International Journal of Engineering & Technology, 7 (1.1) (2018) 311-313

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Hadoop high availability through multiple active name nodes

P. Vijaya Lakshmi [1], K.V.S Ramesh [2] *, P. Likhitha [2], M. Pranay Kumar [2]

[1] *Assistant professor, Department of Computer Science and Technology, Koneru Lakshmaiah Education Foundation*
[2] *Department of Computer Science and Technology, Koneru Lakshmaiah Education Foundation*
**Corresponding author E-mail: Rameshkadiyala143@gmail.com*

Abstract

HDFS having only single dynamic name node, if that name node occur hardware or software failure, the entire HDFS model will be inactive position until the recovery of name node. So that to reduce that problem the standby name nodes are placed, which they are an inactive position. On failover occur to primary name node all its metadata will transfer to the standby name nodes. After primary name node fails remaining standby nodes elects one of the nodes to take the position of primary name node. But on transferring the metadata to remaining standby name nodes there will be heavy burden to the primary name node

In this paper, we proposed a solution to reduce the load on the primary name node by transferring the metadata to remaining standby name nodes. We compress the entire metadata in the primary name node and sent that data into remaining all standby name nodes.

Keywords: *Name Node; Standby Nodes; Scalability; Availability; Hot Standby.*

1. Introduction

Hadoop Distributed File System (HDFS) is the basic storage system utilized by Hadoop applications. HDFS [5] is to distribute the files that provide the high-performance [4] access to data in the entire hadoop cluster. The HDFS has low cost hardware, so a bit failures in server are common in the cluster.

1.1. Name node

If Client needs to store data, then the client sends that information to name node. The client's duty is to read/write the data from data nodes. If the data be large file then the client divide that large file into blocks of 64MB or 128MB. The client sends a request to the name node to assign the 64MB or 128MB block with replication of three to write the data in it directly. Then name node [4] will assign the data node addresses to the client to store each block of data in multiple locations (Usually 3). That means the name node will act as a mediator between client and data nodes and also the name node will coordinate everything around the cluster. Then the client will send the block of information to the one data node. If data is successfully stored in data nodes at that time the name node sent the acknowledgement to the client that the data be successfully delivered. And also the name node [4] will act as a server that addresses the data in the data node.

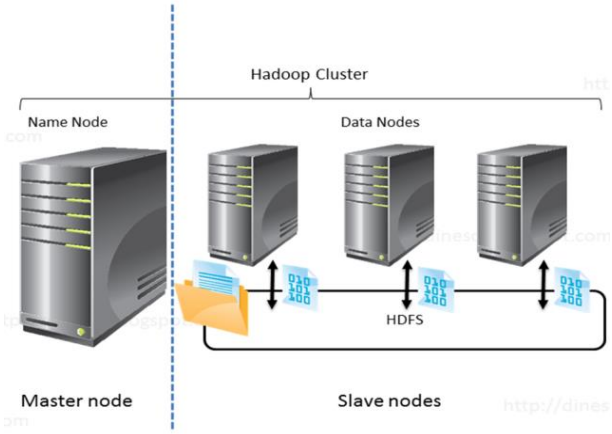

Fig. 1: Master Name Node.

1.2. Data nodes

The client writes data in one data node, then that data node will send the same data to the second data node. After receiving data from the first node it forwards same data to next data node. Because it is easy to access data and also if the data be lost or corrupted then the client will retrieve data from remaining data nodes. As shown in figure 2, this type of process is called Replication Factor. And the data nodes will also occur fault tolerance [3] to reduce that problem data node will send a heartbeat message to name node for every 3sec this is the way of saying that the data node is still in active position.

If the heartbeat message comes, then the name node will identify that data nodes are an inactive position, if not the data nodes are inactive position then name node will wait 10 minutes if any heartbeat message will come. At that time also the name node doesn't get the heartbeat message then it think that the data node is dead the data node will be alive and there was only a network

Copyright © 2018 P. Vijaya Lakshmi et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

failure, but the name node will treat both as same. It cuts the connection between that particular data node.

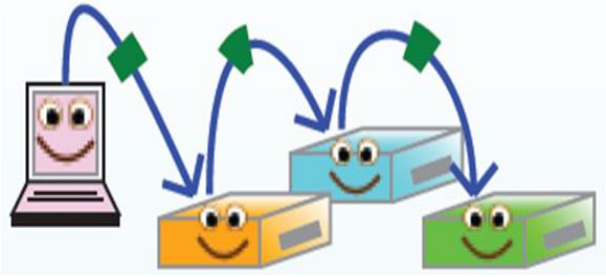

Fig. 2: Data Nodes.

2. Secondary name node

In HDFS, there are 2 name nodes Primary Name nodes [10] and Secondary Name node. Primary Name node [4] acts as a mediator between Client and Data node's and it shows the client where the data present in data nodes. And all information about data is present in name node. So, it is easy to access data or retrieve data. Data node is nothing but accessing data from the client in the presence of name node. If the primary name node is dead the entire cluster is dead. Name node is only the point of failure [3] that means if it fails the whole cluster will stop working. So, here if primary name node is dead the entire cluster be in an inactive position at that time it is difficult for the client to retrieve data from data node.

To reduce this problem there is a Secondary name node will be there but how to store all data addresses to secondary name node. To know that we need to know what operation be done in primary name node. So first, we need to know about the metadata. Metadata [2] data is a set of data that elucidate and gives information about another data and there are two files related to metadata is 1) FsImage 2) Edit Log.

File system Image is nothing but an image of the file system on initial point of the name node. Whereas the Edit Log is the modifications made to the file system after starting the name node. The primary name node it manages all the requests for write and a part of requests for read from HDFS [5] customer and the remaining of name nodes called hot standby and they are all read-only.

Secondary name node is the standby name node which is not in dynamic position. Since it doesn't react to any requests of write or read all it requires is to do regularly fetch File system image also to Edit Log from primary name node and combine them into a new File system image. At the point, when name hub runs a particular measure of period and changes recorded in Edit Log that would be converged with File system image, so that name node could have current file HDFS metadata [2]. And after combining, the changes from Edit Log to file system Image, name-node deletes the previous File system Image copy and replaces it with a more up to date one as it has new File system Image.

Which points to the present state of HDFS and then it shows up the newer Edit Log. So, when failover occurs Secondary name node read metadata from File system image and Edit Log from its local disk, then take the responsibility of initial name node and react to requests from HDFS users.

3. Cluster with multiple active name nodes

As Shown in figure 3, the Client has shared so much of data to data nodes in the presence of name node if that name node occurs the failover then the entire data will be corrupt. In order to reduce that secondary name node will present.

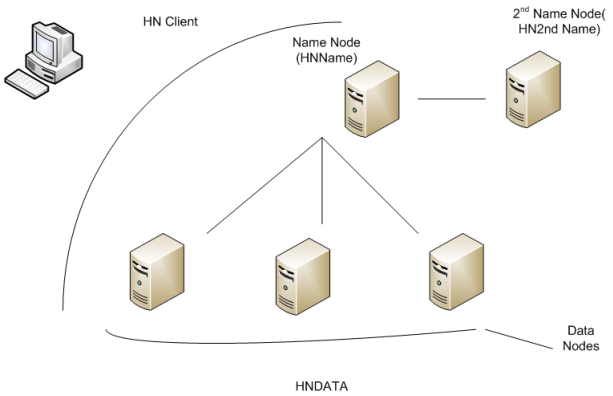

Fig. 3: Secondary Name Node.

4. Existing system

The cluster utilizes single writer multiple reader procedures. One name node called primary name node, it manages entire the requests of write and a bit of read requests from HDFS customers, the remaining name nodes are called hot standby name node, they are all read-only, which implies they could just serve the read requests.

The burden is that the standby name node is not dynamic. In existing system all name nodes are active. The primary name node sends all its metadata [2] to remaining standby name node. The primary name node accepts both read and write requests so that it writes the metadata from its local disk and writes to all standby name nodes local disk. This process is called metadata replication [2].

The failover happens only when initial name node is down. The hot standby name nodes are in charge of requests of read, so that failing of those will not affect the entire HDFS.

The 2 stages that failover will occur are IP address and leader election progress [1]. IP address progress is generally simple in N-cluster. As the primary name node of HDFS is accessed through IP address. At the point when a hot standby is elected and takes role as the new primary node, it changes its IP address to the old primary node IP address, so that it can take over all communications with other nodes. I.e. hot standbys and data nodes [14].

Leader election [1] will be simple hence all metadata over each name nodes are indistinguishable. At the point when the primary name node meets failures, it is possible to pick any another hot standby arbitrarily as the new primary node. The details of leader election, assume N-Cluster contains 'N' hot standby name nodes, Firstly we assign an increasing sequence of number to each of the hot standbys, say 1 to N when hot standbys trust the essential is out of work. After that, the standby name nodes believe that the primary name node is out of work i.e. they have not received the acknowledgement of their heartbeat from primary name node for a quite a while. The hot standby with the most elevated number convey a message to check which hot standby name node has the most recent metadata, [2] at that point it sends this name node's number to the all of the standby to guarantee that each hot standby can understand where to synchronize the most current metadata, when the synchronization procedure is done the hot standby with the most elevated number transforms into the new primary node and takes charge of write request.

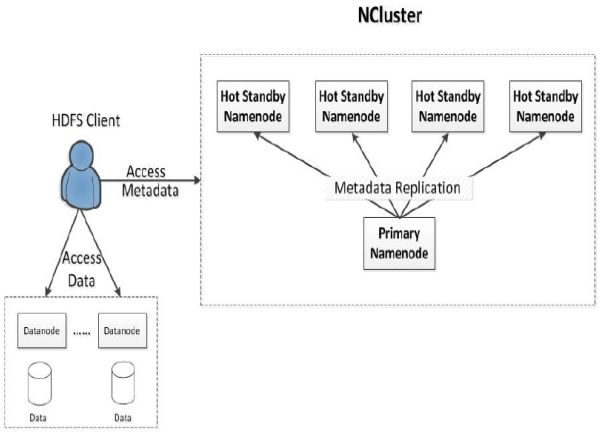

Fig. 4: N-Cluster.

5. Proposed system

We proposed a system that the primary name nodes have a backup name node. If the primary name node writes its metadata [2] to the backup name nodes then there will be a heavy load on the primary name node. Because it takes the portion of both read and write request and also it sends its metadata to standby name nodes so that there will be a load on it. To reduce that load on the primary name node we compress the metadata in the primary name node and writes that compressed data into standby name nodes so that the load will be reduced in primary name node [4]. If any modification is done by the client then the primary name node will send that modification to the remaining name node so that all the name nodes are up-to-date.

After that, the primary name node will not in active position then the standby name nodes which having highest number will take charge of the primary name node and the new primary node will send that information to all remaining standby name nodes.

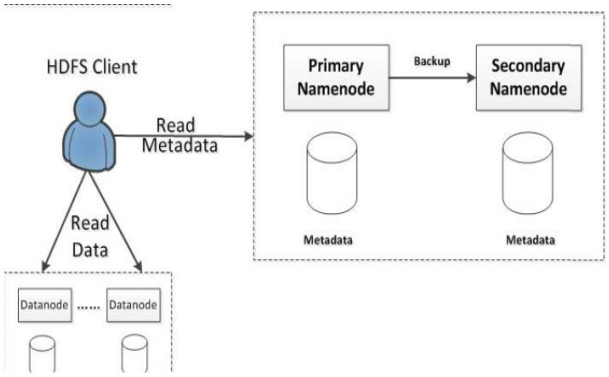

Fig. 5: Master Node with Slave Node.

6. Conclusion

In this paper, the name node writes its metadata[2] into all standby name nodes so that there will be a load on the primary name node, to reduce that load in primary name node we proposed a model to have the high accessibility for name node through load balancing. We proposed a solution that we compress the entire metadata in the primary name node and sent that data into remaining all standby name nodes [1]. On doing this process the name node will reduce the load on it. So that the cluster works efficiently and good throughput [1].

Acknowledgement

We would like to thankful Mrs P. Vijaya Lakshmi for making our review paper to be completed in time with her valuable suggestions.

References

[1] Wang, Z. and Wang, D., 2013, November. NCluster: Using Multiple Active Name Nodes to Achieve High Availability for HDFS. In *High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on* (pp. 2291-2297). IEEE.

[2] Wang, F., Qiu, J., Yang, J., Dong, B., Li, X. and Li, Y., 2009, November. Hadoop high availability through metadata replication. In *Proceedings of the first international workshop on Cloud data management* (pp. 37-44). ACM. https://doi.org/10.1145/1651263.1651271.

[3] Varghese, Lino Abraham, V. P. Sreejith, and S. Bose. "Enhancing NameNode fault tolerance in Hadoop over cloud environment." In *Advanced Computing (ICoAC), 2014 Sixth International Conference on*, pp. 82-85. IEEE, 2014.

[4] Khan, Mohammad Asif, Zulfiqar A. Memon, and Sajid Khan. "Highly Available Hadoop NameNode Architecture." In *Advanced Computer Science Applications and Technologies (ACSAT), 2012 International Conference on*, pp. 167-172. IEEE, 2012. https://doi.org/10.1109/ACSAT.2012.52.

[5] Foley, Matt. "High availability HDFS." In *28th IEEE Conference on Massive Data Storage, MSST*, vol. 12. 2012.

[6] Wan, J., Liu, M., Hu, X., Ren, Z., Zhang, J., Shi, W. and Wu, W., 2012, December. Dual-JT: Toward the high availability of JobTracker in Hadoop. In *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on* (pp. 263-268). IEEE.

[7] Oriani, A. and Garcia, I.C., 2012, October. From backup to hot standby: High availability for hdfs. In *Reliable Distributed Systems (SRDS), 2012 IEEE 31st Symposium on* (pp. 131-140). IEEE.

[8] Aung, Ohnmar, and Thandar Thein. "Enhancing NameNode Fault Tolerance in Hadoop Distributed File System." *International Journal of Computer Applications* 87, no. 12 (2014). https://doi.org/10.5120/15264-4020.

[9] Kim, Y., Araragi, T., Nakamura, J. and Masuzawa, T., 2014, October. A Distributed NameNode Cluster for a Highly-Available Hadoop Distributed File System. In *Reliable Distributed Systems (SRDS), 2014 IEEE 33rd International Symposium on* (pp. 333-334). IEEE. https://doi.org/10.1109/SRDS.2014.61.

[10] Bi, Kun, and Dezhi Han. "Scalable Multiple NameNodes Hadoop Cloud Storage System." *International Journal of Database Theory and Application* 8, no. 1 (2015): 105-110. https://doi.org/10.14257/ijdta.2015.8.1.12.

[11] Devi, S., and K. Kamaraj. "Architecture for Hadoop Distributed File Systems." *Architecture* 3, no. 10 (2014).

[12] Le, Hieu Hanh, Satoshi Hikida, and Haruo Yokota. "NameNode and DataNode Coupling for a Power-proportional Hadoop." *Book name Database Systems for Advanced Applications Lecture Notes in* (2013).

[13] Shvachko, Konstantin, Hairong Kuang, Sanjay Radia, and Robert Chansler. "The hadoop distributed file system." In *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on*, pp. 1-10. IEEE, 2010. https://doi.org/10.1109/MSST.2010.5496972.

[14] Oriani, Andre, and Islene C. Garcia. "From backup to hot standby: High availability for hdfs." In *Reliable Distributed Systems (SRDS), 2012 IEEE 31st Symposium on*, pp. 131-140. IEEE, 2012. https://doi.org/10.1109/SRDS.2012.33.

[15] Donvito, Giacinto, Giovanni Marzulli, and Domenico Diacono. "Testing of several distributed file-systems (HDFS, Ceph and GlusterFS) for supporting the HEP experiments analysis." In *Journal of Physics: Conference Series*, vol. 513, no. 4, p. 042014. IOP Publishing, 2014.