



L OVELY
P ROFESSIONAL
U NIVERSITY

Handling Large Data

A Dissertation Proposal Submitted

By

Sajida Parween

Abhijit Borah

Sandeep Sharma

To

Department of Computer Application

Submitted in Partially Fulfillment of the Requirement for the

Award of Degree of

Master of Computer Application

Under the guidance of

Navpreet Kaur

Assistant Professor

May 2015

ACKNOWLEDGEMENT

It is great pleasure to acknowledge the assistance of contribution of many individuals to this effort.

We take this opportunity to thank all the people who have helped us through the course and producing this report. We sincerely thank Ms. Navpreet Kaur for her guidance, help and motivation. Apart from the subject of our course, we learnt a lot from her, which we sure, will be useful in different stages of our life.

We are grateful to Mr. Rishi Chopra, Discipline of Computer Application for helping us in pursuing this topic in a smooth and organized manner.

We appreciate the contribution our friends whose name has not been listed here but has been much helpful not on the course of this project but also in the course of my study.

Lastly, but not least, we gratefully acknowledge the support, encouragement and patience of our families.

Sajida Parween (11401811)

Abhijit Borah (11401844)

Sandeep Sharma (11401455)

CERTIFICATE

The work described in this report has been carried out by Abhijit Bohra, Sajida Parween and Sandeep Sharma has completed their MCA Research Paper Writing Proposal titled “**Handling Large Data**” under my supervision. I certify that this report work is their confide work, original and no part of the dissertation proposal has ever been submitted to any other degree or diploma.

The proposal is fit for the submission and the partial fulfillment of the conditions for the award of the degree of Master in Computer Applications.

Date

Supervisor

Place:

.....

DECLARATION

We do hereby declare that the report entitled “**Handling Large Data**” is an authentic work developed by us, under the guidance of *Ms. Navpreet Kaur* and submitted as partial fulfillment of the requirement for the award of degree of MCA.

We also declare that, any or all other contents incorporated in this report have not been submitted in any form for the award of any degree or diploma of any other institution or university.

Sajida Parween (11401811)

Abhijit Borah (11401844)

Sandeep Sharma (11401455)

INDEX

Topic	Page No
1.Abstract	6
2.Introduction	7
3.Literature Review	8
4. Proposed Objective	10
5.Problem Definition	20
6. Proposed work plan	21
7.Scope of the study	26
8.Conclusion	27
9.Reference	28

ABSTRACT

Big data contain a large collection of datasets, so it needs technologies to handle it. In this paper we are going to discuss about two technologies that mostly companies are using nowadays. The most common one is Hadoop and another new one is Cassandra that comes up with some new addition feature as compared to Hadoop. Both stores terabytes of data and has ability to handle big data in more efficient way. This paper talk about the problem occurs in both and has some proposed work.

Keywords: *Hadoop, HDFS, Cassandra and CFS*

INTRODUCTION

Data is the collection of values and variables related in certain sense and differing in some other sense. The size of data has always been increasing. Storing this data without using it in any sense is simply waste of storage space and storing time. Data should be processed to extract some useful knowledge from it.

Big data is data that exceed the processing capacity of conventional database systems. The data is too big, processed slow, or doesn't fit the structure of our database architecture. Today data are generating from everywhere like various sensors, transportation data, weather forecast, social network etc.

When we are thinking about how we are going handle big data and process it with efficient and effective manner then we come up with two frameworks that are: **Apache Hadoop and Apache Cassandra.**

Hadoop is an open source framework given by Apache software foundation for storing huge dataset and processing huge dataset with a cluster of commodity hardware. As we say that Hadoop use commodity hardware which means cheap hardware. Here Hadoop used one old but effective technique "Divide and conquer" to store and process huge amount of data.

Cassandra is also an open source framework which is decentralized in structure. Cassandra is peer-to-peer distribution of data stored in system which can hold large amount of data like terabytes. It replicates data over many different nodes to provide consistency, high performance and availability. Cassandra offers is high availability with no single point of failure, robust support for clusters over multiple data centers resulting in the lower latency operations for the clients.

LITERATURE REVIEW

By: Samson Oluwaseun Fadiya, Serdar Saydam and Vanduhe Vany Zira - **Advancing big data for humanitarian needs**

At the present moment, almost every business is witnessing a bursting of data. Big data now available for analytics present complex, daunting challenges due to the vast number of digital data generated daily by different organizations. The vast amount of data has improved the global community's ability to defend and allow for progress of rights of vulnerable people around the globe. The main motive of this paper is to propose a big data platform for large-scale data analysis by using the Map Reduce framework for unstructured data stored into integrating distributed clustered systems such as NoSQL (Not Only SQL) and Hadoop Distributed File System (HDFS).[3]

By: Chanchal yadav, Shuliang wang, manoj kumar - **Algorithm and approaches to handle large data**

In recent times there has been progression shift in the amount of data capture and stored. This shift has taken place from users to the client, from clients to connected devices like sensors, weather forecast, social network etc and then the explosion in the usage of those devices. This has led to an inception in the amount of data capture and stored which cannot be handled by the traditional database systems. The paper describes different methodologies, algorithm, and architecture to handle large data sets and their related issues.[4]

By: Ashish A. Muth and Vaishali M. Deshmukh - **Cassandra File System over Hadoop Distributed File System**

This paper is discussed about the how Name Node in Hadoop is eliminated by Cassandra. This paper talks about the HDFS, how it process and its benefits' which is different from CFS, its process and its benefits. From this paper we can easily define how HDFS is master-slave architecture and Cassandra is masterless architecture and Cassandra remove single point of failure.[1]

By: Avinash Lakshman and Prashant Malik - **Cassandra - A Decentralized Structured Storage System**

In this review paper Avinash and Prashant bring a good use of Cassandra in form of Facebook. They discussed how Facebook use Cassandra for "Inbox Search". Search returns all messages that sent or received of entered user name or message keyword. This inbox schema consists of column families that are done in two ways. First one is: user-id becomes the key and key-word of message becomes super column. After that all message identifier become column that comes under super column. The second one is: here also user-id becomes Key but receivers-id becomes super column. The individual message identifier comes under these super columns. To provide the fast access it use caching technique in which if any search bar is clicked by user, message is send to Cassandra cluster's buffer cache with user's index. [2]

By DATASTAX Corporation - **White Paper**

With the coming of RDBMS databases in the picture, applications extended their abilities and discover new possibilities that weren't possible with the old file system. But along with the time passage, applications arose up to new levels of scalability that weren't expected back in the old days. But to match the client's needs, scaling isn't enough; organizations need their services and applications to be always available and lightning fast in their performance. This is where the need of a completely new concept becomes visible. And this is where Apache Cassandra completes the picture. It is a fully distributed architecture that fulfills the need of 3V's of Big Data – Velocity, Variety, and Volume. This paper covers brief overview of Apache Cassandra and includes necessary details. Also, this paper explains that why and how out of various distributed databases vendors, only Cassandra offers exactly what is required to handle the legacy of the modern applications.[10]

PROPOSED OBJECTIVE

The objective of this paper is to handle the big data in efficient way. As we know to handle the large data we need some terabytes of storage medium and make it in such a way that it can be processed swiftly without any failure. This paper talk about the two technologies which are currently used to handling big data those are: Apache Hadoop and Apache Cassandra.

Hadoop and Cassandra provide scalability, reliability and high performance to handle big data but at some point both has some disadvantages too in which reliability may affected and performance may decrease. In this paper, we discuss about the some disadvantages of each and by assuming some different way, we try to solve the problem.

In Hadoop architecture the problem comes in name node and job tracker and in Cassandra problem comes when replication is done in neighboring node. The proposed work in this paper gives a brief idea of how to overcome from these problems that occurs in Hadoop as well as Cassandra.

We are trying to handle name node as well as job tracker in Hadoop by increase the number of name node in a single cluster and by use of Avatar node which has all the information and ready to take place whenever it required. We are thinking a different way to store replicas in different nodes and keep track of all replicas node in an index table. So that there is no single point of failure occurs and it can handle query as fast as it can.

HADOOP

While talking about big data, there always a word that people are liked to talked about that is “Hadoop”. When we heard that word our mind filled with many questions like what is Hadoop. How it is going to store large amount of data and how it able to process such a huge amount of data?

Hadoop is an open source framework which has been overseen by apache foundation. It is used to store huge dataset and process it. But it’s not recommended with small one reason behind is that the main rule of Hadoop is divide and concur. If we are distributing a small work into different pieces than it requires more time to aggregate them into a single one. For example if we are distributing 10 GB of data to different machine for processing and the same amount of data process in a single machine. It seems single one works faster than distribute one.

Hadoop basically use cluster of commodity hardware to store huge data set and process it. Cluster basically set of machine in a single LAN. Commodity hardware is the cheap hardware. For example the hardware that are used in our laptop or in PC.

Hadoop used two terms HDFS and Map Reduce. HDFS stands for Hadoop Distributed File System. Which is used for storing data? It is specially designed file system for storing huge data set with cluster of commodity hardware with stream access pattern. Here we used a term “specially designed” file systems are storing file and directories. We store our files and directories in a hard disk so we may call it as a file system. It is able to store or retreat file when we boot its voice.

a. How HDFS is different from Traditional Structure



Figure: 1.1 Basic difference between HDFS and Traditional Structure

Suppose we have 500 GB of Hard disk which is collection of 4 kb of blocks. Now we are going to store 2kb of file in 4kb block which means the remaining 2kb of that block not in used and it will be waste. This is happened in normal file system.

Now in the above of this traditional file system we are going to design HDFS or install Hadoop with HDFS. HDFS, by default give some block size which is 64 MB that is why it is called as specially designed file system. We can make it as 128 MB as well.

After install HDFS. Suppose we are going to store a 35 Mb of file in a block which means 29 MB is still remain in that block. In normal file system it will not be released but using HDFS it will release that remaining amount of space, so that we will be able to use it in future.

Now in terms of block size which creates a huge difference between 4 kb and 64 MB the reason behind is here we are going to store huge dataset. Which include data about data known as Metadata? If block size are small than the size of Meta data will be increase and it requires more space to store that information.

b. How HDFS work

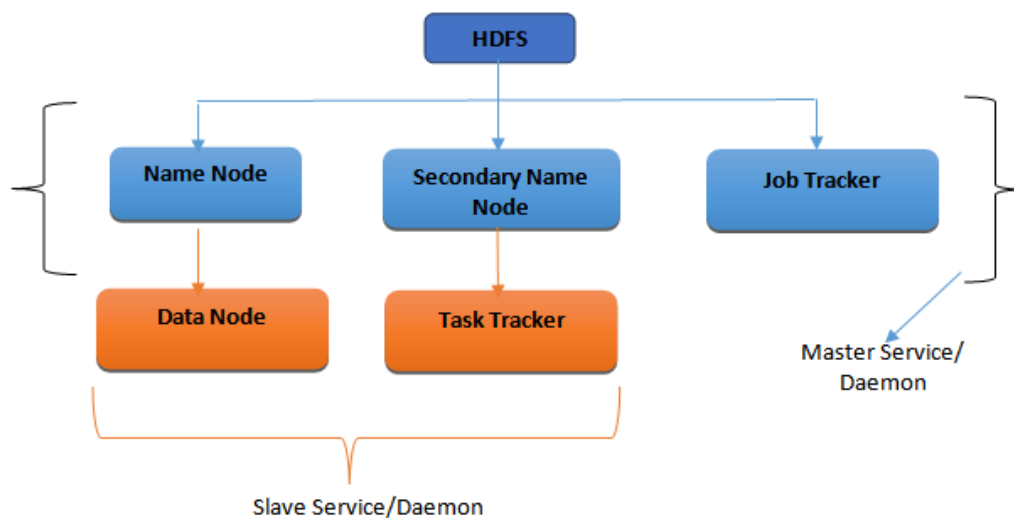


Figure: 1.2 HDFS Services

Basically HDFS provide 5 different services or daemons. Which is combination of Master and Slave Here three are taken as master and remaining two are taken as Slave. Here we first start with master and later on slave also. Each of the daemons run in its own Java Virtual Machine (JVM).

Name Node: These daemons stores and maintain Meta data

Secondary Name Node: It performs housekeeping functions for Name node.

Job tracker: Manage Map Reduce jobs; distribute individual tasks to machines running the task tracker.

So these are the basic introduction about Master nodes. Now let's go through with slave one

Data node: Actual HDFS data blocks.

Task Tracker: Responsible for instantiating and monitoring individual Map and Reduce Tasks.

Here we used one term daemon which means background process. It doesn't have any physical appearance. All this services are running internally. Here every master can communicate with each other and every slave can communicate with each other. But slave cannot communicate with other master and master can't communicate with other slave. For example: name node able to communicate data node and vice versa, and job tracker able to communicate task tracker and vice versa. But name node can't communicate with slave task tracker and vice versa, and job tracker can't communicate data node vice versa.

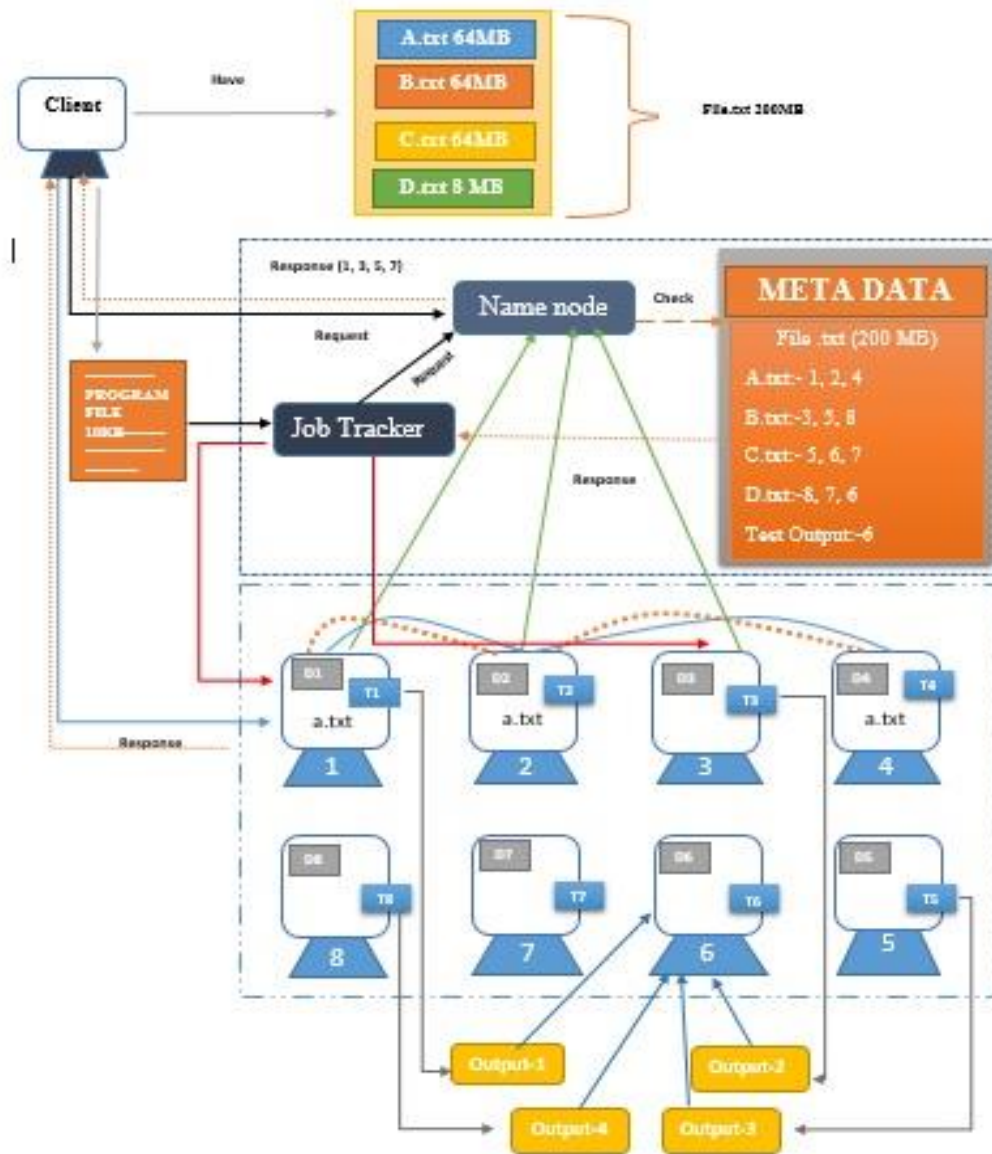


Figure: 1:3 Data Store and Process work of HADOOP

Suppose we have a file named as File.txt and size of file is 200 MB. You may think how 200 MB is considered as large data set. The reason behind is for computation purpose we assume that files as size 200 MB. If we are going to store this file into HDFS then it split into 4 sub parts. Here first three parts use 64 MB each and Last part use 8 MB of blocks.

At first Client send a request to name node. After receiving it request name node start filling Meta data about file.txt. Which include name of file, size of file, and split file name? After fill those information name node give a proper response to client and provide the data node where data is going to store. For example here name node send acknowledgement to client that D1, D3, D5, and D7 is free to store your data. After receiving this much information client knows very well which are the data node and where are the free spaces. Now client approaching system 1 and its keeping a.txt file.

Here one thing we keep in our mind that files are not store into sequence order or allocate into a sequence order in a block it can be in any order.

One thing we already discussed that Hadoop used cluster of commodity hardware. So if hardware is going to down than we are going to lose A.txt file. So to overcome these problems Hadoop gives by default 3 replication of same file. Which means if our file size is 200 MB than it takes 600 MB by creating 3 replica of each of input split? Suppose after storing A.txt in D1, D1 internally keeps it in D2, and D2 internally keeps it in D4. After these processes complete its time to give a report to client that input split is save without facing any problem. To do this first D2 receive acknowledgement from D4 and D1 receive acknowledgement from D2 and after that D1 send a report to client with all other replica address.

All Data node give a proper block report and heartbeat to name node after a period of time. Which include either some data is stored or not, and working properly or not.

Suppose one node not give block report at given point of time than Name node assume that there must be some problems in it and try to find where its replica belongs. For example if D1 doesn't give proper heart beat after a period of time than it thinks that D1 is dead and name node try to find address of other replica. After find the first replica it's going to create a third replica into a new system and save its address to metadata.

c. How map reduces work

Map Reduce that is basically going to process the data which stored. Now take a scenario that suppose we have 10 KB program through which we are going to process 200 MB of data. There may two possible cases that come out either we send program towards data store or we take back our data to the local machine where we write the program. Before Hadoop it's not possible to send program towards data because computation is always processor bound. So here our another master daemon come into the picture named as Job Tracker who is going to taking care of these request, trace them, Manage Map Reduce jobs, distribute individual tasks to machines, and running the task tracker. For better understanding let's take one example suppose we have 10 KB of program file which we want to process on file.txt. Now Job tracker take that request where client may write a program of count the number of words in a file with its output directory name which named as TestOutPut.

Here the twist comes Job Tracker doesn't know the locations of all the data nodes to process the program. So it's going to communicate with name node to know the locations of all the data nodes. After check it from the Meta data name node is going response back accordingly. After receive all the information now it's time to assign the task to the entire task tracker. As we are already discussed that same file have by default create three replicas into a three different systems. Suppose Job tracker wants to process a.txt file which is available into system number 1, 2, and 4. Here job tracker doesn't assign the same task to three different task trackers. To assign a task to task tracker its check some parameters like which task tracker is the nearest one, which one is free or able to process it in time? Suppose here job tracker assign the task to T1. Now T1 take that 10 KB of program and process it on local A.txt which called as Map. After assigning task to T1 now Job Tracker want to process the same program into B.txt file. So to do that first it finds the location and then assigns the task to task tracker. Here it assign task to system number 3. In Hadoop number of map running in a cluster = number of input split of an input file. Here also Task Tracker sends a heart bit to Job tracker after 3 second to know that it's in proper work. Suppose one Task tracker not sends its bit then Job tracker will wait for 10 times more than regular one to get response from it. Even if it not gets bits than it thinks that it may be in very slow or not working. If these kind of scene comes than job tracker assign task to next task tracker corresponds to data node where it's available.

From each of the input splits file we get an individual output. Here we have 4 input splits assume it that after process each file it gives 4kb of output. But it's not taken as output of File.txt. Here Reducer combines all the output of the input splits and stores it into a local data node. When this data node give block report and heart bit to name node, it's going to generate a fresh Meta data about it. One thing always keep in mind that number of reducer= number of output file.

Client can always watch up to what extend job is done. If it's complete than it check the metadata to get output file.

APACHE CASSANDRA

Cassandra Apache is an open distributed DBMS that use NoSQL database. It handles the vast amount of data across different servers or multiple data centers. Cassandra Apache emphasizes Denormalization and provides features like Collection (Map, List and Set). Therefore, it (Cassandra) does not support "Joins and Subqueries" except for batch analysis through Hadoop. It is decentralized in nature as it is master less node and all nodes in cluster has the same role so, any node can serve any request from client. Dataset and processing huge dataset with a cluster of commodity hardware.

As Cassandra use NoSQL database that basically aims to the data which is unstructured, has dynamic nature, and has potential for big data research work. The traditional databases are too costly for that and also unfeasible & unpractical to handle with SQL databases because of the lack of structure, scalability demands, and the elasticity requirements. NoSQL data subjects like MongoDB and Cassandra promises a perfect platform for fast and better queries on data.

It is developed to substitute the old Hadoop Name Node. Its major advantage is that it simplifies the operational overhead of Hadoop by eliminating the failure in the single point of Hadoop Name Node.

The main features of Cassandra are:

- Continuous availability.
- No single point of failure.
- Linear scalability.
- Maximum flexibility.
- Fast response time.

a. Cassandra Architecture

Cassandra cluster follow peer-to-peer distribution over nodes in form of “**Ring**” architecture that is easy to maintain, So Cassandra is **master less architecture**. All nodes contain a unique token in ranges and have key space. All nodes in ring are interacting with each other through gossip protocol. It provides replicas of data across nodes; so if any node goes down or failed, there is another mode from where we can use that replicas data.

Replication can be done in two ways:-

1. Rack Unaware Strategy: in this replicas are always stored on next node.
2. Rack Aware Strategy: replicas are stored in the first node of another data center. It also avoid “hot spots”.

Example: if there is A, B, C, and D nodes in increasing Token Order. If A and B in datacenter1, C and D in datacenter2 then A and C will contain more data because they contain replicas for all token of other datacenter and hence it provides fault that can be tolerable.

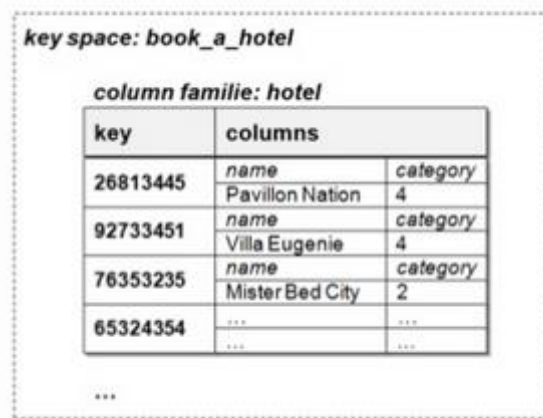
Cassandra capable of **handle Peta-bytes of data** and it is **Scalable in nature**, means that we can add as many node as needed.



Figure: 2.1 Scalable- perform operation per second according to nodes.

b. How Cassandra works Using CFS (CASSANDRA FILE SYSTEM):

All nodes are in form of “Ring architecture” and all nodes contain key spaces. CFS Contain information about metadata in key space, which further contain Column families that actually hold the data. These data in the column families are replicated over the data center to maintain fault tolerance.



The diagram shows a key space named 'book_a_hotel' containing a column family named 'hotel'. The column family is represented as a table with three columns: 'key', 'name', and 'category'. The table contains four rows of data, with the last row showing ellipses to indicate more data.

key	name	category
26813445	Pavillon Nation	4
92733451	Villa Eugenie	4
76353235	Mister Bed City	2
65324354

Figure: 2.2 Key Space contain column family

The “inode” work as main node which track all files of data in block and store all column families. “inode” hold its own column families which consists of file-name, user, file-type, path and list of block. Another column families is of sub block which contain the data that is traced by inode column families. When new data is added to the node then CFS writes metadata of that new node to “inode” and all data about new node is written into the new sub-block.

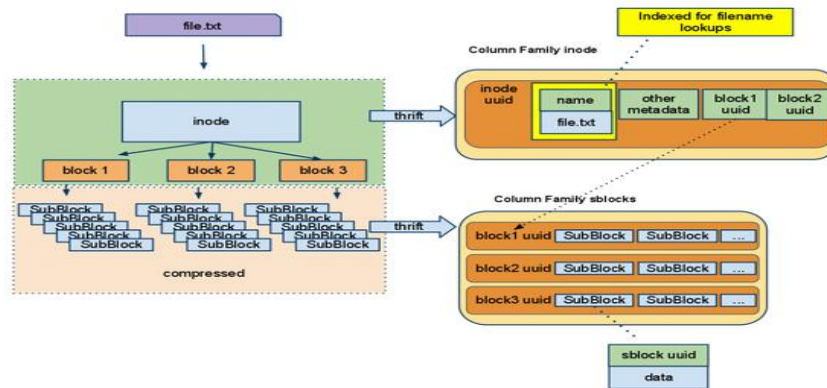


Figure: 2.3 How to access file or data in node

As all nodes has same roll, so client can connect to any node to perform any request. The connected node play role of coordinator node which handle the client request. It also stores the replica of requested data. If coordinator node has not the required data then coordinator node send request to other replica node to handle the query.

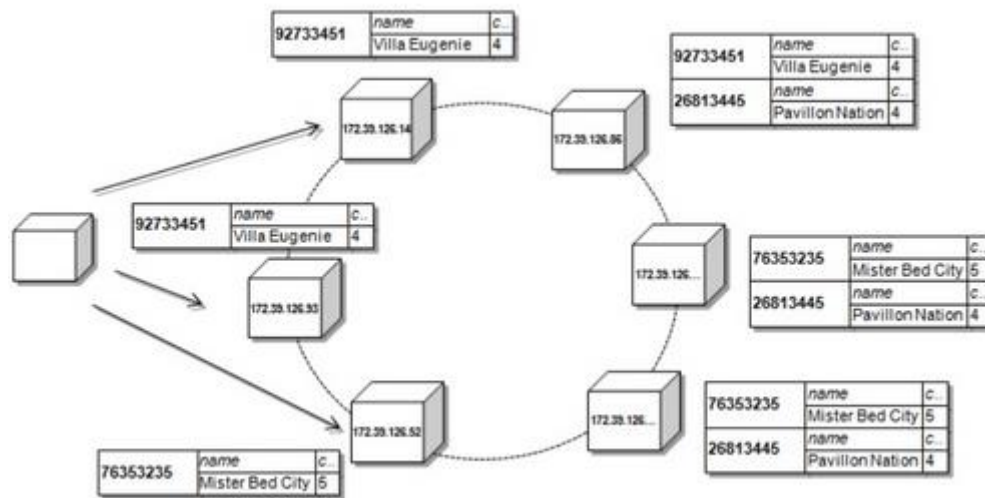


Figure: 2.4 Data store in node

For make is easy, Cassandra use partition to determine the replica node. It is done in two ways of partition:-

- a. Random Partitioner: - RP provides the key-value-pairs throughout an area network that results in good load balancing. In this we have to access more nodes to get no. of keys.

- b. Order Preserving Partitioner: - OPP provides the key-value pairs in such a way that similar keys are close not far away. In this fewer nodes have to be accessed but sometime distribute the uneven key-value pairs.

By default Cassandra use RP and first replica is determine by using MD5 hash technique. From this row key is calculated and after that that node is selected key comes within the token range. And that token range node is responsible to store the replica of requested data.

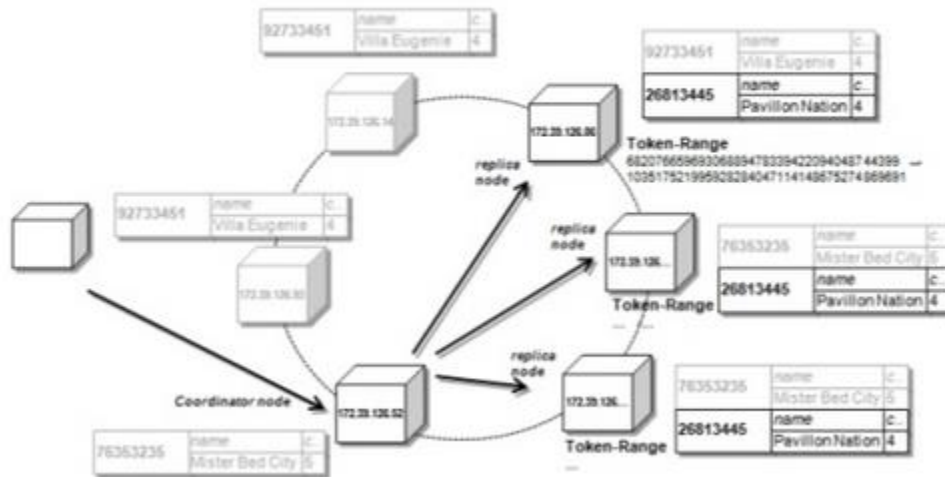


Figure: 2.5 working with coordinator node

PROBLEM DEFINITION

1. Bottleneck of HADOOP (In Hadoop)

Single Point failure: When start working in a distribute architecture there are many things that start working wrong like hardware failure, nodes are not in properly work which effect on the availability and reliability of our data. In Hadoop architecture the problem comes in name node and job tracker. If job tracker is fail to work properly or it's slow than it can affect on task because while we start process on Hadoop job start use different type of hardware resource during its life time and coordination become unbalanced. Suppose in the middle of your work job tracker start not responding than all the information of task tracker are in lost. Same scenario comes under Name node which contains all the information. If name node dies than Meta data is going to be loose as well which means you have your data in HDFS but you don't know where it is actually. Sometime, it happened that in the middle of your job task node start not responding. To solve this problem job tracker start search for another data node and restart that task once again. We have one secondary name node but that is only for back up purpose it's not ready to take place name node in the time when primary name node is going down.

Troubleshooting problem: We already discussed that client always able to check how much works complete. There are many monitoring tools available which helps client to know about their works which are not efficient enough makes it difficult to isolate the root cause of problems and drives a lot of inefficient behavior, like guess-and-check restarting and asking users about jobs they submitted. As size of a cluster grows and businesses depend on Hadoop, such methods will become unsustainable.

Cheap makes it Expensive: We never able to predict how much workload will come in next time. The main goal is that not to lose information about data and tasks because Hadoop's inefficient, up front allocation of resources every time we require highly expensive hardware. Here utilization of resources is not efficient enough.

Multi Tenancy: If you go to the deep of Hadoop working then you may find that Hadoop only work one type of job in a particular system. This means you are not able to assign another job to the same machine.

Everything we run into Hadoop run through map reduces. Now map reduce programming model only helps in batch processing kind of jobs we can't do any real time analysis, graph processing or any interactive jobs.

2. Replication in neighboring node may leads to delay in accessing data (In Cassandra)

Currently replication is done to next successor nodes. At this point if any two or more neighboring node fails then it can lead to network traffic problem. Due to network traffic problem it leads to unavailability of node for some time and due to unavailability of node further delay in accessing the data. So replication of data to next successor nodes makes the effect of failed node locally. Any failing node has significant effect on neighboring nodes.

The huge network data traffic can have significant impact especially on if two or more neighboring nodes fail at a time. One expected result could be that huge amount of data would be unavailable for long time. The failed node's neighboring nodes could be heavily overloaded at the time of balancing the number of replicas and as result could not serve requests for the whole recovery time.

PROPOSED WORK PLAN

1. Bottleneck of HADOOP (In Hadoop)

Everyday world is going to be change. New idea arises to solve a different problem which helps us to find a better solution according to the requirement that we can produce same result in different ways. For example if someone give you to write factorial program we can write that in different ways using functions , class, objects etc.

Our first object is to tackle the Single point of failure and increase its reliability and availability. To do this first we have to increase the number of name node in a single cluster so that the workload is to be decrease. Every name node has one Standby node named as Avatar node which has all the information and ready to take place whenever it required. Every name node able to communicate with other name node and able to creates maximum of 3 replica of meta data file which is going to store in other name node with their Meta data. So in extreme condition Avatar, Primary node, and Secondary node is going to down then there is a possibility to recover 1the data.

To reduce the work load of a Job tracker here corresponds to every name node we proposed to increase the number of Job tracker maximum up to 2 and there must be a log file which going to maintain its activity. So that if there any problem occur than we don't need to start it from beginning. Here both the Job tracker divides their task and work accordingly. If one is going to down than it send information to second one to do the remaining task. Through this approach it reduce the time, increase its performance, as well as it reliability also.

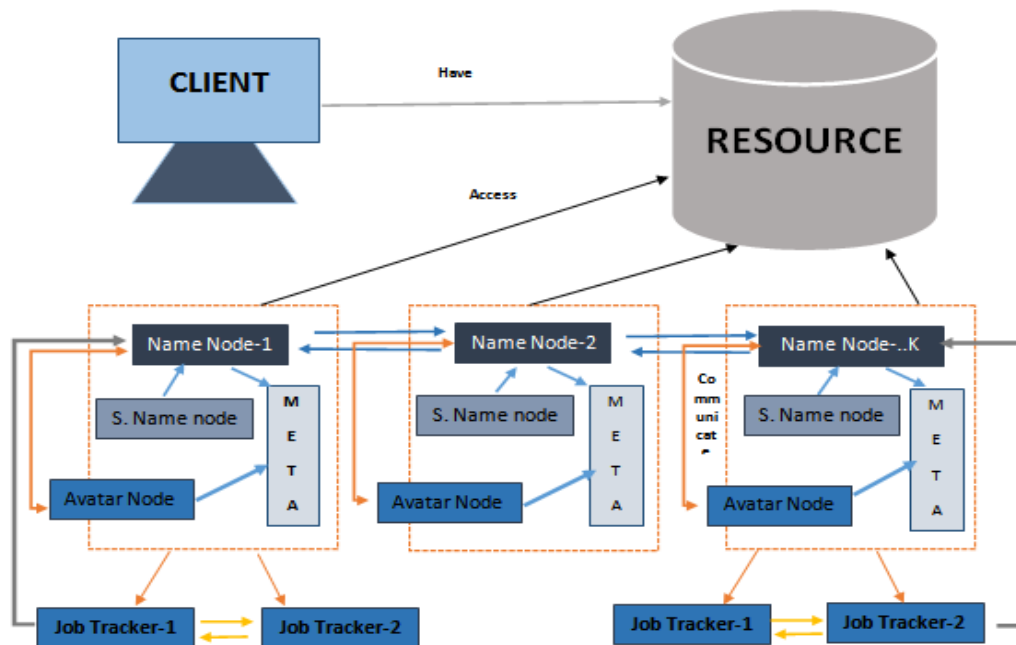


Figure: 3.1 Proposed Architecture

2. Replication in neighboring node may leads to delay in accessing data (In Cassandra)

a. Solution:

Solution of the above problem is to store the replication to different node instead of storing it to neighboring nodes. As if two neighboring nodes fail due to network problem then it will turn out to be network traffic which can leads to unavailability of node for some time and which further delay in accessing the data. So it's better to store the data to different node. Storing data to different node will be providing more chances to have access to the required data. In this case, it doesn't matter if two or three neighboring nodes failed as we will have another way or node to reaching required data.

As we plan to store the data in different- different node then here it's required one table which can keep track of all node and corresponding to their replicas. So we make one table which will store nodes, range, keys and replica node name which help us to maintain the list off all nodes as well as replicas node.

⇒ Table contains:

- a. **Node:** name or value of all nodes in network or server.

- b. **Range** : token range
- c. **Key**: contain all key value of each node.
- d. **Replicas**: name of replicas node corresponding to each key.

b. Proposed Worked:

Maintain a index table of every node that contain the key space name or key value, Token range using MD5 hash algorithm and name of those node which having requested data replicas.
Example: -

A				B			
Range::0-25000				Range: 25001-35000			
Key	Replicas			Key	Replicas		
20345	B	C	D	30089	A	C	D
20367	B	D	E	30068	A	C	D

Figure: 3.2 Index table for Cassandra framework

This index table using “hinted handoff writes” mean if any network issue or problem occur then coordinator stores has hint that which replica node is goes down or which replica node is not responding and mark that replica as dead for some hour. If after 3 hour, again that node is not responding then mark it as permanently dead.

A				B			
Range::0-25000				Range: 25001-35000			
Key	Replicas			Key	Replicas		
20345	B	C	D	30089	A	C	D
20367	B	D	E	30068	A	C	D

Figure: 3.3 Index table using HHW

c. Algorithm for making index table:

```
Step 1   Make column for N node (if new node is added then new column is established).
Step 2   for each (node)
        {
Step 3     Add token range by using Random Partitioner MD5 algorithm;
        }
Step 4   for(every node)
        {
Step 5     Int x=initial token range,y=final token range;
Step 6     for(i=x;i<=y;i++)
        {
Step 7       add( key and corresponding to that where its replicas are stored );
        }
        }
Step 8   End;
```

d. Algorithm for accessing data:

```
Step 1   Request comes from client side;
Step 2   Find in what range query comes into;
Step 3   Then request goes to “ Index table ” ;
Step 4   While(Parse the range)
        {
Step 5     If( found)
        {
Step 6       Find the key and check which replica is working or not;
Step 7       while (parse each replicas)
        {
Step 8         If(working)
        {
Step 9           Then goes that node and access the data;
        }
Step 10        Else
        {
Step 11         Continue;
        }
        }
        }
}
```


}
Step 12 End;

e. How it works:

Whenever the client request for accessing the data, first its token range is define by using MD5 hash algorithm. Then process goes to index table to handle the query. From index table the range will match and afterward key will identify according to the query. After identifying the key, now its turn to search where its replicas present. In index table, corresponding to key there is another column which store name of replicas node. From where we get which node contain replica of that key as well as which replica nodes are working perfectly and which are not by using the hinted handoff writes. Then directly control goes to the replica node and finally has access to that data. It maintain the scalability also like as soon as new node will add, one new column will also add to index table and store data corresponding to the new node.

For example:

A, B, C, D, E, F are nodes. Replicas of A stores in C & E, B stores in D & F, C stores in A & E and so on. Now let us assume that we want to access the data of A node and suddenly due to network problem A, B and C node fails. In this situation, if we are taking current methodology then according to that we can't access to data of A node as there is only three level of access is define. But according to our proposed implementation or solution, we can take help from index table to know more information about replicas node of A and from there we come to know that E node also contain the replica of A node. Finally data can be accessed without any traffic and delay.

SCOPE OF THE STUDY

In recent years there has been a shift in the way data is being stored with the explosion in the usage of new devices due to which data is being generated in bulk. Since these data are important source of information. Thus there's a need of new technique by which we can handle large amount of data in a most efficient manner. It need of emergence of new technique that could combine more approaches to resolve the problems that occur in handling bulk of data.

There is a big scope in field of combining these two technologies. Already research is going on integration of Cassandra and Hadoop that can be results in remarkable performance improvement for organizations working at that level. Combining these two gives big data tools and processing power to manage large data sets effectively. Multiple copies of data blocks are created and are distributed on the nodes of the computer through a cluster to be able to enable computations which are reliable. By integration provides distributed processing and Cassandra provides availability, performance and scalability. Both together may provide effective tool to manage the big data.

CONCLUSION

Cassandra and Hadoop are open source of distributed database management system. Both have an ability to handle the big data in different manner. Cassandra is scalable and gives high performance without any failure. But if its two or more neighboring node fails then it leads to poor performance and delay in data access, so if we store replicas into different node instead of neighboring node, it work well even if network problem occur as well as two or more neighboring node fails. Cassandra overcomes the problem of single point of failure in Hadoop but Hadoop has its own good feature with which Cassandra cannot compete. So if we remove the problem in Hadoop by increasing the number of name node in a single cluster that decrease the workload as well as providing Avatar node to every node which has all the information and increase the number of Job tracker maximum up to 2 level so that if there any problem occur than we don't need to start it from beginning which is a big advantage for Hadoop over Cassandra.

REFERENCE

Article:

[1]http://en.wikipedia.org/wiki/Big_data

[2]http://www.sas.com/en_us/insights/big-data/what-is-big-data.html

[3]http://www.explainingcomputers.com/big_data.html

[4] Eight (No, Nine!) Problems with big data by Gary Marcus and Ernest Davis April 6, 2014
http://www.nytimes.com/2014/04/07/opinion/eight-no-nine-problems-with-big-data.html?_r=1

[5] <http://www.datastax.com/dev/blog/cassandra-file-system-design>

Paper:

[1] Cassandra File System over Hadoop Distributed File System by Ashish A. Muth and Vaishali M. Deshmukh, March 2014.

[2] A Decentralized Structured Storage System (2009) by Avinash Lakshman and Prashant Malik
Cassandra.

[3] Advancing big data for humanitarian needs (2014) by: - Samson Oluwaseun Fadiya, Serdar Saydam, Vanduhe Vany Zira

[4] Algorithm and approaches to handle large data by Chanchalyadav, Shuliangwang, manojkumar 2013

[5] White paper of big data by leading researchers across the United States Feb 2012

[6] Cassandra File System over Hadoop Distributed File System By: - Mr. Ashish A. Mutha, Miss. Vaishali M. Deshmukh

[7] Enhancing NameNode Fault Tolerance in Hadoop Distributed File System. By: - Ohnmar Aung University of Computer Studies Yangon, Myanmar Thandar Thein University of Computer Studies, Yangon, Myanmar

[8] Perspectives on Big Data and Big Data Analytics by: - Elena Geanina ULARU, Florina Camelia PUICAN, Anca APOSTU, Manole VELICANU.

[9] From Databases to Big Data by: - Sam Madden • Massachusetts Institute of Technology.

[10] White Paper by Data Stax Corporation July 2013.