ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 7, July 2015

The Big Data Recovery System for Hadoop Cluster

V. S. Karwande[1], Dr. S. S. Lomte[2], R. A. Auti[3]

ME Student, Computer Science and Engineering, EESCOE&T, Aurangabad, India[1]

Professor, Computer Science and Engineering, EESCOE&T, Aurangabad, India[2]

Assistant Professor, Computer Science and Engineering, EESCOE&T, Aurangabad, India[3]

Abstract: Due to brisk growth of Data Storage in Many internet Service Companies, there is always issue of regarding unstructured data storage which is generated in Terabytes [TB] and Peta bytes [PB]. Hadoop is always deal with the large amount of data Volume. Therefore increase reliability and availability should be maintained. To gain the high availability characteristic of the Hadoop and to improve failure Recovery as early as possible or failure should be avoided. The failure of the HDFS, Name node and Master Node affects the performance of the Hadoop cluster. To overcome this problem, we proposed a system which will select new recovery namenode with less amount of time which will replicate data from namenode. In this paper, we analyze behaviour of the namenode with respect to its failure and recovery from failure.

Keywords: Cloud Computing, Fault Tolerance, Hadoop, HDFS, Recovery.

I. INTRODUCTION

Cloud computing provides dynamically scalable and often virtualized resources as a services over the Internet [1]. Cloud computing has become significant technology trend in industry and in academic as well. Cloud computing is a paradigm that focuses on sharing data and computations over a scalable network of nodes. Hadoop is an open source java framework, workhorse for the cloud computing. Large data volume can be operated by Hadoop with hadoop distributed file system and map reduce.

Hadoop gives the robust Hadoop Distributed File System (HDFS) and a Java-based API that allows parallel computing across several nodes of the cluster [2]. HDFS architecture is a master-slave architecture. Namenode is acts as a master and datanodes is act as a slaves. Namenode manages the file-system metadata. HDFS stores the file-system as a differ series of the blocks, random size of each block is by default allocated 64MB [3]. Blocks are stored on file-system of datanodes. The mapping between block and datanodes stores in the namenode's memory. Datanodes sends periodic block report to namenode. All working datanodes sends heartbeats to the namenode every 3 seconds. Heartbeat is an instant message which shows the availability of datanode as well as namenode [3]. Blocks are replicated on the several datanodes to retrieve data from failure of particular datanode. Block replication depends upon the replication factor which is three [3].The Replica placement policy generally used to improve HDFS Data availability, reliability and utilization of Network bandwidth. Suppose replication factor is three then HDFS Data placement policy is to put one replica any one random node in rack, remaining two replica is place in two different rack [3].

Fig. 1 shows the architecture of Hadoop. Secondary namenode stores latest image and log files. Datanode can't possible connecting to secondary namenode.

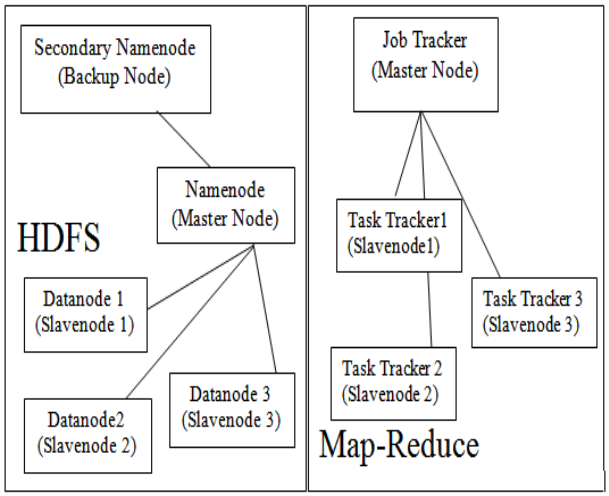

Figure1.1. Hadoop Architecture in Multi-node Cluster [3][14].

Secondary namenode is incapable of updating itself with the help of datanodes. To reduce the time overhead of restarting cluster recovery system should be introduced.

The rest of paper is as organized as follows. Section II provides brief information about HDFS architecture as well as it analyses the behavior of HDFS under failures. The proposed scenario of the system is explained in the section III.

The architecture of proposed scenario is described in Section IV while Section V is the conclusion of our proposed scenario.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 7, July 2015

II. RELATED WORK

A. *HDFS Architecture*

The HDFS is a distributed file system architecture basically designed to stores metadata and application data in different locations. HDFS is built around the ideas that highly reduce fault-tolerant & the most efficient data processing pattern is readmany times but Write only once times paradigm. A dataset is normally produced, copied or generated from source file system and then various analysis are performed on that dataset over time. A typically file size in HDFS is Tera Bytes and Gega Bytes, Thus HDFS is generated large Data sets. HDFS is optimized for delivering a high-throughput of data and this may be at the expense of latency. In HDFS files are written to by a single writer. There is not providing support for multiple writers or for perform modifications at arbitrary offsets in the file.

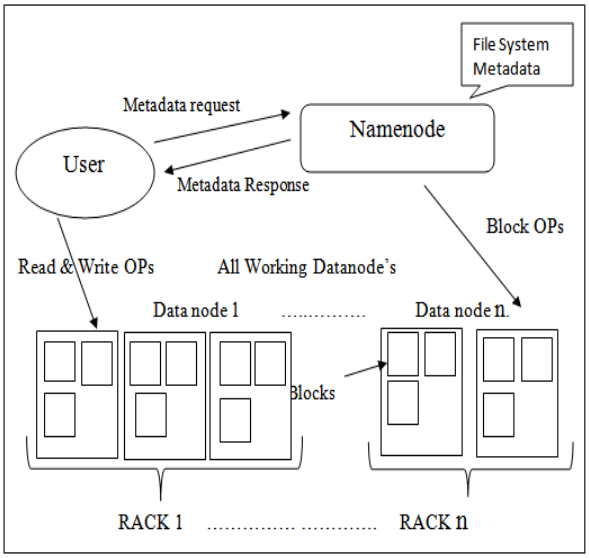

Figure 2.1: Existing HDFS Architecture in Multi-node Cluster. [3]

In the Hadoop Cluster is basically using Client and Server model, all Servers and Clients are connected with each other and communicate with each other in same network. Also based on TCP-IP Protocol suite. The Working Datanodes inHDFS don't depend on with full trust or confidence on data security mechanisms such as RAID to make the data durable. Instead like Google file System is replicated on multiple datanode for reliability. While ensuring database system durability, this strategy has provided different advantage that data transfer bandwidth is multiplied and there are many opportunities for locating computational near the needed data.

The Hadoop Distributed File System (HDFS) is generated large amount of data sets and to transmit or receive those data sets at very high bandwidth to end users application programing. In a Hadoop cluster, many servers host directly attached storage and execute user application tasks. By dividing storage of blocks and computation across many servers, the resource can grow with demand and remain scalable. The architecture of HDFS is also using to manage 40 petabytes of enterprise data is described at Yahoo!

B. *Failure in HDFS*

HDFS architecture is mainly based on namenode and datanode, where namenode act as a master while datanode act as a slave. If datanode fails then namenode will divert the work of failed datanode to other available datanode [6][9]. However, the failure of a name node affects reliability of metadata.

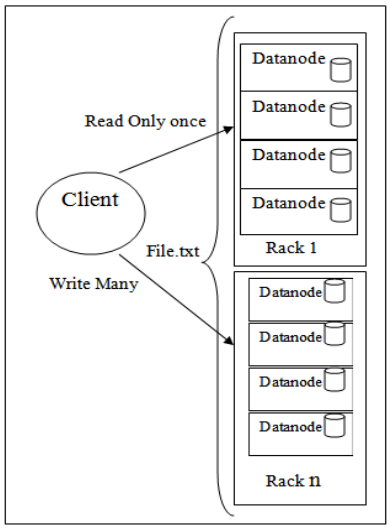

Figure 2.2: Hadoop Cluster File storage.

To avoid this, HDFS architecture selects a secondary namenode which will work after the primary namenode fails. Fig. 1 shows the architecture of Hadoop with a secondary namenode. Secondary namenode is used for copying for latest image and logs. Datanode can't connect to secondary namenode. Secondary namenode is incapable of updating itself with the help of datanodes. To reduce the time overhead of restarting cluster recovery system should be introduced.

III. PROPOSED WORK

Namenode failure affects the performance of Hadoop cluster. As stated earlier namenode is a pillar of HDFS architecture and which contains metadata of the data which are stored on different datanodes. To overcome failure, we propose recovery system, replicate the entire namenode on the other datanode called as *Recovery Namenode*. Recovery namenode will update all the information of namenode simultaneously. After failure, recovery namenode will act as namenode [17][20][19].

Recovery namenode keeps track of corresponding namenode. After a periodic time interval, recovery namenodeis updated. Initially all the datanodes sends Instant Messages to corresponding namenode, when namenode gets down recovery namenode, itself broadcasts a message to all datanodes about new namenode. Now, they have to send Instant message to corresponding recovery namenode.

The detailed description of the proposed recovery system is given in the architecture section. It selects new namenode from available datanodes.

DOI 10.17148/IJARCCE.2015.4792

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 7, July 2015

IV. ARCHITECTURE

The Behaviour of the HDFS architecture is shown in the figure 4.1.There are two cases; the first one gives the architecture of the new HDFS before failure while the second one gives the architecture of new HDFS after failure with new Namenode

A. *Selection of Recovery Namenode*

This section describes appropriate methods for the efficient selection of recovery nodes such that the performance of the cluster is not adversely affected. Active nodes are considered for the selection of recovery namenode.

Here, 'N' is the namenode while '$D1, D2 \ldots, D_n$' are the datanodes. From these datanodes our proposed recovery system selects one node as a recovery namenode. As stated, after every three seconds datanode sends a heartbeat to Namenode to show his availability.

In our scenario, each datanode sends a Instant message along with its time of generation i.e. T_{dt}. Every datanode has variant time of Instant message generation such as T_{dt1},T_{dt2},.....,T_{dtn} for 'n' nodes. The Instant messages of each datanode are stored in a log referred *Instant message beat Log* along with arrival time T_{at} of corresponding datanode in the namenode.

Now, the time taken to reach namenode from a datanode is given by

$$TT_{dt} = T_{at} - T_{dt}$$

Where, TT_{dt} =actual time taken by a Instant message from datanode to namenode. For every datanode we are considering first 'x' readings.

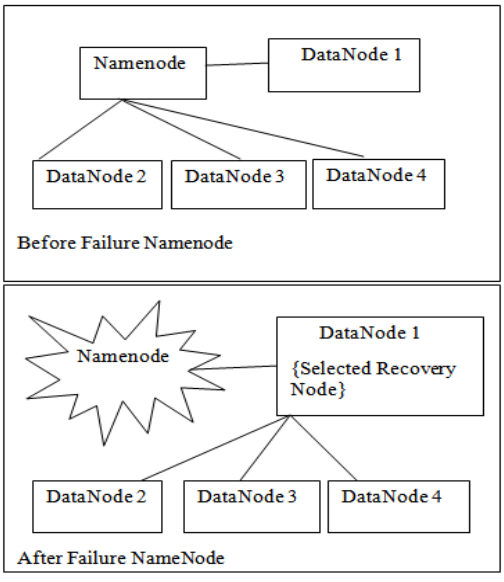

Figure 4.1: Proposed HDFS Architecture in Multi-node Cluster.

The meantime to reach a namenode from a datanode is given by

$$mT_{dt_n} = \frac{1}{x}\sum_{i=1}^{x} TT_{dt}$$

We denote the log of all datanodes with their respective mean time 'mT_{dt}'. The sorted log is stored as list called *recovery namenode list*. According to this *recovery namenode list*, the first node from the list is selected as recovery namenode.

B. *Create Recovery Namenode List*

If recovery namenode gets down before failure of the namenode, the system became unreliable. A recovery namenode list is created for every recovery namenode, to increase the reliability of the system.

An algorithm first checks Instant message response of a datanode. Active datanodes are added in recovery log list while calculating mean response time. New recovery namenode is selected from new recovery namenode list.

C. *Communication between Namenode and Recovery Namenode*

After selection of recovery namenode, the communication between namenode and recovery namenode is an important factor. There will be an instant messaging from namenode to recovery namenode. As the datanode generates Instant message after three second, namenode follows the same scenario and will generate an instant message and send it to the recovery namenode to announce that it is alive.

Recovery namenode assumes the role of the namenode if it does not receive a message from the corresponding namenode in the next 600 seconds, and broadcasts a message to all the datanodes to announce the role change.

D. *Set a Checkpoint*

Checkpoint method is widely used in different recovery models [10][19]. It allows system to recover from unpredictable fault. The idea behind this system is the saving and restoration of the system state.

Here, checkpoints are nothing but a time interval which is periodic. To replicate namenode on recovery namenode, atime interval is considered. On a certain time interval checkpoints are created. We have set this time interval to 300 seconds. After every 300s namenode is replicated on recovery namenode. It means checkpoints are created after 300s. Checkpoints are sets only for namenode. Creating periodic checkpoints is the way to protect the metadata of file system.

E. *Availability of Namenode*

As per the HDFS architecture, namenode does share any information about his failure, to overcome this problem in our scenario; namenode will generate an instant message, sends to recovery namenode after three second to ensure his availability. After namenode failed to send instant message up to 600s then namenode will be declared as dead node.

After that recovery namenode sends a message to all datanode, to announce the change of status. Recovery namenode start working from the last checkpoint, before failure of the namenode.

DOI 10.17148/IJARCCE.2015.4792

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 7, July 2015

F. *Failure of Recovery Namenode*

After the failure of a namenode, the corresponding recovery node assumes the responsibility of the namenode.After a recovery namenode becomes a namenode, a newrecovery namenode is selected by using same parameters.

According to available datanodes new recovery namenode list is generated in a similar way. Despite increased overheads on namenode and datanode, the scheme provides high availability.

G. *Algorithm for Create Recovery Namenode List.*

```
To Create Recovery Namenode List

create_recovery_namenode_list()
{
get_all_node_list();

while (list is not empty)
{
if(node_IM_response time == TRUE)
{
node_mean_time=calculate_IM_mean_time(node);
add_node_to_recovery_namenode_list(node,node_mean_time);
goto_next_node;
}
else
{
goto_next_node;
}
}
```

Figure 4.2:Algorithm for Create Recovery Namenode List.

H. *Algorithm for Selection Recovery NameNode.*

```
To Select Recovery Namenode

calculate_IM_mean_time(node)
{
  total_travelling_time=0;
for (from starting time up to a certain time)
{
IM_start_time[]= get_start_time_time();
IM_recieved_time[]= get_received_time();
IM_traveeling_time=IM_recieved_time[]-IM _start_time[];
total_travelling_time=total_travelling_time+
IM_traveeling_time;
            }
node_mean_time= total_travelling_time / (certain_time –
starting_time);
return  node_mean_time;
}
quick_sort (recovery_namenode_list with respect to
node_mean_time);
```

Figure 4.3: Algorithm for Selection of Recovery Namenode.

V. RESULT

Hadoop cluster Namenode is working condition after long time Namenode is remains down, then data recovery performance is calculated both existing and Recovery system.

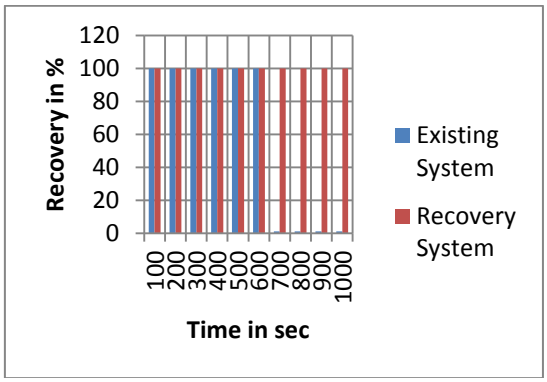

Figure 5.1: Performance of the System

The Figure 5.1 graph shown by After 600 second Existing system is not possible to Data Recovery because exiting system is needed to restart hadoop cluster manually and performance result of Recovery system is after 600 second is also possible to data recovery. Because the recovery system not need to restart. If instant message is not received then after 600 second Namenode is declared as dead. and after 600 second automatically selected recovery name node as Namenode.

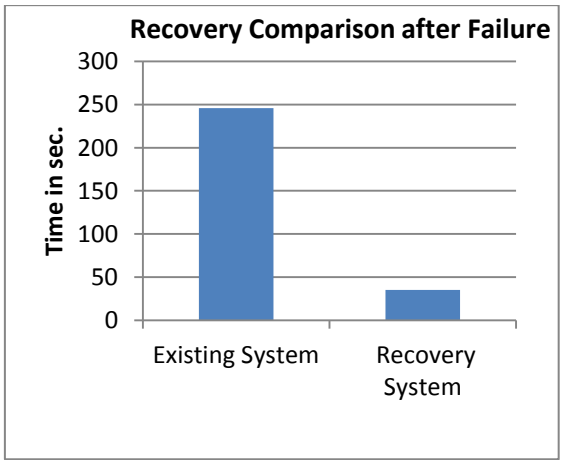

Figure 5.2: Recovery time after failure

For a long time if Namenode remains down, then cluster should be restarted manually. Then select node as Namenode and then start respective jobs.

But in Recovery system, Recovery Namenode will be the new Namenode if Namenode gets down for a long time. After some time Namenode gets up then old Namenode will be continue. But if it is down permanently, then no need to restart cluster. The Figure 5.2 graph shown compares the time require for the system to get up after occurrence of failure. Recovery System requires less time to get up than existing system. The Recovery system is required to up recovery node time is 48 second. And the Existing system is need to restart cluster time is 250 second.

VI. CONCLUSION

In cloud computing, unstructured data storage is popular issue. Hadoop deals with unstructured data

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 7, July 2015

storage. In this paper, we have studied and analysis the architecture of the Hadoop Distributed File System under namenode failure. To overcome namenode failure in HDFS we have proposed an architecture which increases reliability as well as availability of Hadoop. We also focused on selection of recovery namenode after failure of namenode.

REFERENCES

[1] Florin Dinu, T. S. Eugene Ng" *Understanding the Effects and implications of compute Node Related Failure in Hadoop"*, HPDC'12, Delft, The Netherlands, (June 18-22, 2012.).
[2] Jeffrey Shafer, Scott Rixner, and Alan L. Cox, TheHadoop Distributed Filesystem:Balancing Portabilityand Performance, Presentation, ISPASS 2010, March30th 2010.
[3] Dhruba Borthakur.2007. [E-book]. "*The Hadoop Distributed File System: Architecture and Design*". Available through:<http:/ hadoop.apache.org/common/docs/r0.18.0/hdfs _design.pdf>
[4] Ronald Taylor, Pacific Northwest National Laboratory, Richland, WA, "*An* overview *of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics"*, Bioinformatics Open Source Conference.2010doi:10.1186/1471-2105-11-S12-S1.
[5] Konstantin Shvachko, HairongKuang, Sanjay Radia, Robert Chansler[E-book]. "*The Hadoop Distributed File System*"2010.
[6] MohommadAsif Khan, Zulfiqar A. Menon, Sajid Khan"*Highly Available Hadoop Namenode Architecture*," International Conference on Advanced Computer Science Applications and Technologies, 2012.
[7] AsafCidon, Stephen Rumble, Ryan Stutsman,SachinKatti, John Ousterhout and MendalRosenblum, "*Copysets: Reducing the Frequency of Data Loss in Cloud Storage*" SAN JOSE CA ,2013 USENIX Conference.
[8] FarazFaghri, SobirBazabayev, Mark Overholt, Reza Farivar, Roy H. Campbell and William H. Sanders, "Failure Scenario as a Service (FSaaS) for HadoopCluster", SDMCMM'12, Montreal, Quebec, Canada,December 3-4,2012.
[9] Florin Dinu, T. S. Eugene Ng, "*Analysis of Hadoop's Performance under Failures*", Rice University.
[10] Jorge-Arnulfo Quiane-Ruiz, Christoph Pinkel, JorgSchad, Jens Dottrich, RAFT at Work: Speeding-UpMapReduce Applications under Task and NodeFailures, SIGMOD'11, Athence, Greece, June 12-16,2011.
[11] Cod-E-mphasis 2012. " *Big data-hadoop HDFS and MapReduce*". Available at: <http://codemphasis.wordpress.com/2012/09/27/big-data-hadoop-hdfs-and-mapreduce//.>.
[12] HadoopGettingStartedhttp://docs.hortonworks.com/HDPDocuments/HDP1/HDP-Win-1.1/bk_getting-startedguide/content/ch_hdp1_getting_started _ chp3.html.
[13] Brad Hedlund.."*Understanding hadoop clusters and the network*" .Available at:<http://bradhedlund.com/2011/09/10/understanding-hadoop-clusters-and-the-network/>".2011.
[14] Hadoop in Practice http://techannotation.wordpress.com/2012/09/10/hadoop-inpractice/
[15] The Building Blocks of Hadoop http://pramodgampa.blogspot.sg/2013/06/the-buildingblocks-of-hadoop.html.
[16] Jin San Kong , Min Ja Kim, Wan Yeon Lee, Chuck Yoo and Young Woong Ko.,"*Multi-level Metadata Management Scheme for Cloud Storage System* ", International Journal of Multimedia and Ubiquitous Engineering Vol.9, No.1 (2014), pp.231-240, 2014.
[17] Arvind K. Bansal, Kotagiri Ramohanarao, Anand Rao, "distributed multiple agent systems".1997
[18] Kyle Chard and Kris Bubendorfer, " *High Performance Resource Allocation Strategies for Computational Economies* " , IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 24, NO. 1, JANUARY 2013
[19] Bahman Javadi, Derrick Kondo, Jean-Marc Vincent and David P. Anderson, "*Discovering Statistical Models of Availability in Large Distributed Systems: An Empirical Study of SETI@home*"IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 22, NO. 11, NOVEMBER 2011.
[20] Samir Jafar, Axel Krings, and Thierry Gautier, " *Flexible Rollback Recovery in Dynamic Heterogeneous Grid Computing* " IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. X, NO. Y, JANUARY 2008.