I.J. Information Engineering and Electronic Business, 2017, 6, 27-34
Published Online November 2017 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijieeb.2017.06.04

A Proposal for High Availability of HDFS Architecture based on Threshold Limit and Saturation Limit of the Namenode

Sabyasachi Chakraborty, Kashyap Barua, Manjusha Pandey and Siddharth Rautaray
School of Computer Engineering, KIIT University, Bhubaneswar, India
Email: c.sabyasachi99@gmail.com, kashyapbarua@gmail.com, manjushafcs@kiit.ac.in, siddharthfcs@kiit.ac.in

Received: 02 June 2017; Accepted: 07 August 2017; Published: 08 November 2017

***Abstract*—**Big Data which is one of the newest technologies in the present field of science and technology has created an enormous drift of technology to a salient data architecture. The next thing that comes right after big data is Hadoop which has motivated the complete Big Data Environment to its jurisdiction and has reinforced the complete storage and analysis of big data. This paper discusses a hierarchical architecture of Hadoop Nodes namely Namenodes and Datanodes for maintaining a High Availability Hadoop Distributed File System. The High Availability Hadoop Distributed File System architecture establishes itself onto the two fundamental model of Hadoop that is Master-Slave Architecture and elimination of single point node failure. The architecture will be of such utilization that there will be an optimum load on the data nodes and moreover there will be no loss of any data in comparison to the size of data.

***Index Terms*—**Hadoop, Namenode, Datanode, Big Data, Architecture

I. Introduction

Big Data, as we all know, is becoming a new technological trend in the industries, in science and even businesses. Our activities on a daily basis, be it interaction with machines, surfing websites and every other activity that we perform generate data, on a huge scale. These generated data come with responsibility for a proper administration, proper storage and processing mechanisms.

Numerous researches have been done and are still being done to make the storage and processing mechanism effective so that the end-users have a hassle-free experience with these technologies. One of the most popular frameworks that capacitate the motion of storing, managing and analyzing of the huge amount of data is Hadoop.

What and why Hadoop? Due to the open-source nature of Hadoop, it is one of the preferable tools for the task of storing an enormous amount of data and legitimate analysis of the data to generate effective results. Hadoop is a database which is used for processing large number and large size of datasets. It is based on the concept of Master-Slave architecture. The Master (Namenode), which is responsible for the storage of the metadata about the Slaves (Datanodes). Moreover, the Hadoop architecture also ensures the elimination of Single point Node failure as this can lead to huge amount of data loss.

Presently, the minimization of Single Point of Failure in Hadoop Distributed File System has been taken into consideration in the architecture by deploying Quorum Journal Manager that further deploys Journal Nodes across the system to maintain concurrency between the Namenodes. The Quorum Journal Manager connects with the Namenodes of the Architecture and tries to maintain the concurrency between the Namenodes by maintaining edit logs of the system. If at any point in time if an Namenode fails then another Namenode can simply take the place of the failed Namenode by fetching out the edit logs.

The Namenode of Hadoop contains the directory tree files of all the blocks of data that are distributed over the data nodes for the faster computational purpose, but if at any point of time the Namenode fails, then the complete metadata of the distribution of data files across the Datanodes will be lost. So for preventing such kind of failures, a separate metadata table, FSImage file and edits log file is maintained in the Secondary Namenode. However, the Secondary Namenode can never be considered as the replacement of Namenode because it only takes care of the metadata of the Namenode.

The computation of all the task that takes place in Hadoop is achieved by enforcing the Map-Reduce algorithms onto the data sets. Whenever data is stored in the Hadoop Distributed File System, the data is divided and also replicated across the Datanodes based on the block sizes and the replication factor which are either set up by the administrator or the default values that comes with Hadoop versions for the proper maintenance of data. Now when a Map-Reduce task is enforced upon the data for certain computations, then the task or the job is carried out parallelly across the Datanodes. This process of computation on large datasets that prevails in today's market, when performed parallelly across various Data Nodes gives results efficiently and accurately.

The High Availability Hadoop Distributed File System is an architecture that allows us to maintain the Hadoop Distributed File System in such a way that the data storage is never affected by the occurrence of any failures of the Data Nodes or the Namenodes. HAHDFS follows a hierarchical model of storing the metadata of the Datanodes that is connected to it. Moreover, the HAHDFS follows a load balancing method to check the load on a particular Namenode for the computation of certain task. However, at any instant of time if it is found that the load on a particular Namenode has reached the threshold limit, and the computation of certain task is querying for more resources, then the system activates more Namenodes based upon the computational requirements.

The Master Slave Architecture of Hadoop is a prevalent method of the Hadoop Distributed File System. The Hadoop Distributed File System ensures that the Master Node is kept with Data Nodes attached to it. The Master Node maintains the metadata table for the concurrency control between the data that is stored in the Data Nodes. Whenever a data is inserted in the Data Node, the Data Node basically breaks the data into tiny pieces for faster optimization. Also the Pieces of the data are then replicated into various Nodes for proper fetching during processing of the Data. The Replication of the Data in the Data Nodes are then maintained and kept in record in the metadata table that is maintained in the master node of the Hadoop Distributed File System. The Metadata table is always kept updated by the system for maintaining the proper concurrency between the data splits. The Splitting of the data is basically determined by the block size of the Hadoop Distributed File System as each data split account for the size of block. The default block size of the Hadoop Distributed File System is 64MB.

The Single Point of Node Failure of the system is addressed by keeping a Secondary Name in the system. The Secondary Namenode maintains the metadata table. The Secondary Namenode comes into effect when the Master Namenode either fails or gets disconnected to the system. The Secondary Namenode does not act as the Master Namenode but acts as a backup for the Master Namenode to prevent the meta data to get lost. The Secondary Namenode previously had to be manually activated by the Administrator of the system, but presently it can be activated automatically by the Zookeeper whenever the Master Namenode fails.

II. State of the Art

In the present world, each and every second holds the generation of the enormous amount of data. The storage of such large and useful data is one of the most primary tasks, and also, the effective analysis of such huge amount of data to get adequate results is very important. On a second note, such huge amount of data may lead to consumption of a huge amount of time and also require high-end hardware to process the data. So, for reducing the time consumption by the systems and reduction of load on every system, we go for a new architecture of systems of Hadoop Distributed File System based on the hierarchical approach of control.

In the year 2013, Azzedin[1] proposed an architecture which reduces the dependency of the size of metadata on the Namenode and also proposed a fault tolerant and highly scalable Hadoop Distributed File System. The metadata of the storage allocation and replication of the data is mainly stored in the RAM of the Namenode, so storing the metadata of a huge amount of data results in the increase of load on a single Namenode. Therefore, the architecture establishes itself properly on managing huge metadata by enrolling itself into a Chord protocol based architecture which directly connects to the HDFS to provide with a suitable solution to the scalability of the system. As the system uses a Chord Protocol, it impacts the system in the way of increasing complexity of Single HDFS Namenode Architecture.

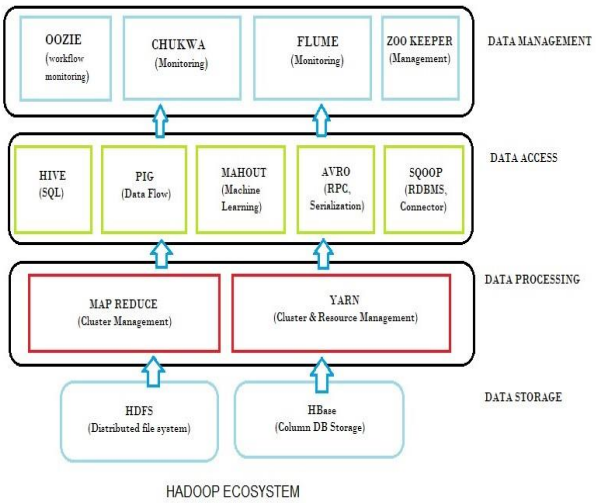

Fig.1. Hadoop Ecosystem

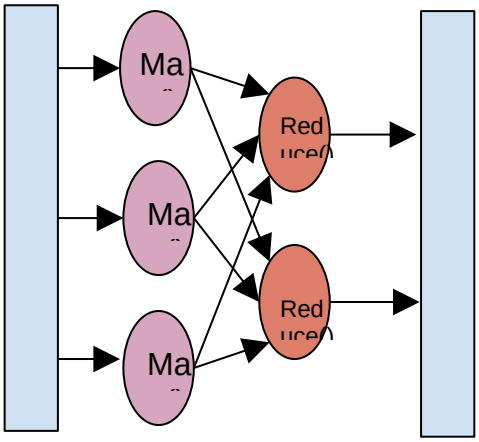

Fig.2. Map-Reduce process

Wang[2] in the year 2013, proposed an application framework of HDFS for processing of small and medium sized data. The Hadoop Architecture, which is one of the most sought after database framework for storing and analyzing Big Data has got some faults regarding

processing small and medium sized data sets. The relativity regarding the processing speed of the data is quite unbalanced in Hadoop as the processing speed of large and small data is more or less the same. So, in the proposed architecture, a caching mechanism is combined with the HDFS architecture to optimize the processing of small files. The increase in the size of the disk cache has significantly improved the processing speed of small data sets and led to the optimization of the system.

In 2014, KIM et al.[4] suggested a new HDFS Architecture consisting of several Namenodes to resolve the significant issues faced by an HDFS such as Single-Point- of-Failure (SPOF), Namespace limitation and Load Balancing Problem. Single-Point- of-Failure can be easily resolved by Quorum Journal Manager (QJM) which is available in Hadoop Versions of 2.1 and later. QJM manages and advocates Journal Nodes which carry the edit logs of the Namenode so that the Single-point- of-Failure can be removed in such a case. However, the namespace limitation and load balancing problem still pertains in the system. Therefore, to remove such dependencies, the paper proposes an architecture of establishing multiple Namenodes where one acts as the primary Namenode and others as the backup Namenodes.

Islam et al.[10] in 2015, proposed a Hybrid Design (Triple H) for ensuring effective data placement policies to speed up HDFS on HPC Clusters. The major idea behind the Hybrid design was the inclusion of High-Performance Hardware to the HDFS. The addition of such High- performance hardware has evaluated the system to be more optimized and more scalable. As the Hadoop Architecture maintains the complete metadata into the primary memory of the Namenode, so if the primary architecture of the Namenode is replaced with High-performance hardware then the I/O bottlenecks will also be minimized. This system will further result in an effective utilization of the Heterogeneous Storage resources that are available on HPC clusters.

In 2017, Stamatakis et al.[9] proposed a general-purpose architecture for High Availability Metadata Services. Metadata is one of the most important components of distributed file system. The metadata services are often considered as an independent component of the system, rather than a part of the data servers due to the simplicity of the design. Accessibility of metadata is looked forward as a key procedure to the interaction between the client and the data set that is stored in the data server because if a client is unable to access the metadata properly, then he might not even be able to access the datasets. Parallel Virtual File System (PVFS) and Hadoop Distributed File System (HDFS) have been offering highly available metadata services. However, both the distributed file system have specific disadvantages. Such as PVFS provides stateless replication of data blocks of the data on defined metadata servers over a shared network accessible storage due to which it suffers single point failures. The HDFS provides quorum based replication of data blocks which also puts a limit to the storage of metadata onto the main memory due to its checkpoint and roll forward solution. Thus the paper proposes a high availability solution to the metadata services by incorporating the file systems with a general purpose architecture of replicated metadata service (RMS). The RMS implements a type of service which replicates the metadata for guaranteed accessibility throughout the file system.

In the year 2013, Liu et al.[6] proposed a new architecture for maintaining metadata effectively and efficiently by incorporating the system with two symmetrical nodes and Quorum-based third party node. As the Namenode in Hadoop is susceptible to Single-point- Failure, the author has proposed two symmetrical Namenodes where one of the effective nodes is in an active state and the other one in the passive state. The Active node remains as the primary Namenode when it is performing a task but, if at any point in time the Active Node fails, then it is replaced by the passive node. Here the concurrency between the contents of the Namenode is maintained by the Quorum Journal Nodes(QJN). The QJM holds the complete edit logs of the Namenode, and the passive node is capable of reading all the edit logs and make changes to its respective namespace from the QJM. Therefore the act of QJM in this architecture proposes a simple concurrency control for maintaining the metadata across the Namenodes and also avoids the single-point-failure problem.

In the year 2017, Jena et al.[11] proposed some exceptional methods on the optimisation of the Big Data System by providing the special system of feature locality Sensitive Bloom Filter for the increase of the metric of performance. The Bloom Filter basically supports the quick selection of data in a networked system. The Bloom Filter keeps much hold in increasing the efficiency of such algorithms as it accounts for a much lower time complexity. The paper also explains the usage of Locality Sensitive Bloom Filter for increasing the efficiency of the data query optimization in Big Data and Statistical Analysis. The Paper has also made a comparative analysis of the Big Data Query Optimization Techniques in terms of Performance, Accuracy, Cost and Complexity. In this comparative analysis the Bloom Filter restricts itself in giving high performance and high accuracy by keeping the complexity and cost as low as possible. There has also been a discussion on the optimization of parameters in SVM on Big Data. The sequential optimization of the parameters in SVM is done by focussing on the conspiracy plots of the cross validation accuracy. But the major problem that was faced during the comparative analysis of the sequential optimization of the parameters of SVM was that it showed moderate level of performance and low accuracy by accounting moderate complexity and low cost.

Liu et al [6] in the year 2015, explained the method of Big Data Optimization in Smart Grid using the multi block ADMM [Algorithms based on alternating Direction Method of Multipliers]. The paper basically reviews the parallel and distributed optimization algorithms of Big Data optimization problem in smart grid communication networks. As we know that with modest size of data with the unprecedented number of algorithms on Data Science

may give an amplified form of accuracy with the massive amount of data set for resolving all many personalized problems of business such as Health Care Management, Intelligent Social Media Feed Analysis, Smart Grid Optimisation Techniques etc. But on the other hand the collection and processing of such sheer amount of data for the optimisation of the models is unpractical in every way. Also such huge data is full of noise, heterogeneous variety, outliers and may also be prone to the amicable amount of cyber attacks. Again, the Big Data Problems comes with some problems based on Time Complexity where a moderately accurate answer is accepted with less time than a Highly accurate answer with more time. As ADMM comes into the focus where it can break the huge amount of Data to smaller bits by addressing the convex problem in a much more discrete way. The established formulation of such large scale Big Data Problems may address the ADMM with much more sophisticated modification.

Jens et al.[32] discussed the technique of reducing the performance gap to well-tuned database systems. The similarities and differences between the techniques used in Hadoop with those used in parallel distributions were pointed out. In the modern world, dealing with terabytes and even petabytes of data is a reality. It is very essential to process these huge amounts of data sets in an even more efficient way. Thus, the use of the MapReduce processing framework has become quite popular due to the ease-of-use, scalability and failover properties. But these performance tweaks come with a price that the overall performance is nowhere near the performance of a fine-tuned parallel database. Therefore, numerous researches are being conducted to improve the performance of the MapReduce jobs in various aspects.

III. Proposed Architecture of High Availability Architecture

High Availability HDFS is an architecture which has conditioned itself in maintaining the three issues of HDFS such as Single-point-of-failure, metadata consistency across the nodes and balancing of the load on a particular Namenode when lot many Datanodes are interacting with the namespace. The High Availability HDFS defines a simple model of placing the Namenodes and Datanodes in such a way that the load on a single Namenode is always balanced by maintaining a threshold limit of data nodes and also the concurrency control between different Namenodes are established with the help of the Quorum Journal Nodes (QJN). The System follows a hierarchical structure for maintaining and processing a huge amount of data optimally and efficiently.

In Fig. 3, it is depicted that the Data Nodes are connected to the Namenode in a particular manner that all the Namenodes are having the same number of the Data Nodes.The numbers are decided by determining the load balancing capability of the Namenode. Also, the Namenode will maintain two limits on the number of Datanodes that is a threshold limit and a saturation limit. The Threshold limit k is the optimum number of Datanodes that can be connected to the Namenode for optimized processing and perfect load balancing, and the saturation limit is the maximum number of Datanodes that can be attached to the Namenode in any case. The reason for maintaining two limits on the connectivity of Data Nodes and Namenodes is that the threshold limit is the limit on which the Namenode can work optimally. However, at any point in time if a Datanode is switched on in the system, and all the Namenodes associated with the system have reached the threshold limit, then turning another Namenode for accompanying the newly arrived Datanode will be a waste of resource as it will be connected to only one Datanode. Therefore, we propose a saturation limit for the Namenodes to connect with the Data Nodes so that at any point in time if a Datanode arrives and the Namenodes achieve their Threshold then one of the Namenode will be able to connect with the newly arrived Datanode.

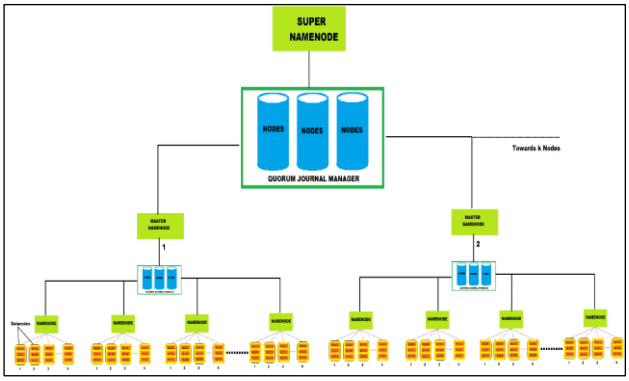

Fig.3. Double Layered Namenode Management

Also, we have included Quorum Journal Manager[QJM] to the architecture. The Quorum Journal Manager deploys Journal Nodes in the system that accesses the edit logs of the Namenodes and keeps all the Namenodes aware of all the read and writes that are occurring on the Data Nodes for processing of a particular log to maintain a check in the concurrency control between the Namenodes. As the architecture highly follows the model of hierarchy to establish a proper control over the data processing and establishing high scalability, so QRM is an essential part of the system.

Fig. 2 also depicts the presence of a Master Namenode. Moreover, the Master Namenode is connected to the QRM which is connected to the Namenodes. The Master Namenode also keeps a check on the QRM to maintain a proper concurrency control with the Namenodes. The main reason of this Master Namenode it to remove the Single point of Failure from the system design.

As at any point in time if any Namenode fails then it will be able to activate another Namenode which will further connect to the Datanodes of the failed Namenode and will also be able to access the edit logs from the Quorum Node Manager for further processing of data stored in Datanode.

IV. Methodology

The complete working of the architecture is such a way that all the Datanodes are connected to the Namenode, and the Namenode maintains proper threshold and saturation limits while connecting to the Datanodes. As all the Datanodes are connected to the Namenode so it may happen that a Datanode may get disconnected or may run into any breakdown, this breakdown will lead to a breakage in the connection and will also lead to a vacancy in the connection architecture of the Namenode. The vacancy may further be fulfilled by any of the Datanode that arrives first and requests for connection. Now, when at a certain point of time the disconnected Datanodes become active, then it will again request a connection to the system. Whenever a Datanode request for a connection, it first sends the ping requests to the Master Namenode. The Master Namenode then initiates a search between all the Namenodes tied under its QJM for an optimum solution. So now we come across two cases either it will find a vacancy or not. If a vacancy is found across any of the Namenodes, then it will first initiate a check in the QJM for any edit logs of the particular Datanode, and if an edit log is found, then it will fetch the edit log to its namespace for maintaining the proper metadata table of the allocation. However, if a vacancy is not found then the Datanode will be attached to the Namenode it was previously connected considering that it has not reached the saturation limit of attaching the Datanodes.

In Fig. 4, the Datanodes are represented by D with a number of their identification and the Namenodes are represented by N with also a number for their identification. In the first interpretation, we can observe that the Datanode D1 gets disconnected from the Namenode N1 and becomes non-active. Now as the N1 has one Datanode less than the threshold limit so that it will have a vacancy in its connection architecture. As D7 arrives for connection and puts a demand on N1 for connectivity, N1 first sends a request to its immediate QJM for any existence of D7's edit logs. If any edit logs of D7 is found then, N1 will fetch the edit log and will implement the same in its Namespace, and the connectivity between N1 and D7 will be established. However, if N1 does not find any existence of D7 in its immediate QJM, then it will further request the Master Namenode for any existence of edit logs in its QJM, this process will go until the last Namenode in the hierarchy is requested. If at the end no edit logs are found for a particular Datanode then it will be simply added to the Namenode.

In the second interpretation, it is observed that D1 again becomes active and demands connectivity to the Namenode in the hierarchy. However, now N1 will not be able to accommodate D1 as it has already reached the limit of the threshold. So now the request will be sent to the Namenode which is present in the immediate top of the hierarchy that is the Master Namenode. The Master Namenode will now initiate a request between all the Namenodes present in the same level or will have to request in the top of the hierarchy to find a suitable placement for the connectivity of the Datanode D1. So now the Namenode Nx is found after a search among all the Namenodes in the architecture. Also, Nx has one Datanode less than the threshold limit, so it will first search for an existence of edit logs of D1. As D1 was first connected to N1 so, Nx will be able to retrieve the complete edit log from the QJM of N1, and it will implement a connection between itself and D1.The Hierarchical Structure may turn out to be a complex one but helps in fetching the Metadata Table of a particular Namenode without affecting the system and by maintaining optimum accuracy. Therefore this system of fast retrieval will help Nx to completely retrieve the edit log of the Datanode D1 and maintain the concurrency of the Data without compromising the Data of the Datanode.

The interpretations of Fig. 4 are more or less same as that of Fig. 3 with only one difference with a special case of this architecture. The special case is that in the interpretation of Fig. 3 we could find a suitable Namenode for D1 as Nx which had one Datanode less in its connection architecture. However, in this interpretation, the Datanode D1 is connected to N1 when N1 has reached its threshold limit. The reason behind this stays in the fact that, when D1 demands connectivity to N1 then N1 which has reached the threshold by accepting D7 sends a request to the immediate Master Namenode for a search of a proper Namenode. However, in this scenario, all the Namenodes has reached their threshold for accepting any more Data Nodes so allocating a new Namenode for just one Datanode will be a waste of resource. Therefore the Namenode N1 goes for one more Datanode D1 and establishes a connection by fetching the proper edit logs of D1. So, in this case, the primary load balancing is shown where wastage of resource is avoided by not allocating a vacant Namenode to the system, in place we are allocating it to any present Namenode till its final Saturation limit is reached.

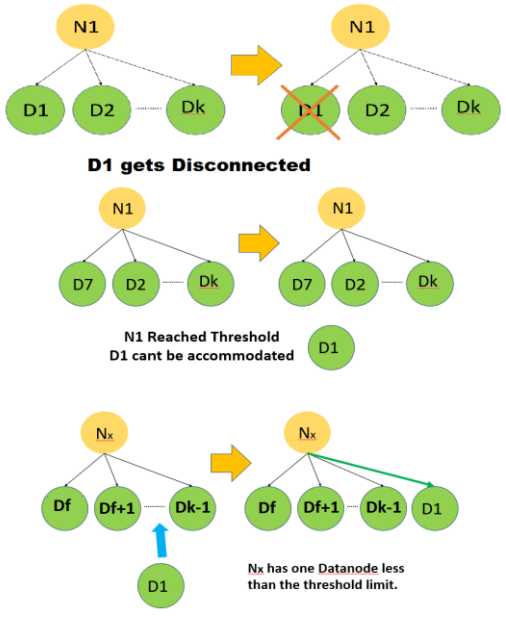

Fig.4. Interpretation of Connectivity Part 1

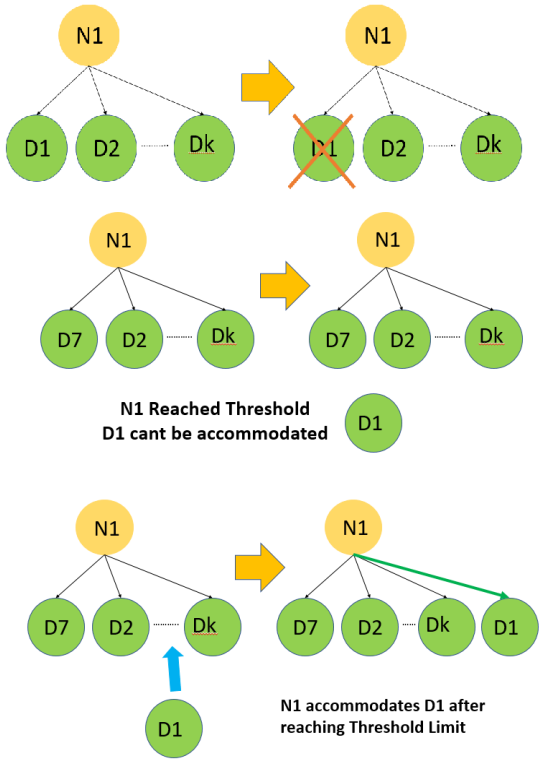

Fig.5. Interpretation of Connectivity Part 2

V. Conclusion

Hadoop Distributed File System has always been used in the field of Big Data Analytics to a huge extent. This system may be one of the most optimum ways of storing the data but have always left us in the position of resource wastage. The wastage of resources has been optimized in this proposal by setting up two limits of allocating Data Nodes, i.e., Threshold Limit and Saturation Limit and also by maintaining a Quorum Journal Node Manager for setting up proper concurrency between the Namenodes for maintaining the namespace. After maintaining the two limits in the Hadoop Architecture, we may achieve a Highly Available Hadoop Distributed File System for storing and processing data. As of now, no further practical implementation of the proposal has been performed. However, when performed practically some advantages and disadvantages of the Namenodes of the system may arise.

Acknowledgment

I express my profound gratitude to the Dean of School Of Computer Engineering, KIIT University Dr. Samaresh Mishra for allowing me to proceed with the report and also for giving me full freedom to access the lab facilities. My heartfelt thanks to Dr. Siddharth Swarup Rautaray & Dr. Manjusha Pandey for taking time and helping me through my work. They have been a constant source of encouragement without which the work might not have been completed on time. I am very grateful for their support. Their ideas and thoughts have been of great importance.

References

[1] Azzedin, Farag. "Towards a scalable HDFS architecture." Collaboration Technologies and Systems (CTS), 2013 International Conference on. IEEE, 2013.

[2] Wang, Xin, and Jianhua Su. "Research of distributed data store based on hdfs." Computational and Information Sciences (ICCIS), 2013 Fifth International Conference on. IEEE, 2013.

[3] Apache Software Foundation, “Quorum Journal Manager “, https://hadoop.apache.org/docs/r2.7.1/hadoop-project-dist/hadoop-hdfs/HDFSHighAvailabilityWithQJM.html

[4] Kim, Yonghwan, et al. "A Distributed NameNode Cluster for a Highly-Available Hadoop Distributed File System." Reliable Distributed Systems (SRDS), 2014 IEEE 33rd International Symposium on. IEEE, 2014.

[5] Apache Software Foundation, “Centralized Cache Management in HDFS,” http://hadoop.apache.org/docs/r2.3.0/hadoop-projectdist/ hadoop-hdfs/CentralizedCacheManagement.html.

[6] Tantisiriroj, Wittawat, et al. "On the duality of data-intensive file system design: reconciling HDFS and PVFS." Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis.ACM,2011.

[7] Zaharia, Matei, et al. "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing." Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association,2012.

[8] Demchenko, Yuri, Cees De Laat, and Peter Membrey. "Defining architecture components of the Big Data Ecosystem." Collaboration Technologies and Systems (CTS), 2014 International Conference on. IEEE, 2014.

[9] Stamatakis, Dimokritos, et al. "A General-Purpose Architecture for Replicated Metadata Services in Distributed File Systems." IEEE Transactions on Parallel and Distributed Systems (2017).

[10] Islam, Nusrat Sharmin, et al. "Triple-H: a hybrid approach to accelerate HDFS on HPC clusters with heterogeneous storage architecture." Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on. IEEE, 2015.

[11] Jena, Bibhudutta, et al. "A Survey Work on Optimization Techniques Utilizing Map Reduce Framework in Hadoop Cluster." International Journal of Intelligent Systems and Applications 9.4 (2017): 61.

[12] Stoica, Ion, et al. "Chord: a scalable peer-to-peer lookup protocol for internet applications." IEEE/ACM Transactions on Networking (TON) 11.1 (2003): 17-32.

[13] Karger, David, et al. "Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web." Proceedings of the twenty-ninth annual ACM symposium on Theory of computing. ACM, 1997.

[14] Ananthanarayanan, Ganesh, et al. "PACMan: Coordinated memory caching for parallel jobs." Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association, 2012.

[15] Zulkarnain, Novan, and Muhammad Anshari. "Big data: Concept, applications, & challenges." Information Management and Technology (ICIMTech), International Conference on. IEEE, 2016.

[16] Fetjah, Laila, et al. "Toward a Big Data Architecture for Security Events Analytic." Cyber Security and Cloud Computing (CSCloud), 2016 IEEE 3rd International Conference on. IEEE, 2016.
[17] Demchenko, Yuri, et al. "Addressing big data issues in scientific data infrastructure." Collaboration Technologies and Systems (CTS), 2013 International Conference on. IEEE, 2013.
[18] Ramaprasath, Abhinandan, Anand Srinivasan, and Chung-Horng Lung. "Performance optimization of big data in mobile networks", 2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE), 2015.
[19] Mayank Bhushan , Monica Singh , Sumit K Yadav ," Big Data query optimization by using Locality Sensitive Bloom Filter ",IJCT, 2015.
[20] E. Yildirim, J. Kim, and T. Kosar, "Optimizing the sample size for a cloud-hosted data scheduling service," in Proc. 2nd Int. Workshop Cloud Computing. Sci. Appl., 2012.
[21] Shvachko, Konstantin V. "HDFS Scalability: The limits to growth." ; login:: the magazine of USENIX & SAGE 35.2 (2010): 6-16 .
[22] The Apache Software Foundation, "The Apache Hadoop Project," http://hadoop.apache.org/.
[23] Fischer, Michael J., Nancy A. Lynch, and Michael S. Paterson. "Impossibility of distributed consensus with one faulty process." Journal of the ACM (JACM) 32.2 (1985): 374-382.
[24] Zaharia, Matei, et al. "Resilient distributed datasets: A fault- tolerant abstraction for in-memory cluster computing." Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association, 2012.
[25] Islam, Nusrat Sharmin, et al. "In-memory i/o and replication for hdfs with memcached: Early experiences." Big Data (Big Data), 2014 IEEE International Conference on. IEEE, 2014.
[26] Gray, Cary, and David Cheriton. Leases: An efficient fault-tolerant mechanism for distributed file cache consistency. Vol. 23. No. 5. ACM, 1989.
[27] D. Liben-Nowell, H. Balakrishnan, and D. R. Karger, ―Analysis of the evolution of peer-to-peer systems,‖ in Proc. 21st ACM Symp. Principles of Distributed Computing (PODC), Monterey, CA, July 2002, pp. 233 – 242.
[28] The Forrester Wave: Big Data Predictive Analytics Solutions, Q1 2013. Mike Gualtieri, January 13, 2013. [Online]. Available: http://www.forrester.com/pimages/rws/reprints/document/8 5601/oid/1-LTEQDI
[29] Morton, Guy M. A computer oriented geodetic data base and a new technique in file sequencing. New York: International Business Machines Company, 1966.
[30] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51.1 (2008): 107-113.
[31] Raji, R. Pillai. "MapReduce: Simplified Data Processing On Large Clusters." (2009).
[32] Jens Dittrich Jorge-Arnulfo Quian´eRuiz, "Efficient Big Data Processing in Hadoop MapReduce."

Authors' Profiles

Sabyasachi Chakraborty was born on the 9th of September 1996 in Shillong. He is continuing his studies in School of Computer Engineering as a B.Tech undergraduate at KIIT University. His research areas include Data Analytics and Big Data, Brain Computer Interface, Natural Language Processing and Machine Learning.

He is currently interning at HighRadius Technologies, Hyderabad, India as a Machine Learning Engineer. His paper on "A Proposal for Shelf Placement Optimization for Retail Industry using Big Data Analytics" was accepted at Data Science Congress 2017. He has also had his article on "Healthcare after the advent of Information Technology" published in CSI Communications.

Sabyasachi Chakraborty is the member of the IET (The Institution of Engineering and Technology).

Kashyap Barua was born on the 26th of November, 1994 in Assam. He is pursuing B.Tech degree in Computer Science & Engineering from KIIT University, Bhubaneswar, India. His field of research includes Big Data Analytics, Data Science and Machine Learning.

He has worked as a software developer intern at Zaloni, Guwahati. He is also the student coordinator of EMC^2 Academic Alliance at KIIT University. His paper on "A Proposal for Shelf Placement Optimization for Retail Industry using Big Data Analytics" was accepted at Data Science Congress 2017. He has also had his article on "Trends in Big Data" published in CSI Communications.

Kashyap Barua is the member of the IET (The Institution of Engineering and Technology).

Siddharth Swarup Rautaray, PhD (Computer Science), Member IEEE is Professor at the School of Computer Engineering, KIIT University, Bhubaneswar. He has more than a decade of teaching and research experience. Dr Rautaray has published numbers of Research Papers in peer-reviewed International Journals and conferences. His areas of interest are Image Processing, Data analytics, Human Computer Interaction.

Manjusha Pandey, PhD (Computer Science), Member IEEE is Professor at the School of Computer Engineering, KIIT University, Bhubaneswar. She has more than a decade of teaching and research experience. Dr Pandey has published numbers of Research Papers in peer-reviewed International Journals and conferences. Her areas of interest are WSN, Data analytics.

How to cite this paper: Sabyasachi Chakraborty, Kashyap Barua, Manjusha Pandey, Siddharth Rautaray," A Proposal for High Availability of HDFS Architecture based on Threshold Limit and Saturation Limit of the Namenode", International Journal of Information Engineering and Electronic Business(IJIEEB), Vol.9, No.6, pp. 27-34, 2017. DOI: 10.5815/ijieeb.2017.06.04

