS is broken into different blocks and namenode will do the allocation of data chunks. Datanodes will again further replicate the chunks to other datanodes so that recovery can be done in case of any failure.

Namenode keep communicating with the datanodes and status of its processing. After the mapping is complete, the datanodes continuously poll the namenode to provide information regarding local changes as well as receive instructions to create, move, or delete from the local disk. Figure1.5 shows replication policy.

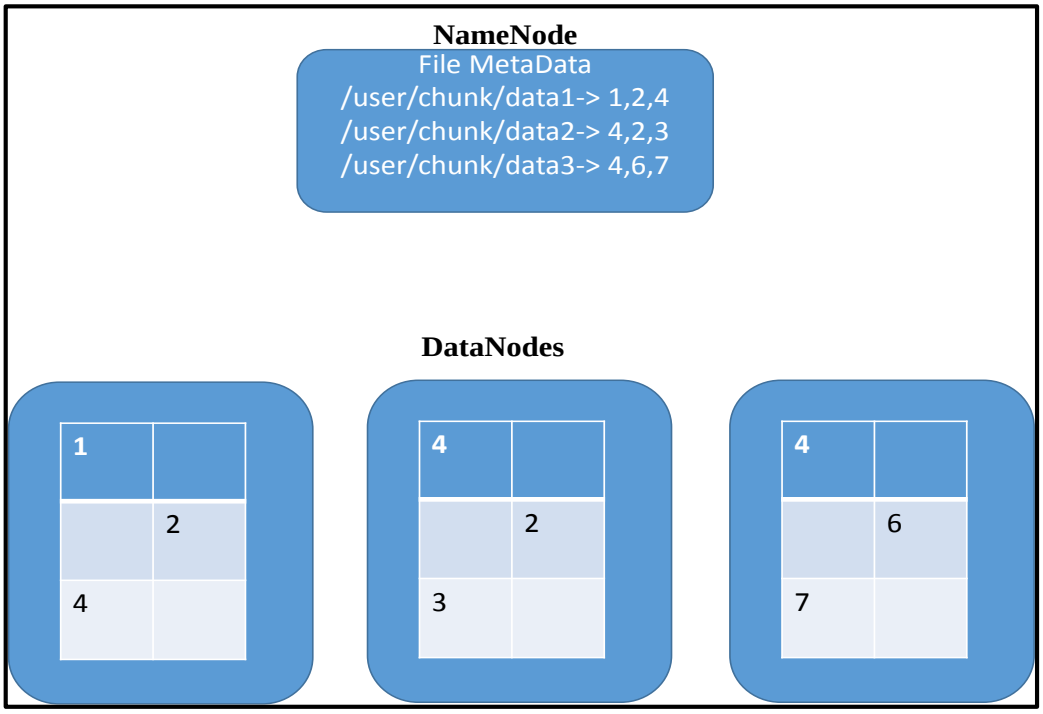


Figure 1.5 Namenode and Datanode containing data

Secondary NameNode: Secondary namenode also contains a namespace image and edit logs like namenode. Secondary namenode after certain interval of time

`

copies the image from namenode and combine this namespace image with the edit log and copy it back to the secondary namenode so that it will have the fresh copy of namespace image and edit log. Now suppose after certain time the namenode goes down then we can restart some other machine with the namespace image and the edit log that's why we have with the secondary namenode and hence can be prevented from a total failure.

Secondary namenode consume same resources as compared to the namenode in the Hadoop system so usually in large clusters they are kept in the separate nodes. But if the cluster are small and there are little chances of failure they are not started in the system in order to save the overhead which will be beard by the system to handle the secondary node. The process in Figure1.6 and it consists of two parts.

i. **EditLog**: For continuously recording every change that occurs to file system metadata. It creates a new file in HDFS for the namenode to insert a record into the EditLog.
ii. **FsImage:** To store entire file system namespace, including the mapping of blocks to files and file system properties. The FsImage is stored as a file in the namenode's local file system.

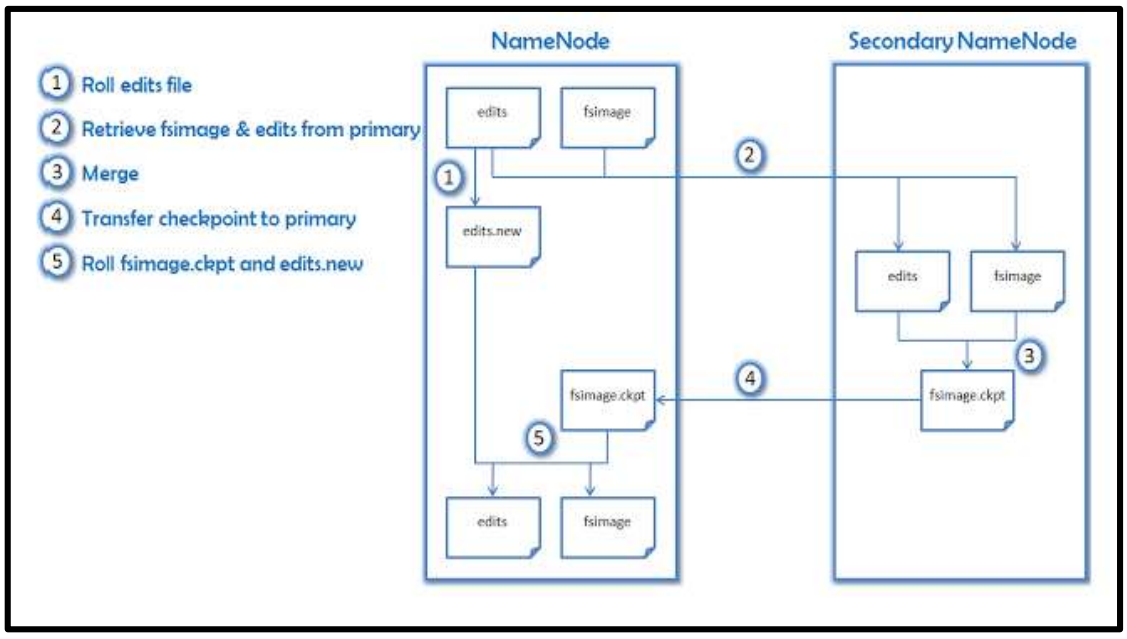


Figure 1.6 Secondary NameNode [27]

`

JobTracker: when you submit your program of map-reduce to cluster nodes then it will determines the scheduling plan execution by deciding files to process, assigns chunks to different nodes and check all tasks as they're running. If a task fail, then it will automatically restart the task from different node on which the chunk is kept. There will only be one jobtracker in the cluster as shown in Figure1.7.

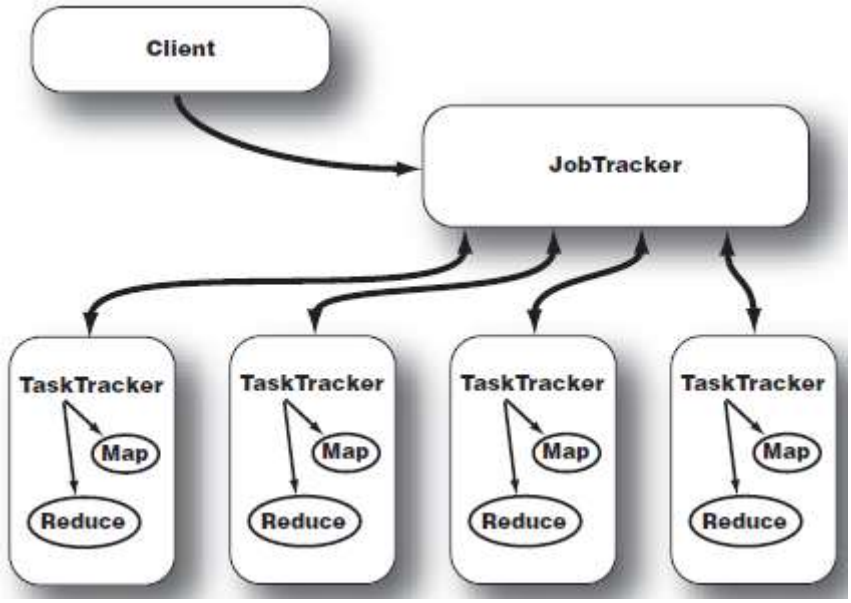


Figure 1.7 JobTracker and TaskTracker [26]

TaskTracker: Jobtracker and tasktracker follow a master/slave architecture. The jobtracker is the master which will see map-reduce job and the tasktracker will monitor the execution of each tasks on every datanode. Every tasktracker is held responsible for executing the each tasks that jobtracker assigns to them. Although there is a single tasktracker per datanode node, each tasktracker can have multiple map and reduce task in parallel [27].

1.5 Thesis Objective

Following are the objectives of the thesis:

i. To study the various NoSQL databases and its types and identify their particular area of usage.
ii. To study the hadoop framework and the daemons in it which runs on Hadoop cluster nodes.

`

iii. To carry out detailed and comparative study on column oriented and document oriented databases and there benefit over row oriented databases.

iv. To integrate the selected databases from column oriented and document oriented database with Hadoop.

v. To analyse both combination by evaluating them on different parameters.

1.6 Structure of the thesis

The rest of the thesis is organized as follows:

Chapter 2:– This chapter gives the detailed study on map-reduce, Hadoop distributed file system, comparison of selected NoSQL databases from document oriented and column oriented databases.

Chapter 3:– It covers problem statement and the need to do analysis.

Chapter 4:– This chapter explains the experiment step-up, implementation and the analysis done on the results gathered.

Chapter 5:– This chapter gives conclusion of thesis and future work possible in this area of research.

`

Chapter 2

Literature Survey

This chapter explains the concept of map-reduce, HDFS, comparison between MongoDB and Cassandra and there benefits over row oriented databases.

2.1 Map Reduce

Map-reduce is a data processing tool which performs the parallel processing. It is mainly used because of its capability of easily scaling data over the network having multiple nodes. It consist of two main phases, called mapper and reducer. In mapper phase it will take the input from the Hadoop distributed file system (HDFS) and distribute the task among the multiple nodes by generating different maps. In reducer phase data from all the maps is collected after the distribution and data is stored back in HDFS after the processing. Dory et al. [12] had compared the elastic scalability of Cassandra, MongoDB and HBase but without the mapreduce consideration. Figure2.1 and Figure2.2 shows the mapping task with single and multiple reducers respectively.

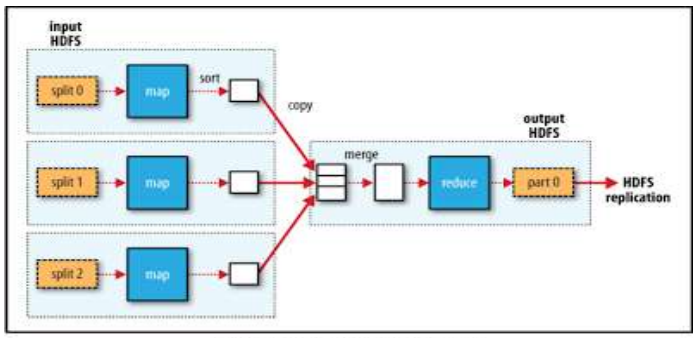


Figure 2.1 Map-reduce with single reduce task [28]

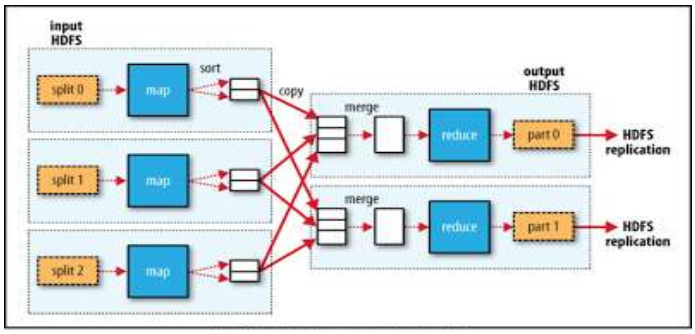


Figure 2.2 Map-reduce with multiple reduce task [29]

`

2.2 Hadoop Distributed File System

Hadoop Distributed File System (HDFS) was developed to store very large datasets with streaming data access which will be running on large computer clusters with economic hardware. Dede et al. [13] evaluates the HDFS and Cassandra by considering the different parameters of Cassandra partitioning, scalability, replication factor and data locality

Large datasets: The size of file which is going to be processed having the size of petabytes, exabytes and above.

Streaming data access: Hadoop distributed file system was built around the concept that data is written once and read many times and this is the same procedure which is followed in facebook because facebook users writes post once but it is read by many peoples.

Economic hardware: Hadoop distributed file system does not require use of heavy servers for processing but it can run on normal computer clusters which can be either heterogeneous or homogeneous.

Hardware Failure: Hadoop distributed file system may consist of thousands of nodes each processing a file chuck allocated to it. Special process called namenode is deployed which tracks the failure of each node and relocate its remaining task to other node. The reallocation of the nodes is decided automatically by the namenode which act as the master node of the system.

Simple coherency model: It follows the principle of write once and read many so when file is written it is not updated so no coherency issues occurs because once written something is not edited in future.

Process data near request: Instead of moving the data to other location where it can be processed it will be processed at the nearest location where it is located therefore reducing the cost of movement of the data.

Multiple Environment: It can work on both heterogeneous and homogeneous environment.

`

2.2.1 Difference between Hadoop distributed file System and Normal file system

Table 2.1 Difference between HDFS and NFS

Hadoop distributed file System	Normal File system
Block Size: Block size determines the minimum amount of data that is written and read from a disk .In case of HDFS it is 64MB and it grows in its multiples like 128MB , 256MB, 512 MB	**Block Size:** Its size is very small as related to HDFS which is normally 512 bytes for many systems
Storage System: No technique is followed to store the data in HDFS but all the data is dumped in the Namenode (Generally called master node) of the system.	**Storage System**: Data is stored hierarchically i.e. There is a folder XYZ**,** inside that folder there is again one another folder UVW**,** and inside that there is abc.zip file.

2.2.2 Read operation in HDFS

Hadoop distributed file system has a master and slave architecture. Namenode and datanode work as master and slave respectively. All the metadata information is with namenode and the original data is stored on the datanodes. The Figure2.3 shows how data flow happens between the Client interacting with HDFS, i.e. the namenode and the datanodes.

These are steps involved in reading the file from Hadoop distributed file system. Let's take example of a client who wants to read a file from HDFS. So following are the steps involved in the procedure:

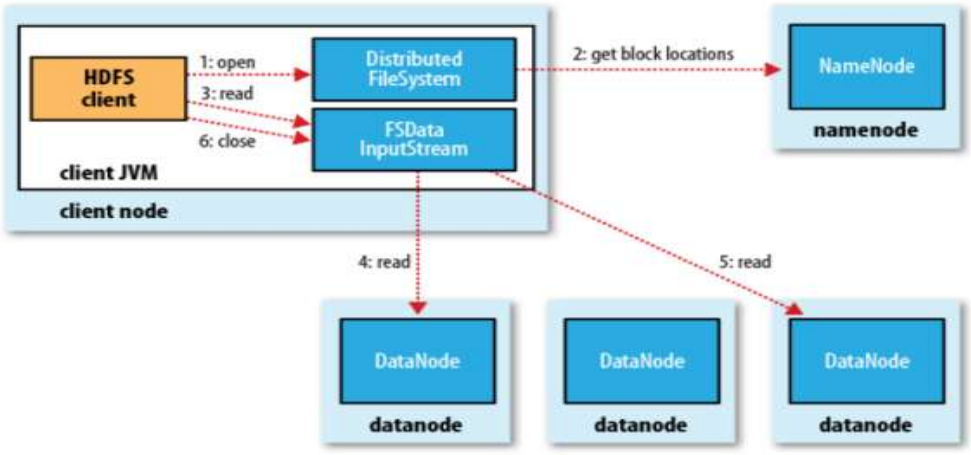


Figure 2.3 Read operation in HDFS [27]

`

Step 1: First the call will be made to open the file to file system object, which for HDFS is an instance of distributed file system class.

Step 2: DFS will call the namenode to determine the locations of the blocks for the first few blocks of the file. The namenode returns the address of object of FSDInputStream (an input stream that supports data file seeks) to the client for it to read data.

Step 3: The client then calls read on the stream, which has stored the closest datanode addresses for the first few blocks in the file and then connect to that datanode.

Step 4: Data is streamed back to client from the datanode, which calls read repeatedly on the stream.

Step 5: If the end block is reached, it will call the close to connection of the datanode and then find the nearest datanode for the next block.

Step 6: It will now call the namenode to retrieve the datanode locations for the next batch of blocks as needed. When the client has finished reading, it calls close on the FSDataInputStream.

2.3 NoSQL Databases

The working principle of NoSQL databases was introduced through CAP theorem by Eric Brewer in 2000 but due to its deficiency BASE theorem was derived from it and later on CAP was redefined in 2012 [18]. NoSQL databases are highly optimized for simple retrieval and appending operations because they do not follow the ACID properties. Cattell et al. [3] had examined the various RDBMS and NoSQL theoretically. Padhy et al. [14] compare the data model and architecture of NoSQL technologies.

2.3.1 MongoDB

MongoDB is an open source NoSQL document oriented database system developed by MongoDB Inc. As already discussed in document oriented databases Figure2.5 shows that data is stored in documents and not in tabular format despite all this MongoDB provides many features of relational databases which include indexing, replication, queries and provides much more improvement in load balancing and file storage.

`

MongoDB also provides its own map-reduce features for the distributed processing of the data.

Although MongoDB provides its own map-reduce but it has limitation as compared to the apache Hadoop. To utilize the MongoDB map-reduce the program has to written in JavaScript and since not so many data processing libraries are written in JavaScript so its scope for data processing is limited but Hadoop provides the feature to write the programs in python or java which are already known for their large number of libraries and "Hadoop Streaming" can be used to implement the map-reduce in other languages. Apache Hadoop is specially build for fully distributed and multi-threaded execution of data processing, so it performs much better than MongoDB. As MongoDB is providing the data storage and processing both so map-reduce will be added to its work load and eventually results in lower performance but if we use Hadoop for processing data the load will be taken by the Hadoop and NoSQL can do the storage part in parallel without degrading its performance. Along with this Hadoop provide the integration with other technologies like Pig, Hive, Zookeeper and many more which can be add on advantage for processing of the data. MongoDB is used by many companies such as EBay, New York times and Foursquare. Govindaraju et al. [11] evaluates HDFS and MongoDB but they had done experiments on Cray XE6 machine having 153,216 "cores" which is used for the scientific purpose only and cannot be used in commercial activities by the regular companies and for the scalability purpose they had used the "cores" of computer but it cannot work in companies because they do not have the machine which have very large number of "cores" of this range. Figure2.4 shows the workflow of the map-reduce operation on MongoDB and Cassandra using Hadoop.

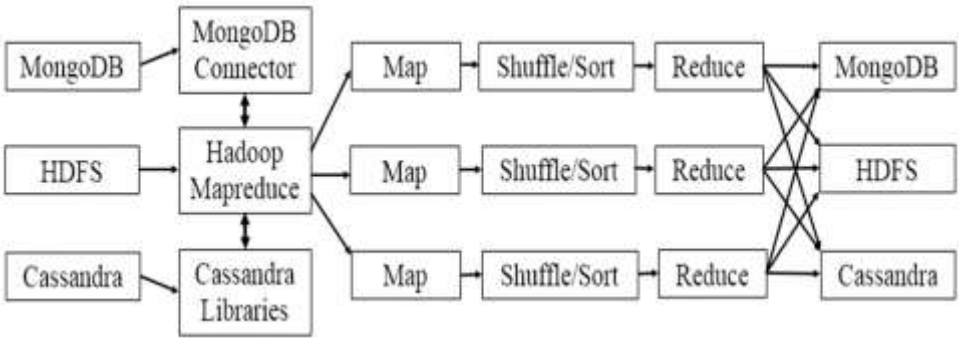


Figure 2.4 High level architecture for MongoDB-Hadoop and Cassandra-Hadoop integration.

`

The Figure2.4 shows that the data is collected from MongoDB, HDFS and Cassandra and after applying the various phases of map-reduce operation the data is stored back either to MongoDB, HDFS or Cassandra.

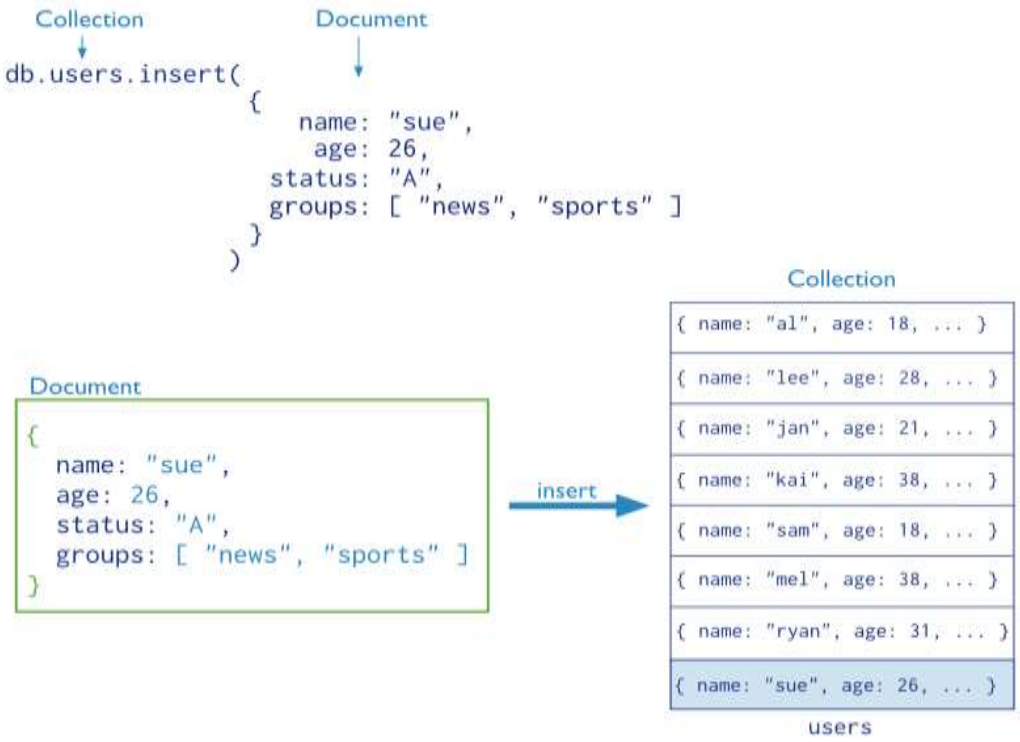


Figure 2.5 MongoDB Storage format [30]

2.3.2 Cassandra

Cassandra is an open source column oriented database system developed by apache software foundation. Unlike MongoDB which follows the master-slave architecture the Cassandra works on peer-to-peer architecture which overcomes the limitation of massive scalability and high availability. Each node in the cluster is assigned a specific role and if any node fails the data is automatically distributed across all nodes with no downtime without wasting the time in calculations and assign data to each node. To detect the failed nodes gossip protocol is used and for cluster information it uses the concept of "seeds" which contains the IP address of all the nodes where information is stored. For partitioning purpose it uses two strategies "RandomPartitioner" and "ByteorderPartitioner". In "RandomPartitioner" data is distributed evenly across number of nodes and in "ByteorderPartitioner" partitioning can be done according to

`

range specified. Cassandra do not have its own map-reduce but its integration with Hadoop is provided by the DataStax enterprise. Cassandra is used by many companies such as Facebook, Netflix, and Twitter. Abramova et al. [10] do comparison between the MongoDB and Cassandra without the Hadoop attachment but they had done all the experiment on single virtual machine without considering the scalability parameter which was the main reason for development of NoSQL databases.

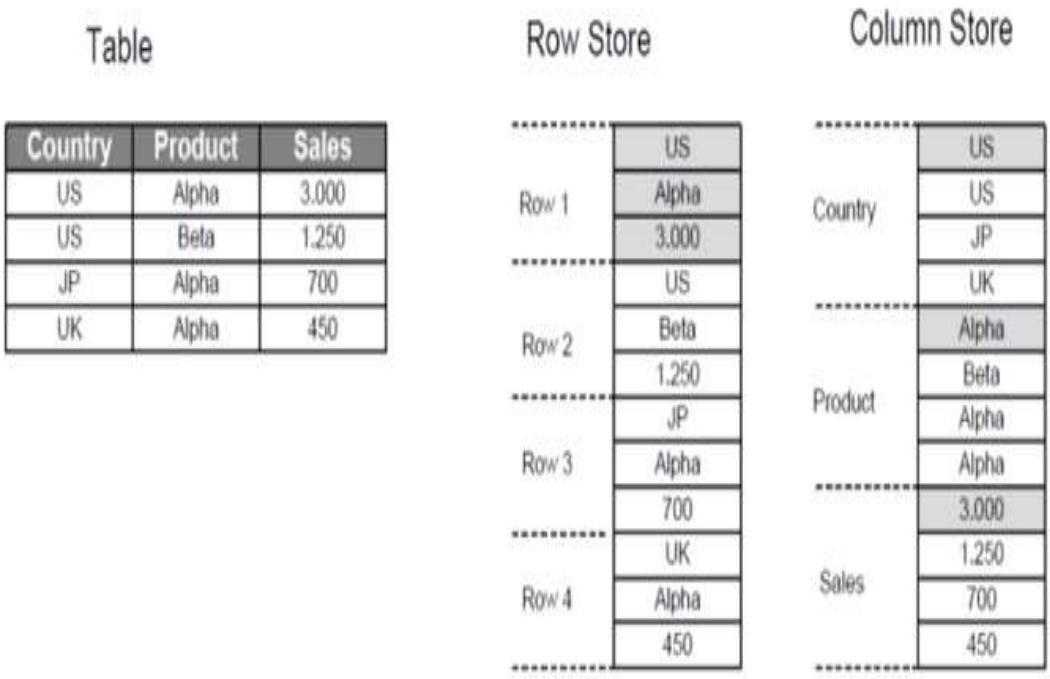


Figure 2.6 Cassandra Storage Format [31]

Apart from MongoDB and Cassandra there are only few other NoSQL technologies which are designed to utilize the map-reduce framework of Hadoop such as HBase [15]. HBase which is developed by apache is also a column oriented database like Cassandra and it is shown that architecture changes of Hadoop and HBase can improve the storage and processing capabilities [16]. But when Cassandra and HBase is compared the Cassandra over powers the capabilities of HBase [17]. We have chosen both NoSQL technologies depending on market applicability, usability areas and rank among their respective database types. These studies do not directly represent the performance measurement between Hadoop-MongoDB and Hadoop-Cassandra. Rapid increase in use of these technologies in real time analytics led us to evaluate their performance.

`

2.3.3 Comparison between HBase, MongoDB and Cassandra

Table 2.2 Comparison of HBase, MongoDB and Cassandra

Parameters	HBase	MongoDB	Cassandra
Database Type	Column Oriented Data store	Document Oriented Data Store	Column Oriented Data store
Development Language	Java	C++	Java
License	Open Source	Open source	Open source
Works on Operating system	Linux, Unix, Windows	Linux, OS X, Solaris, Windows	BSD, Linux, OS X, Windows
Database schema Used	Schema-less	Schema -less	Schema -less
Predefined Data types	Yes	Yes	Yes
Secondary Indexes	No	Yes	Restricted
Structured Query language	No	No	No
CAP theorem	Consistency, Availability	Consistency, Portioning	Availability, Partition Tolerance
Application programming Interface	Java API REST HTTP API Thrift	JSON Protocol	Proprietary protocol
Consistency	Immediate	Immediate and Eventual	Immediate and Eventual
Mainly Used for	Read	Read	Write
MapReduce	Through Use of Hadoop	Yes	Through Use of Hadoop
Single Field Indexes	Yes	Yes	Yes
Compound Indexes	No	Yes	No
Multi key Indexes	No	Yes	No

`

Parameters	HBase	MongoDB	Cassandra
Text Indexes	No	Yes	No
Hashed Index	No	Yes	No
Partitioning	Dynamic	Sharding	Sharding
Rebalancing of Nodes in Failure	Automatic	Automatic	Automatic
Compression of Data	Yes	Yes	Yes
Languages Used for programming	Java, Python	JavaScript	Java, Python
Triggers	No	No	No
Replication or redundancy	Through replication factor	Master Slave	All nodes are equivalent
Foreign Keys	No	No	No
JOIN Concept	No	No	No
Transaction	No	No	No
Concurrency	Yes	Yes	Yes
Durability	Yes	Yes	Yes

2.3.4 Benefit of Column Oriented Databases over Row Oriented Databases

i. Column oriented database are efficient than row oriented database when computation is to be done on multiple rows but it is constrained to small number of columns because reading of all those columns will require less time than reading all other data.

ii. They are efficient when data is to be inserted into single column of all rows in the database

iii. Column Oriented database are of same type so there is advantage to compress the data which cannot be done in row oriented database.

2.3.5 Benefit of Document Oriented Databases over Row Oriented Databases

i. Document oriented database is schema-less so dynamic entities like user customizable entities with optional fields can be done which is not possible in relational database.

`

ii. Document oriented database is schema-less so dynamic entities like user customizable entities with optional fields can be done which is not possible in relational database.

iii. Document oriented database do not create the view model from scratch on every request. This leads to reduced computation, reduced number of remote calls and improved overall performance.

iv. Objects can be stored as documents in document oriented database.

v. Entire object is read and written. There is no need to perform a series of insert statements.

vi. Since documents are independent it improves performance and decreases concurrency side effects.

`

Chapter 3

Problem Statement

The apache's Hadoop map-reduce framework is considered best in the field to process the big data while many companies have introduced the map-reduce technology but the apache's Hadoop performance is much ahead among them whether there is an issue for computational time, fault tolerance in a very large network, working on the cluster which are either heterogeneous (computers with different configurations) or homogeneous (computer with same configuration) and batch processing of data.

But there are still drawbacks of Hadoop because HDFS system is not able to handle and store the real time data and this is the reason Hadoop was not being used in areas except analysis and scientific research. Through the use NoSQL it is now possible to store the real time data and it is now being used extensively in social networking sites to store the images, user profile data, tweets of twitter website and later on analysis is done by Hadoop on that data by extracting the data directly from the NoSQL database and in last computation results will be stored back to the NoSQL database.

The aim of this thesis is to analyse the performance of MongoDB and Cassandra by integrating them with Hadoop to evaluate them on different parameters and analyse the results produced.

`

Chapter 4

Implementation Details and Testing

This chapter gives detail about network setup, implementation of various parameters for analysis and the experiment results gathered.

4.1 Experiment Setup of Cluster

We have configured the Hadoop-MongoDB (integration of Hadoop and MongoDB) and Hadoop-Cassandra (integration of Hadoop and Cassandra) on eight node cluster. Each node has Intel 3.40 GHZ i7 as the processor, 4GB of RAM and 64bit Ubuntu 12.10 operating system.

Hadoop Setup: we have used the Hadoop 1.1.2 version and default settings of mapper and reducer is used. Hadoop is started by typing "start-all.sh" command and Figure4.1 shows the starting procedure of Hadoop in which datanode and tasktracker will be started on all eight nodes while namenode, secondary namenode and jobtracker will be started only on master node. Figure4.2 shows the five running processes after Hadoop is started along with their processes id's on master node.

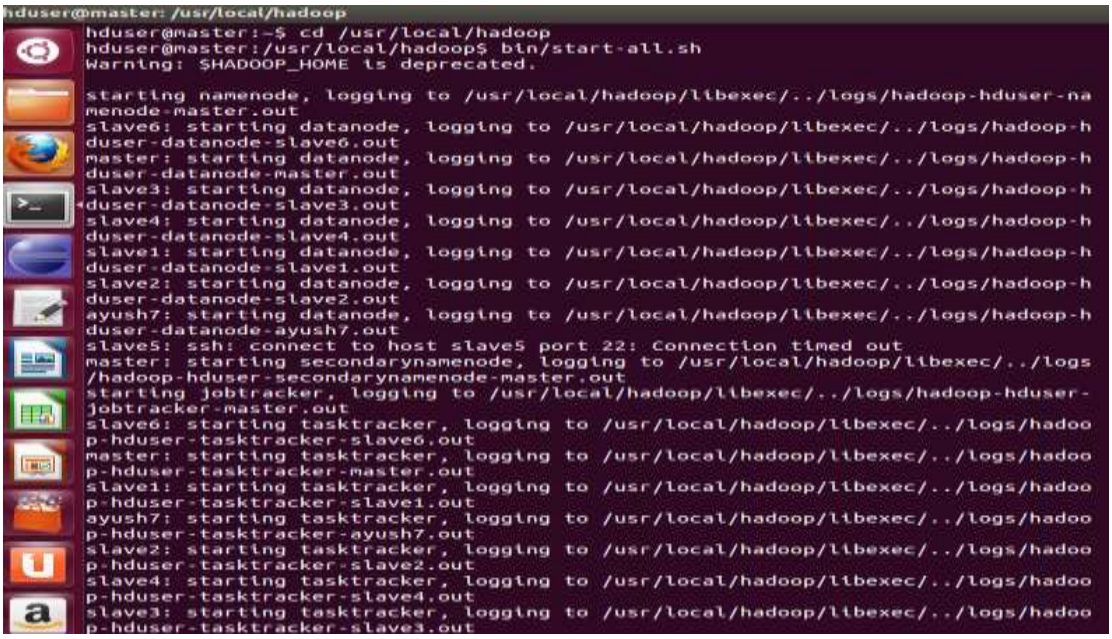


Figure 4.1 Screen shot of starting Hadoop on 8 (eight) node cluster

`

In Figure4.3 only two running processes datanode and tasktracker with their process id on slave nodes is shown. Figure4.4 shows the names of all the eight nodes and their tasktracker id's.

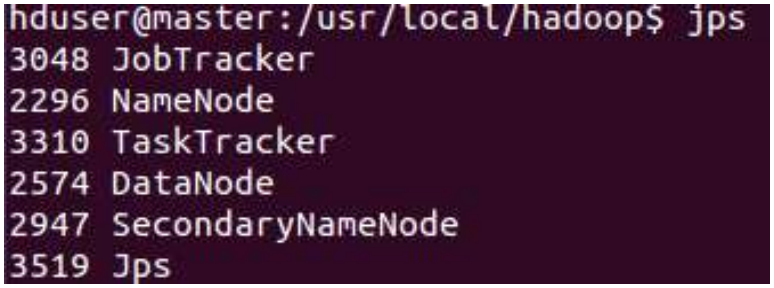


Figure 4.2 Screen shot of running daemons on master node

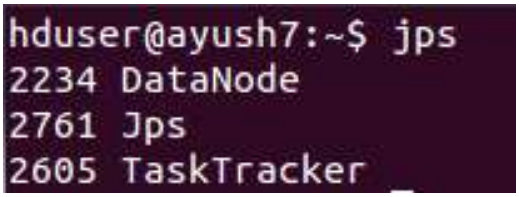


Figure 4.3 Screen shot of running daemons on slave nodes

Name	Host
tracker_master:localhost/127.0.0.1:33180	master
tracker_slave2:localhost/127.0.0.1:59907	slave2
tracker_slave5:localhost/127.0.0.1:60775	slave5
tracker_ayush7:localhost/127.0.0.1:42469	ayush7
tracker_slave3:localhost/127.0.0.1:58808	slave3
tracker_slave6:localhost/127.0.0.1:55223	slave6
tracker_slave4:localhost/127.0.0.1:58959	slave4
tracker_slave1:localhost/127.0.0.1:56563	slave1

Figure 4.4 Screen shot of list of all 8 (eight) nodes with their name and tasktracker port number

MongoDB Setup: we have used the MongoDB 2.2.3. To make the connection of

`

Hadoop and MongoDB connector is used so that we can run the program through eclipse integrated development environment (IDE) and Hadoop terminal.

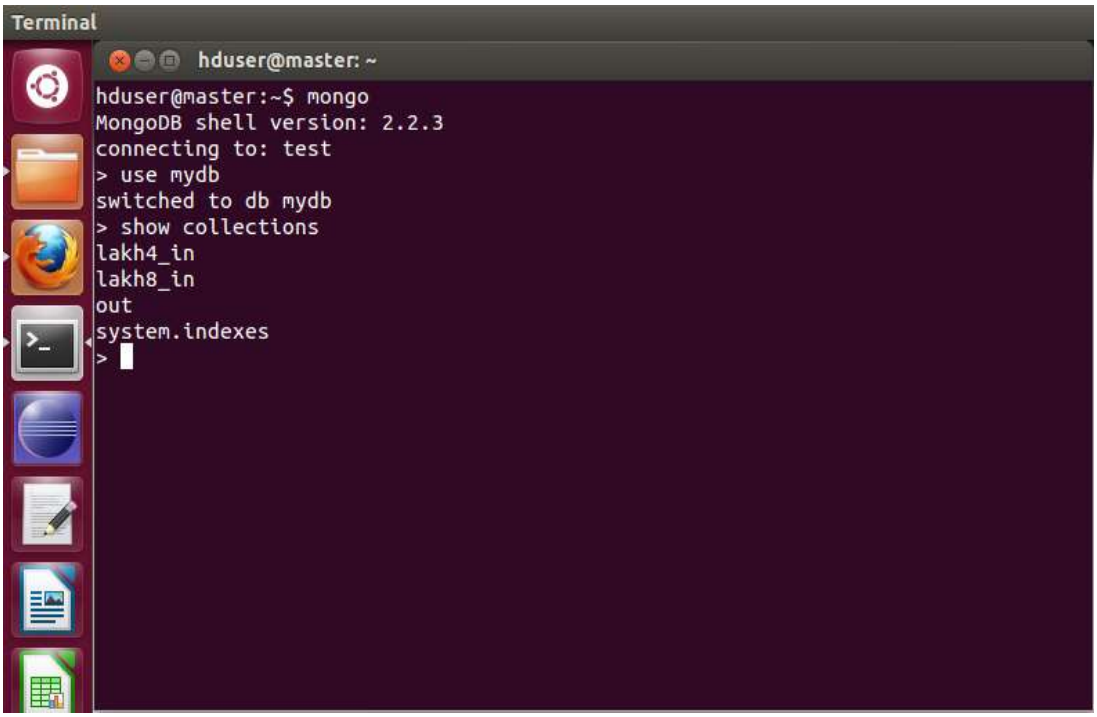


Figure 4.5 Screen shot of starting of MongoDB

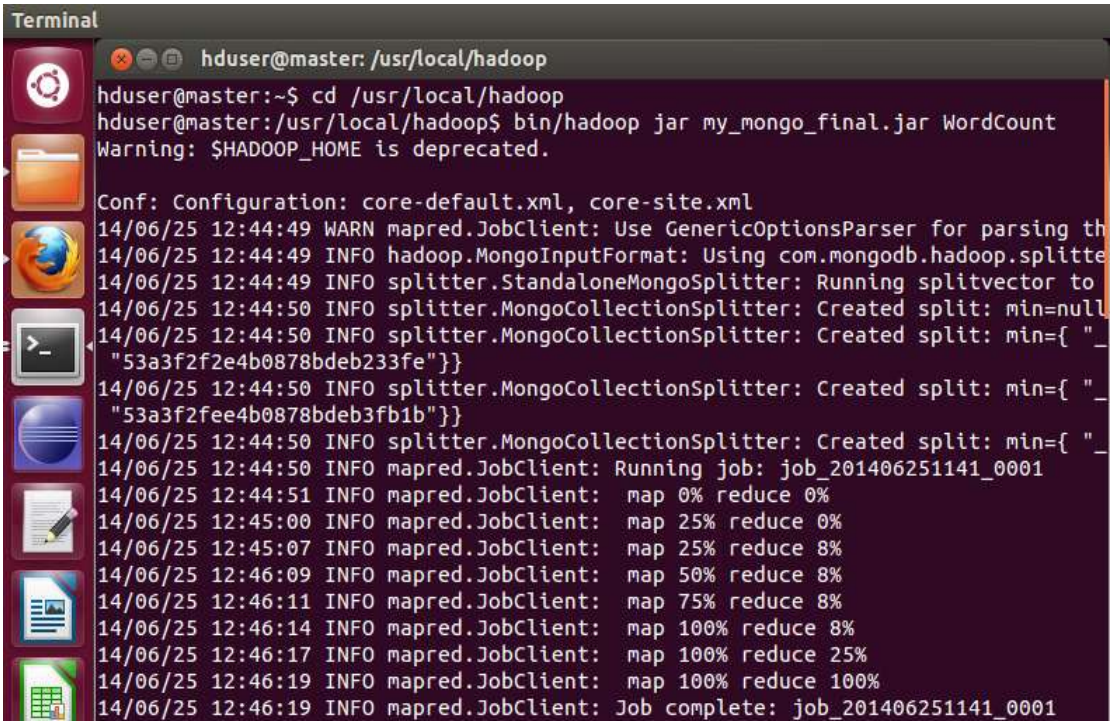


Figure 4.6 Screen shot of execution of Hadoop-MongoDB job

`

The MongoDB is started by typing the command “mongo” in the terminal. Figure4.5 shows the MongoDB terminal with all the collections in its “mydb” database. Figure4.6 shows the execution of Hadoop-MongoDB job.

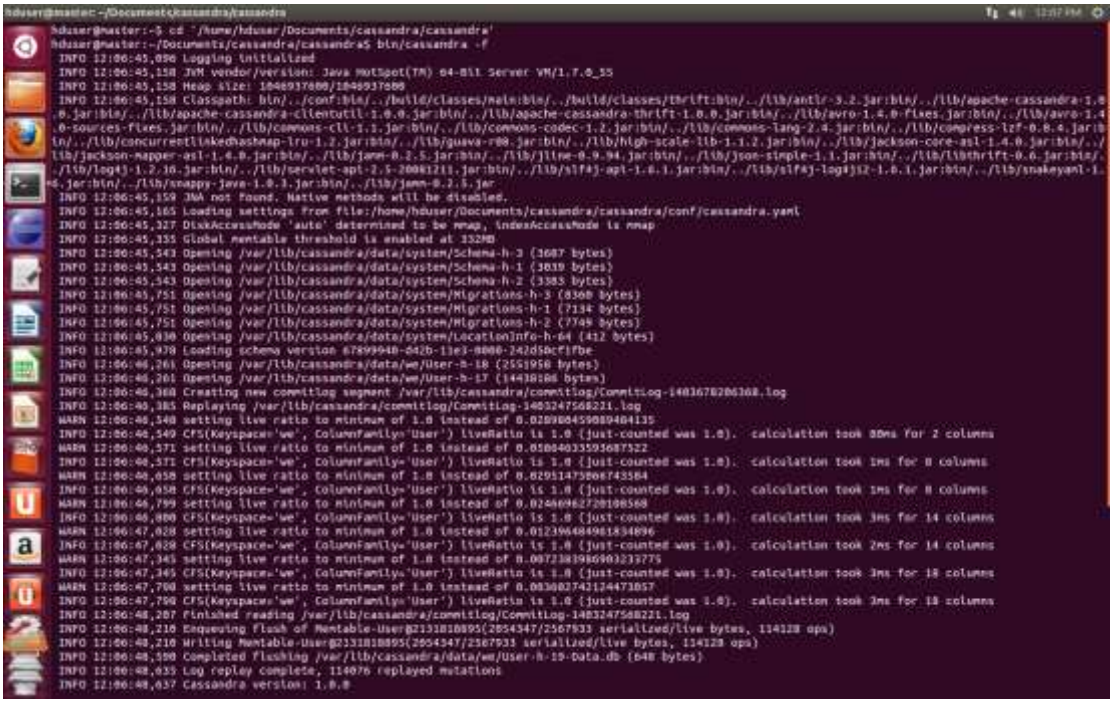


Figure 4.7 Screen shot of starting of Cassandra

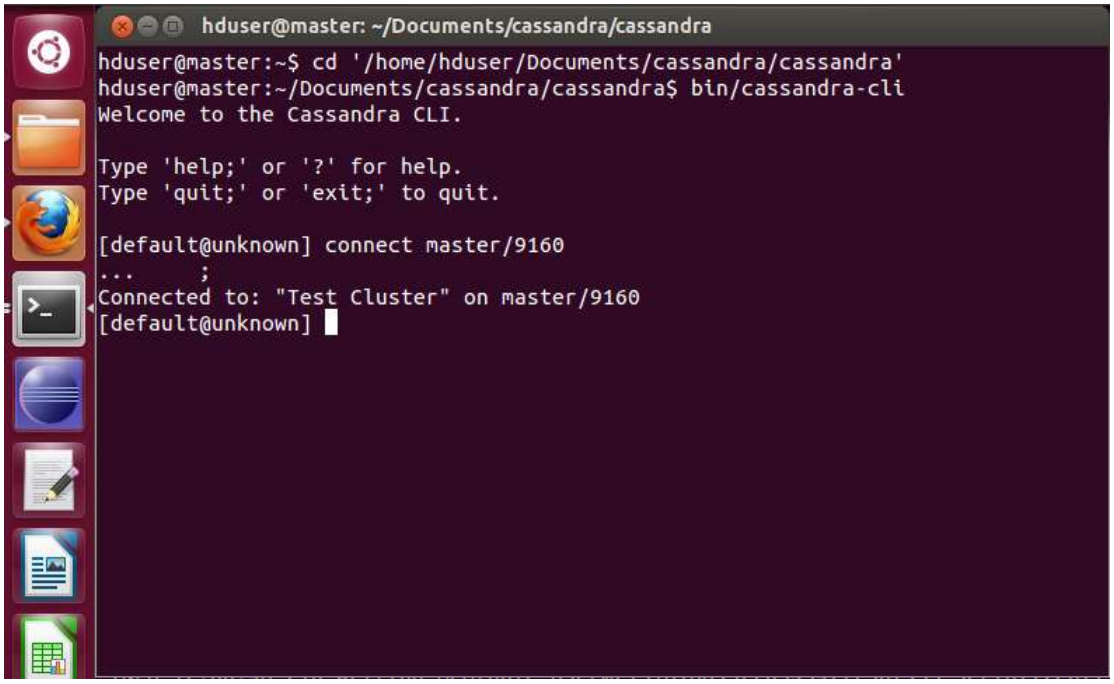


Figure 4.8 Screen shot of starting of Cassandra terminal

`

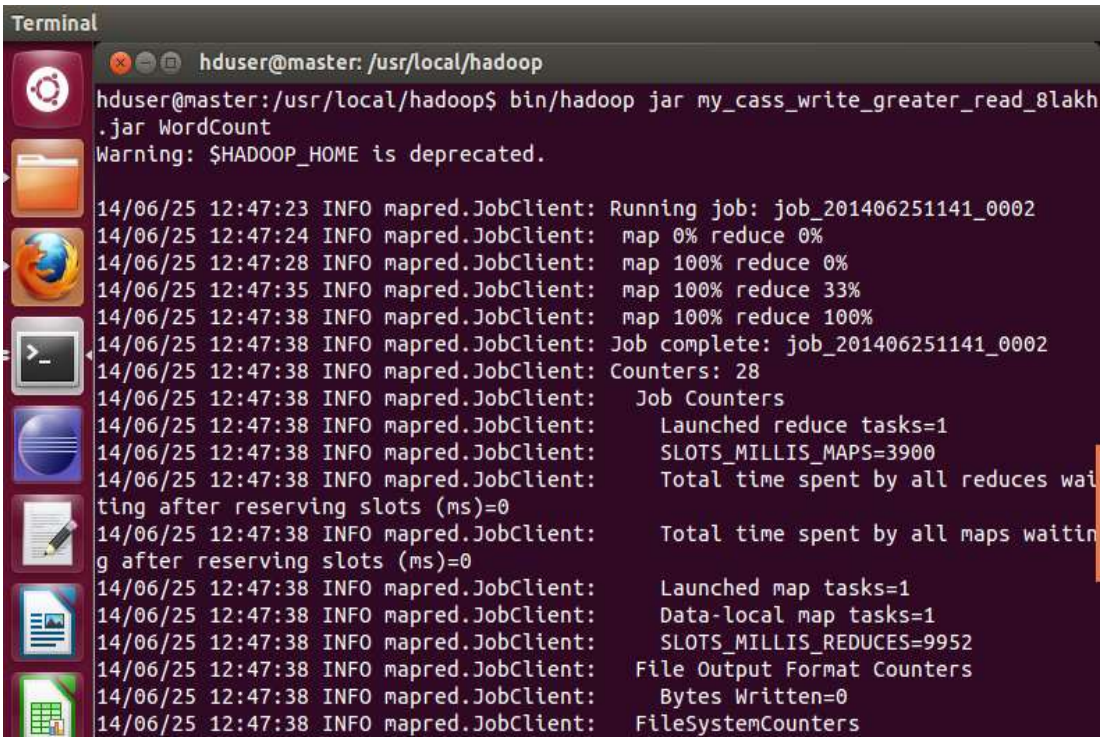


Figure 4.9 Screen shot of execution of Hadoop-Cassandra job

Cassandra Setup: we have used Cassandra 1.0.0 version. To make connection between Hadoop and Cassandra, libraries are used which are provided in the Cassandra package. Cassandra can be started by typing “Cassandra -f” command and after that Cassandra terminal is started by typing “Cassandra-cli” command to insert the data in database as shown in Figure4.7 and Figure4.8 respectively. Execution of Hadoop-Cassandra job is shown in Figure4.9.

The cluster summary in Figure4.10 consist the number of nodes and its status which includes the number of dead nodes and blocked nodes etc.

Cluster Summary (Heap Size is 63 MB/889 MB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes	Graylisted Nodes	Excluded Nodes
0	0	2	8	0	0	0	0	14	14	4.00	0	0	0

Figure 4.10 Screen shot of cluster summary

The list of all successful and failure nodes is given in Figure4.11 which consists the description of all executed jobs.

`

Completed Jobs

Jobid	Started	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information	Diagnostic Info
job_201406251141_0001	Wed Jun 25 12:44:50 IST 2014	NORMAL	hduser	word count	100.00%	4	4	100.00%	1	1	NA	NA
job_201406251141_0002	Wed Jun 25 12:47:23 IST 2014	NORMAL	hduser	wordcount	100.00%	1	1	100.00%	1	1	NA	NA

Figure 4.11 Screen shot of jobs finished list

4.2 Evaluation of Hadoop-MongoDB and Hadoop-Cassandra

We have analysed the performance of MongoDB and Cassandra having Hadoop integration under different scenarios. All the experiments are repeated three times and the mean result is shown over here and all comparisons are done on the basis of processing time in seconds against the number of records.

4.2.1 Test Cases for Evaluation

i. Read and Write Workload: To check performance by varying number of reads and writes intensity. This test case can be further divided into followings test cases.
   a. Read Intense Workload: Number of read operations more than writes.
   b. Write Intense Workload: Number of write operations more than reads.
   c. Read Write Workload: Number of read operations equal to writes.

ii. Scalability: Evaluate by changing number of nodes to process data.

iii. Fault Tolerance: To check the fault handling capability of system.

4.2.1.1 Read and Write Workloads

In this section different workloads are taken for the evaluation from Yahoo cloud serving benchmark (YCSB) list of workloads [17]. YCSB is a tool which is used to evaluate the performance of NoSQL databases by evaluating them on different parameters.

4.2.1.1.1 Read Intense Workload

In this experiment major portion of processing will be taken by the read operations and rest will be taken by the write operations. The application of this type of workload is

`

found in social networking websites because most of the times users reads the data which is given by the friends of the user and not often writes himself on the social networking websites. Figure4.12 shows that for read mostly workload the Hadoop-MongoDB initially for $4*10^5$ records takes 7.36 times more time as compared to Hadoop-Cassandra and for $32*10^5$ records this is reduced down to 5.86 times. Throughout all the test cases of number of records both techniques have almost maintained the same consistency in processing time but Hadoop-Cassandra time consumption is far less as compared to Hadoop-MongoDB . Table 4.1 and table 4.2 shows the total mapping and reducing time spend by both techniques which includes time taken by all the successful and failed tasks.

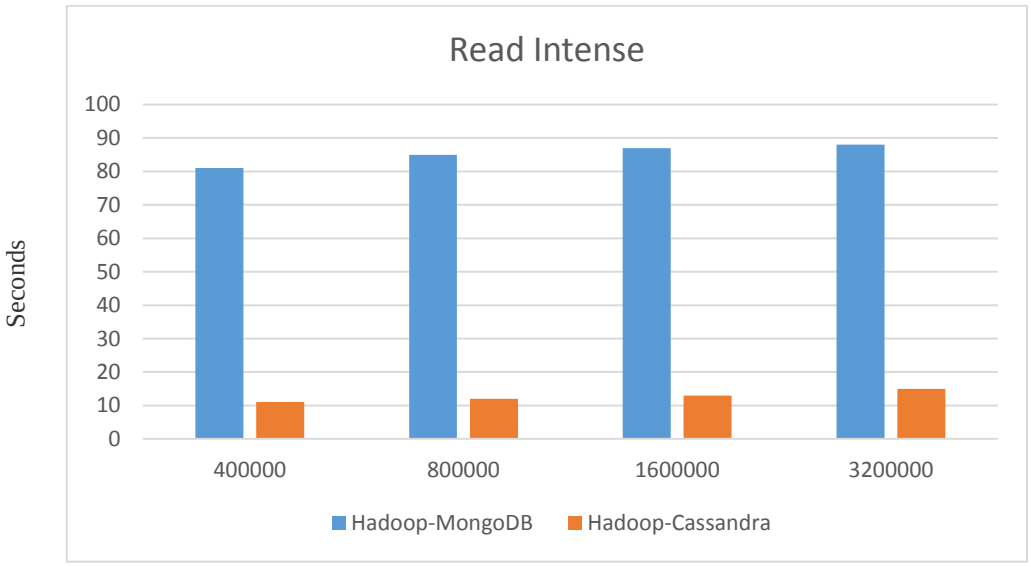


Figure 4.12 Performance evaluation of read intense workload.

Table 4.1 Total mapping and reducing time of Hadoop-MongoDB for read intense workload

Hadoop-MongoDB Technique	**4,00,000**	**8,00,000**	**16,00,000**	**32,00,000**
Total Mapping Time(ms)	166,883	306,598	349,476	1,019,655
Total Reducing Time(ms)	73,116	68,131	93,244	73,708

`

Table 4.2 Total mapping and reducing time of Hadoop-Cassandra for read intense workload

Hadoop-Cassandra Technique	**4,00,000**	**8,00,000**	**16,00,000**	**32,00,000**
Total Mapping Time(ms)	3,671	3,734	4,737	6,747
Total Reducing Time(ms)	7,263	7,287	7,574	8,045

4.2.1.1.2 Write Intense Workload

For this workload the count of writes constitute more than that of reads. The application of this type of workload is found in scientific research where we need to read some records and with time we add various properties to the records as the research is done. Figure4.13 shows that from the starting Hadoop-Cassandra has shown better results than Hadoop-MongoDB. For $4*10^5$ records to $32*10^5$ records Hadoop- MongoDB has shown almost consistent growth in time consumption while for Hadoop-Cassandra the time taken is almost constant from $4*10^5$ to $16*10^5$ records and vary on $32*10^5$ records. Table 4.3 and table 4.4 shows the total mapping and reducing time spend by both techniques including time taken by both successful and failed tasks. Since Cassandra is designed to be write efficient NoSQL that's why the time taken to write have almost constant growth but MongoDB is a read efficient so when records are doubled its time taken also increases in similar manner as that of number of records are increased during the evaluation.

4.2.1.1.3 Read Write Workload

This workload contains 50 (fifty) percent read and writes. Its application will be in the session store recording recent actions. Figure4.14 shows the Hadoop-MongoDB has taken time 6.04 greater when compared to Hadoop-Cassandra during 32 $*10^5$ records computation. The time taken here again by Hadoop-Cassandra is constant and is much less as compared to Hadoop-MongoDB as shown in table 4.5 and table 4.6.

`

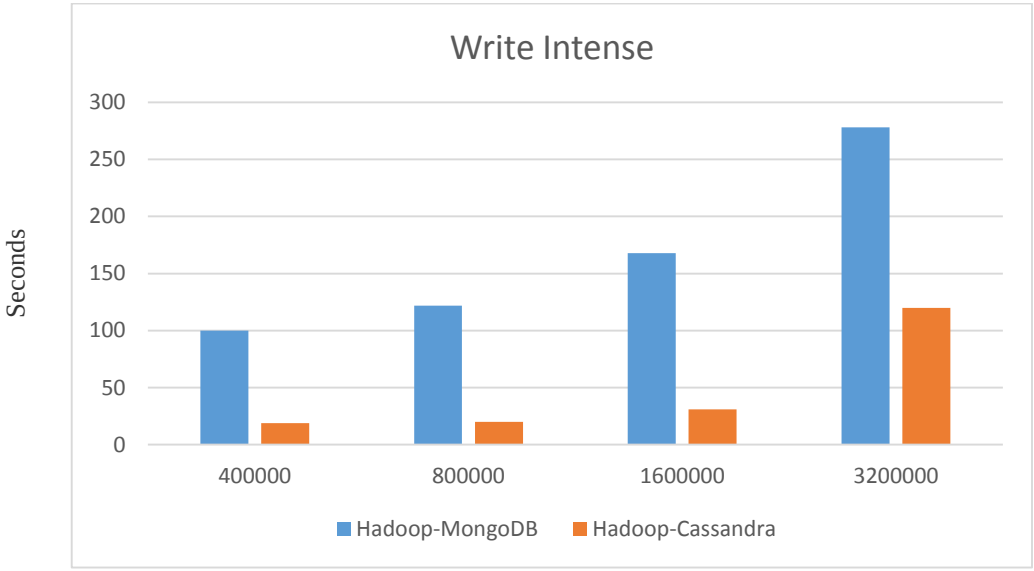


Figure 4.13 Performance evaluation of write intense workload.

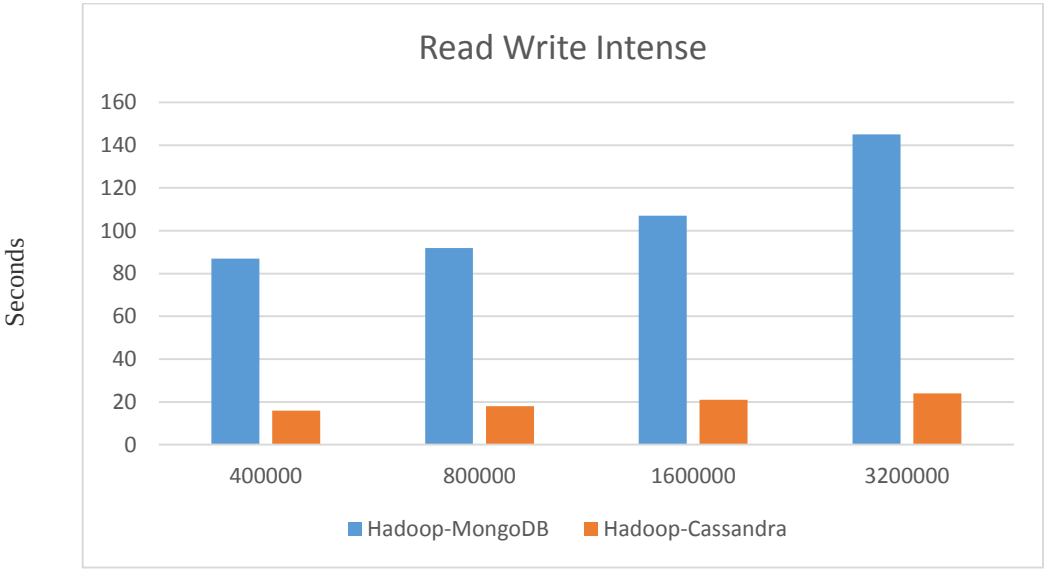
Table 4.3 Total mapping and reducing time of Hadoop-MongoDB for write intense workload

Hadoop-MongoDB Technique	**4,00,000**	**8,00,000**	**16,00,000**	**32,00,000**
Total Mapping Time (ms)	184,129	427,173	450,006	711,906
Total Reducing Time (ms)	92,127	97,679	152,359	257,414

Table 4.4 Total mapping and reducing time of Hadoop-Cassandra for write intense workload

Hadoop-Cassandra Technique	**4,00,000**	**8,00,000**	**16,00,000**	**32,00,000**
Total Mapping Time(ms)	3,701	3,735	4,709	7,108
Total Reducing Time(ms)	14,696	15,574	25,435	112,850

`



Number of Lines

Figure 4.14 Performance evaluation of read write workload.

Table 4.5 Total mapping and reducing time of Hadoop-MongoDB for read write workload

Hadoop-MongoDB Technique	**4,00,000**	**8,00,000**	**16,00,000**	**32,00,000**
Total Mapping Time (ms)	184,884	355,170	371,426	721,496
Total Reducing Time (ms)	78,526	78,453	93,669	123,482

Table 4.6 Total mapping and reducing time of Hadoop-Cassandra for read write workload

Hadoop-Cassandra Technique	**4,00,000**	**8,00,000**	**16,00,000**	**32,00,000**
Total Mapping Time(ms)	3,844	4,061	4,734	7,185
Total Reducing Time(ms)	11,395	12,194	17,036	13,669

`

4.2.1.2 Scalability

In this section we have checked the performance of both Hadoop-Cassandra and Hadoop-MongoDB with increasing cluster size. For computational purpose read and write execution time of $32*10^5$ records is used. In Figure4.15 the data of cluster is scaled from node 2 to node 8 and as it shows the processing time is significantly reduced by factor of 0.47 times in Hadoop-MongoDB and is almost constant in Hadoop-Cassandra. Hadoop-MongoDB has reduced the processing time by more times than Hadoop-Cassandra but still computation time of Hadoop-Cassandra is much less and stable as compared to it. Table 4.7 shows that as cluster size increses the mapping time also increses and reducer time decreases when cluster size is incresed but table 4.8 of Hadoop-Cassandra has shown opposite results as compared to Hadoop-MongoDB of table 4.7. In table 4.8 processing time is decreasing during mapping and increasing during reducing.

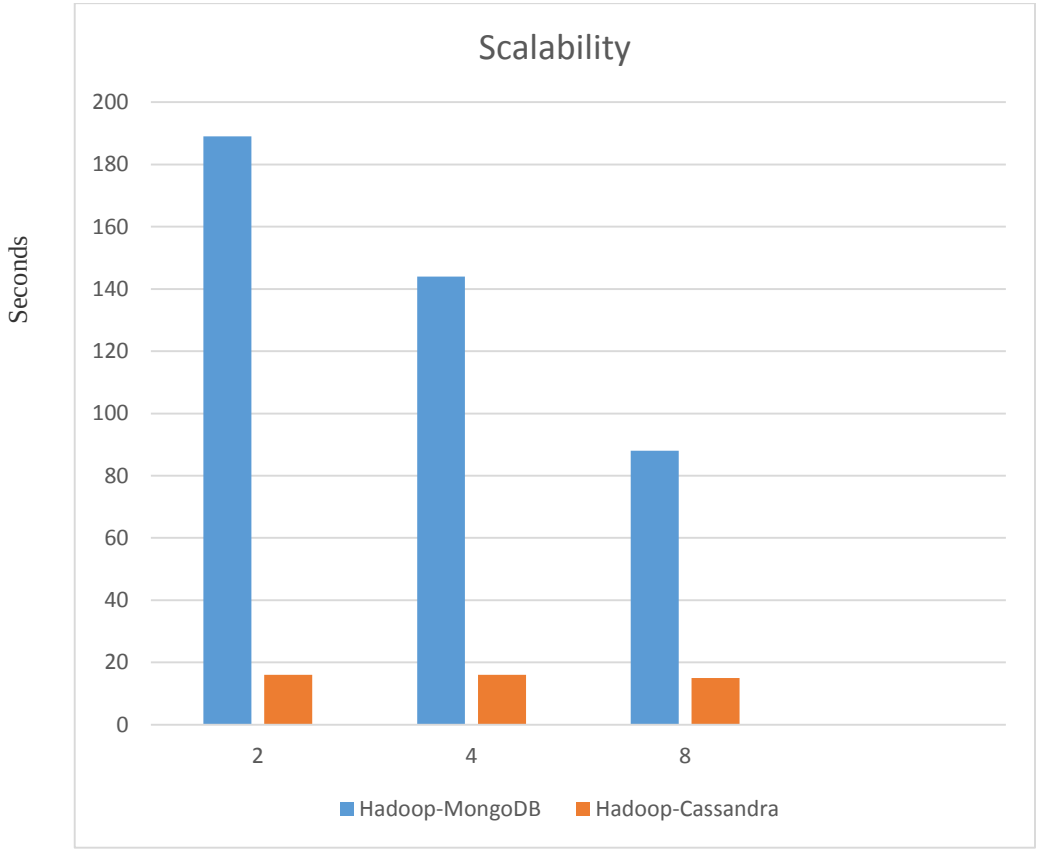


Figure 4.15 Performance evaluation of Scalability

`

Table 4.7 Total mapping and reducing time during varying number of cluster nodes for Hadoop-MongoDB

Hadoop-MongoDB Technique	**2 Node**	**4 Node**	**8 Node**
Total Mapping Time (ms)	679,152	989,832	1,019,655
Total Reducing Time (ms)	169,307	124,164	73,708

Table 4.8 Total mapping and reducing time during varying number of cluster nodes for Hadoop-Cassandra

Hadoop-Cassandra Technique	**2 Node**	**4 Node**	**8 Node**
Total Mapping Time (ms)	7,466	7,059	6,747
Total Reducing Time (ms)	8,006	8,006	8,045

4.2.1.3 Fault tolerance

The fault tolerance of the cluster is checked by failing some node from each cluster while both are doing the map-reduce. Since Hadoop-Cassandra works on peer-to-peer basis the lost nodes data is recovered from the other working nodes and failure of the node have not much effect on the processing time of Hadoop-Cassandra. Hadoop-MongoDB is also able to recover the data with not much delay in processing time but the MongoDB works on the master-slave basis if during the processing the master node is failed the whole map-reduce job will fail. Table 4.9 and table 4.10 shows that both

`

combination can handle the fault tolerance but when the size of cluster is large Hadoop-Cassandra will be able to produce better result because of its peer-to-peer model.

Table 4.9 Total time taken by Hadoop-MongoDB and Hadoop-Cassandra with and without node failure

Technologies	**With 4 Node Failure**	**Without Any Failure**
Hadoop-MongoDB	134 sec	88 sec
Hadoop-Cassandra	36 sec	15 sec

Table 4.10 Total mapping and reducing time during fault tolerance

Total Time	**Hadoop-MongoDB**	**Hadoop-Cassandra**
Mapping (ms)	871,298	6,982
Reducing (ms)	113,796	8,031

`

Chapter 5

Conclusion

In this thesis we have evaluated the performance of two NoSQL databases MongoDB and Cassandra with integration of Hadoop. We have done the experiments on both combination taking parameters of read/write, scalability and fault tolerance. It is shown that the Hadoop-MongoDB is not efficient in reads while it is considered as read efficient NoSql database. Hadoop-Cassandra is much more stable when it comes to writes because for most of the writes it has shown minute increase in the processing time when data size is increased exponentially. When it comes to scalability the Hadoop-Cassandra has shown better processing time in cluster because it distributes the data evenly across the cluster nodes. In fault tolerance because of the peer-to-peer model the Cassandra is able to recover the failed nodes data faster from the other running nodes where its replica was placed. Hadoop-MongoDB is also able to recover the data efficiently and processing time due to failure is not very much effected unless master node of whole cluster fails. Among most of the different categories of evaluation Hadoop-Cassandra has given better results.

In future we can extend this analysis by comparing the indexing capability of Hadoop-Cassandra and Hadoop-MongoDB in map-reduce framework. The analysis can also be done by comparing the data partitioning capability of Hadoop-Cassandra and Hadoop-MongoDB in heterogeneous network.

`

References

1. P. Hitzler and K. Janowicz," Linked Data, Big Data, and the 4th Paradigm," Semantic Web, vol. 4, no. 3, pp. 233-235, June 2013.
2. S.S Nyati, S. Pawar and R. Ingle," Performance evaluation of unstructured NoSql data over distributed framework," Advances in computing communications and informatics, pp. 1623-1627, Aug 2013.
3. Cattell .R, "Scalable SQL and NoSQL data stores," ACM SIGMOD Record, vol. 39, no. 4, pp. 12-27, Dec 2010.
4. A. Boicea, F. Radulescu and L.I Agapin, "MongoDB vs Oracle Database Web Comparison," Third International Conference on Emerging Intelligent Data and Technologies, pp. 330-335, Sept 2012.
5. L. Avinash and M. Prashant, "Cassandra - A Decentralized Structured Storage System," ACM SIGOPS Operating Systems, vol. 44, no. 2, pp. 35- 40, April 2010.
6. Redis website homepage. [Online]. Available: http://www.http://redis.io/
7. Neo4j website homepage. [Online]. Available: http://www.neo4j.org/
8. N. Rajesh, F. Hans, G. Steven, K. Marc, L. Herman, L.C Harry, et al, "Scaling Memcache at Facebook," Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation, pp. 385-398, April 2013.
9. P.Z Cheng, J.J Ze, C.B Xioa and Z.K Zhi, "Real-time analytics processing with MapReduce," International Conference on Machine Learning and Cybernetics, vol. 4, pp. 1308-1311, July 2012.
10. A. Veronika and B. Jorge, "NoSQL databases: MongoDB vs Cassandra," Proceedings of the International Conference on Computer Science and Software, pp. 14-22, Oct 2013.
11. D. Elif, G. Madhusudhan, G. Daniel, C.S Richard and R. Lavanya, "Data Performance evaluation of a MongoDB and hadoop platform for scientific analysis," pp. 13-20, June 2013.

`

12. T. Dory, B. Mejias and P.V Roy. N.L Tran, “Measuring elasticity for cloud databases,” Proceedings of The Second International conference on cloud Computing, GRIDs and virtualization, 2011.
13. D. Elif, S. Bedri, K. Pinar, H.Jessica and G. Madhusudhan, “An Evaluation of Cassandra for Hadoop,” Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing, pp. 494-501, June 2013.
14. R.P Padhy, M.R Patra and S.C Satpathy, “Rdbms to nosql: Reviewing some next-generation non-relational databases,” International journal of Advanced Engineering Science and Technology, 2011, vol. 11, no. 1, pp. 15-30, 2011.
15. HBase website homepage. [Online]. Available: http://hbase.apache.org/
16. K. Bakshi, “Considerations for big data: architecture and approach,” Aerospace Conference, pp. 1-7, 2012.
17. B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan and R. Sears, “Benchmarking cloud serving systems with ycsb,” Proceedings of the 1st ACM symposium on Cloud computing, pp. 143-154, June 2010.
18. B. Eric, “CAP twelve years later: How the "rules" have changed,” Computer, vol. 45, no. 2, pp. 23-29, Feb 2012.
19. Voldemort website homepage. [Online]. Available: http://www.project-voldemort.com/
20. CouchDB website homepage. [Online]. Available: http://www.couchdb.apache.org
21. JavaScript object notation (JSON) website homepage. [Online]. Available: http://www.json.org
22. J. Rats and G. Ernestsons,“Using of cloud computing, clustering and document-oriented database for enterprise content management,” 2013 Second International conference on Informatics and Applications, pp. 72-76, Sept 2013.
23. K. Kaur and R. Rani, “Modelling and querying data in NoSql databases,” IEEE conference on Big Data, pp. 1-7, Oct 2013.
24. Image. [Online]. Available: http://database.51cto.com/art/201107/278609.html
25. Image.[Online].Available:http://www.enterpriseirregulars.com/39209/the-real-potential-impact-of-sap-hana/

`

26. Image.[online].Available: http://socurites.com/124
27. L. chuck ,“ Hadoop in action,” manning ,2011, pp. 24
28. Image.[online].Available:http://www.cs.rit.edu/~ark/spring2012/730/module08/notes.shtml
29. Image.[online].Available:http://www.drdobbs.com/database/hadoop-the-lay-of-the-land/240150854
30. Image.[online].Available:http://docs.mongodb.org/master/MongoDB-crud-guide.pdf
31. Image.[online].Available:http://www.credithealer.com/picspab/index.php?p=database-column-oriented

`

List of Publications

1. Ayush and S. Bawa, “Performance Analysis of NoSQL Databases Having Hadoop Integration,” International Conference on Emerging Research in Computing, Information, Communication and Applications (ERCICA), Aug 2014. Elsevier (ACCEPTED)