INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 4.295
(Volume 4, Issue 2)
Available online at: www.ijariit.com

Adaptive replication in HDFS based on prediction techniques

Nikita Nabajee Sanap
nikita.sanap46@gmail.com
MET Bhujbal Knowledge City, Nashik, Maharashtra

Puja Bhanudas Sonar
misssonar27@gmail.com
MET Bhujbal Knowledge City, Nashik, Maharashtra

Pratiksha Sanjay Khode
pratikshakhode12@gmail.com
MET Bhujbal Knowledge City, Nashik, Maharashtra

Kamini Vijay Phad
kaminiphad75@gmail.com
MET Bhujbal Knowledge City, Nashik, Maharashtra

ABSTRACT

In a previous couple of decade, we've seen associate degree info explosion era and for that giant quantity of distributed knowledge being kept. The amount of applications supported Apache Hadoop is increasing, the explanation behind this is often lustiness and dynamic options of this technique. HDFS provides high availability and reliableness. Characteristics of parallel operations on the application layer, Access rate is completely different for every file in HDFS. HDFS uses cloud storage to implement the functionality. By considering the assorted drawbacks of replication system of HDFS, this paper implements an approach to flexibly replicate the data files. Predictive analysis is used for set the replication factor. The frequent access files are often replicated consistently with their access potential. Therefore, our approach concurrently improves the performance of HDFS and maintains high accessibility of data files.

Keywords: *Adaptive replication, Erasure coding, Proactive prediction, Hadoop.*

1. INTRODUCTION

The big data came with an approach in software and clarification development to abstract process and store required information. It deals with new challenges. In big data analytics, Apache Hadoop is most extolled parallel framework. Apache Hadoop is used to achieve high availability, as well as it is designed to find out and hold the breakdown [1]. Hence data consistency is also maintained. HDFS has been received to keep the high-throughput access and reliability for data-centric applications.

HDFS is one of the appropriate storage frameworks for distributed and parallel computing.
HDFS is originally operating by using an approach that systematically replicates 3 copies of every data file to improving the reliability. This approach is widely used to accomplish fault tolerance problems. 3 copies of every file are replicated to create additional consistency and availableness of data when failure is happened [3].

Apache Hadoop was made-up to get better computation power and performance in manipulation and data processing. Nowadays, Storing data on cloud storage is a standard option [4]. The cloud storage data is fetched using a good range of devices, like tablets, personal computers, and smart phones. Organizational data is stored in private cloud storage using the cloud technology and computing.

One amongst the popular strategies accustomed increase data reliability is allotment of data with inessential information on completely different storage devices. It provides advantages, when a storage component fails, information can still restore the failure element or re-build the original file. The reliability in distributed cloud storage is defined as, tolerance to storage node failure whereas availability means that promptly access to the original file and the storage efficiency is the amount of redundant information stored within the system.

In this cycle, huge volumes of information are created using several distributed system, such as social sites, search engines, data mining applications, etc.

In replication system, a complete block is replicated few times, for providing security if a replica of a block is gone or un-accessible and acquire fault tolerance feature. We have a tendency to implement a flexible replication management system as our project work to acquire high availability of the data in HDFS. Performance of the HDFS in enlarged with available data.

1.1 Existing system

The HDFS is a Distributed file system that overcomes the drawbacks of previous file handling systems. HDFS is the basis on two main elements i.e. Replication System and Map Reduce System [2]. Replication system is used to beat the issues associated with fault tolerance. It provides the reliability and high accessibility for computation by applying a static replication by default.

1.2 Problem withthe Existing system

In a distributed system, the numbers of a node are connected within the network. Every node has some hardware configuration. The configuration depends on the processor of the system, size of RAM, size of memory or storage capability. When a specific file gets access by users in a great deal, then the system gets overloaded. Because of the restricted configuration of nodes, they're powerless to handle a large amount of traffic [4]. As a result, the nodes get overloaded, so it may be going in waiting for state. Sometimes, the system gets crashed. The performance of file system is going to decrease and queries are raised on the availability of files. So, as per above discussion, system overloading is the disadvantage of the existing system.

1.3 Proposed System

In our project work, we implement the flexible replication system in HDFS to beat the drawbacks of the existing system. Here we tend to manage the replication system of the HDFS. We tend to try and modify its default strategy of replication.
Our framework is completely divided into three parts:-

1) Calculate access potential
2) Set replication factor
3) Update replication of every file

Above three sections are helped to implement adaptive Replication Framework.

1.4 Architecture

In the architecture diagram, HDFS is the Hadoop Distributed file system that store range of files within the format of the block. Those files are accessed using the HDFS logging system. The {data|the info|the information} files acts as a training data in the system. Size of those data may be in terms of a megabyte (MB), gigabyte (GB), terabyte (TB) or petabyte (PB). Training data is sent to monitoring system and user interface system. The monitoring system is employed to keep watch or take the feedback of system. The timer is employed for set counting for an explicit period in terms of a minute. There in fundamental quantity, access potential of every file is calculated. Once time is over, the calculated access potential is sent to the replication predictor. Replication predictor is used to set the replication factor for every file [5]. When calculations are done, the data is send to the knowledge base to update the metadata. On the opposite side, predicator sends updated data to replication management system. This system replicates the files on the nodes consistent with its new replication factor. The complete system is update with its new replication.

Algorithm

- Start server systems
- Upload file in HDFS and create an entry in the database
- Set countdown
- Calculate Access Potential
- Calculate replication factor
- Update database
- Set new replication for each file
- End

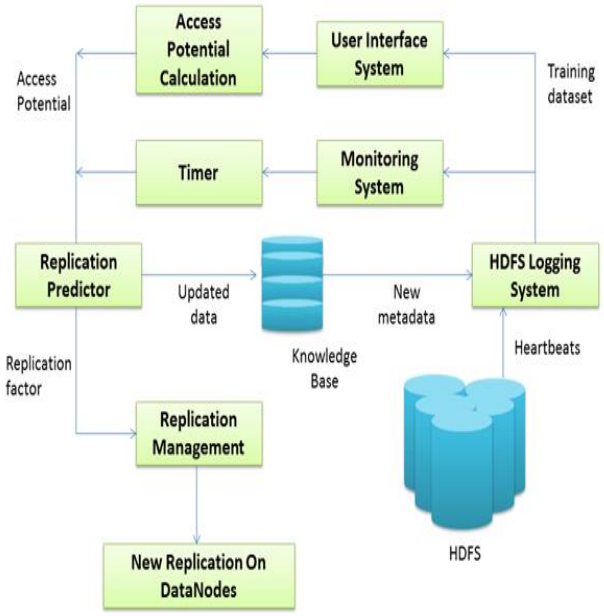

Fig. Architecture of System

1.5 Calculate Access Potential

Access potential is the term that used for calculating hit for the particular file. It provides the idea regarding the traffic of every file. Access potential is calculating the replication factor for every file. Access rate is depending on the number of hits in given unit of time. Data files are replicated using their own access potential. If we tend to think about that, a specific file gets total 72 hits in 4 minutes, the access potential for that file is 72 for four minutes of the time period. It means files get total 18 hits in a minute.

1.6 Replication Factor

The replication factor for every data files is calculated using the access potential of that file. For calculating replication factor, totally different parameters are used. Predication algorithm takes two parameters i.e. name of the file and its own access potential value. After deciding the replication factor, the metadata is updated and replication is completed in HDFS.

Each node has some restricted capability for handling the traffic for every file. The capability of the node relies on the RAM, processor and memory and cupboard space of that system.

Now, we have a tendency to take some parameters for calculating replication factor.

C = capability of the system to handle the traffic or hit for a specific file

T = Time in minutes

H = number of hits in time T

Now, first, we tend to calculate the number of hits per minute,

No_hits_per_min = (no. of hits in time T / time in minute)

= H / T

After calculating the number of hits per minute, the replication factor for files is finding out by referring the capability C of the system. Therefore,

Number of replications is,

No_replication = (No_hits_per_min / C)

According to a default policy of Hadoop, replication factor is 3. Thus we tend to add new replication factor in the existing and new replication factor is calculated.

New replication factor = 3+No_Of_replication

New replication factor denotes the final replication factor of that particular file. Using this methodology, the replication factor for all existing file is calculated and store it within the metadata.

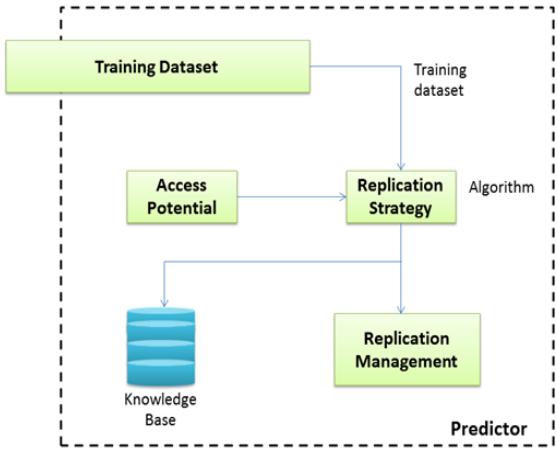

Fig. Predictor System

1.7 Replica Updation

Now, after calculation of replication factor of every file, the replication of existing file is updated and new replication factor is set. Every file of HDFS is replicated with its new replication factor. After successful replication, every file is stored on totally different nodes of racks by using locality policy.

2. NOTATIONS

The notations used in this paper are listed in Table 1.

Table -1: The Notations used in this Paper

Notations used in this paper	
Acronym	Description
C	Capability of System
T	Time
H	NO of Hits

3. PERFORMANCE ANALYSIS

Our proposed system solves the problem of replication management while uploading implementing a secure and efficient access mechanism across HDFS. For performance measure we compare the computational overhead that is incorporated in implementing secure replication and access control mechanism. Computational overhead is involved in process of replication which is measured in terms of time cost required to generate No of hits for data files accessed by N users.

The proposed system is implemented in PHP. The web services are hosted in Apache Tomcat 7. At the front end, php has used. Coding of application is done with JDK 1.7.Here MYSQL 5 database is used as back end for storing data and with INTEL 2.8 GHz i3 processor and 8 GB RAM.

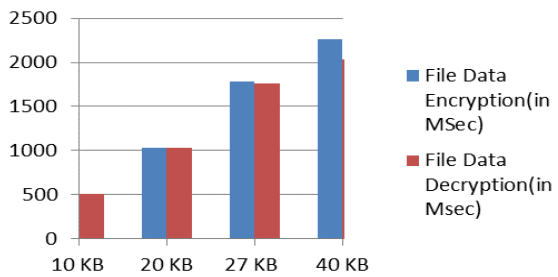

Fig -2: System performance (Performance analysis of CP_ABPRE scheme base on data size)

4. CONCLUSION

In order to improve the availability of HDFS by enhancing the data locality, our contributions focus on following points. First, we design the replication management system which is truly flexible to the characteristic of the data access pattern. The approach proactively performs the replication in a predictive manner. Second, we implement a complexity reduction method to solve the performance issue of the prediction technique. In fact, this complexity reduction method significantly accelerates the prediction process of the access potential estimation. Finally, we implement our method on a real cluster and verify the effectiveness of the proposed approach. With a rigorous analysis of the characteristics of the file operations in HDFS, our uniqueness is to create a flexible solution to advance the Hadoop system.

5. ACKNOWLEDGEMENT

Firstly we gladly thanks our project guide Prof. R. D. Deokar, for his valuable guidance for implementation of the proposed system. We will remain thankful for their excellent as well as polite guidance for the preparation of this report. Also, we would sincerely like to thank HOD of Our department Prof. N. R. Kale and other staff for their helpful coordination and support in project work.

6. REFERENCES

[1]What is apache Hadoop?"https://hadoop.apache.org/, accessed: 2015-08-13.
[2]M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, \Delay scheduling: a simple technique for achieving locality and fairness in
cluster scheduling," in Proceedings of the 5th European conference on Computer
systems. ACM, 2010, pp. 265278.
[3] K. S. Esmaili, L. Pamies-Juarez, and A. Datta, \The core storage primitive: Cross-object redundancy for efficient data repair access in erasure coded storage,"
arXiv preprint arXiv:1302.5192, 2013.
[4] G. Ananthanarayanan, S. Agarwal, S. Kandula, A. Greenberg, I. Stoica, D. Harlan, and E. Harris, \Scarlett: Coping with skewed content popularity in mapre-
duce clusters."in Proceedings of the Sixth Conference on Computer Systems, ser. EuroSys 11. New York, NY, USA: ACM, 2011, pp. 287300. [Online]. Available:
http://doi.acm.org/10.1145/1966445.1966472
[5] A. Papoulis, Signal analysis. McGraw-Hill, 1977, vol. 191.
[6] C. L. Abad, Y. Lu, and R. H. Campbell, \Dare: Adaptive data replication for efficient cluster scheduling."in CLUSTER. IEEE, 2011, pp. 159168.
[7] Z. Cheng, Z. Luan, Y. Meng, Y. Xu, D. Qian, A. Roy, N. Zhang, and G. Guan,\Erms: An elastic replication management system for hdfs."in Cluster Computing Workshops (CLUSTER WORKSHOPS), 2012 IEEE International Conference on, Sept 2012, pp. 3240.
[8] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S.Chen, and D. Borthakur, \Xoring elephants: Novel erasure codes for big data,"
in Proceedings of the VLDB Endowment, vol. 6, no. 5. VLDB Endowment, 2013,pp. 325336.
[9] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, \Dcell: a scalable and fault-tolerant network structure for data centers,"ACM SIGCOMM Computer Communication Review, vol. 38, no. 4, pp. 7586, 2008.
[10] A. Duminuco and E. Biersack, \Hierarchical codes: How to make erasure codes attractive for peer-to-peer storage systems,"in Peerto- Peer Computing, 2008.
P2P08. Eighth International Conference on. IEEE, 2008, pp. 8998.
[11] B. Calder, J. Wang, A. Ogus, N. Nilakantan, A. Skjolsvold, S. McKelvie, Y.
Xu, S. Srivastav, J. Wu, H. Simitci et al., \Windows azure storage: a highly available cloud storage service with strong consistency,"in Proceedings of theTwenty-Third ACM Symposium on Operating Systems Principles. ACM, 2011,pp. 143157.
[12] S. B. Wicker and V. K. Bhargava, Reed-Solomon codes and their applications. John Wiley Sons, 1999.
[13] C. Huang, M. Chen, and J. Li, \Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems,"ACM Transactions on
Storage (TOS), vol. 9, no. 1, p. 3, 2013.
and Grid Computing (ccgrid 2012). IEEE Computer Society, 2012, pp.419426.
[14]\What is ganglia?"http://ganglia.sourceforge.net/, accessed: 2015-08-13.
[15] X. Wu, Performance Evaluation, Prediction and Visualization of Parallel Systems, ser. The International Series on Asian Studies in Computer and Information Science. Springer US, 1999. [Online]. Available:
http://books.google.co.kr/books?id=IJZt5H6R8OIC
[16] R. Gallager, Stochastic Processes: Theory for Applications. Cambridge University Press, 2013. [Online].Available:http://books.google.co.kr/books?id=CGFbAgAAQBAJ
[17] K. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, \An introduction to kernel-based learning algorithms."Neural Networks, IEEE Transactions on, vol.12, no. 2, pp. 181201, Mar 2001.
[18] C. Rasmussen and C. Williams, Gaussian Processes for Machine Learning, ser.Adaptive Computation And Machine Learning. MIT Press, 2005. [Online].Available:http://www.gaussianprocess.org/gpml/chapters/
[19] K. Chalupka, C. K. I. Williams, and I. Murray, \A framework for evaluating approximation methods for gaussian process regression."CoRR, vol. abs/1205.6326,2012.

[20] E. G. Tsionas, \Maximum likelihood estimation of stochastic frontier models by the fourier transform."Journal of Econometrics, vol. 170, no. 1, pp. 234248,
2012.
[21] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kgl, \Algorithms for hyper parameter optimization."in NIPS, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett,F. C. N. Pereira, and K. Q. Weinberger, Eds., 2011, pp. 25462554.

