

Adaptive Replication In HDFS Based on Similarity Based Prediction Techniques

Yogesh S. Sapnar¹, Prof. M. B. Vaidya²

¹Dept of Computer Engineering

²Associate Professor, Dept of Computer Engineering

^{1,2} AVCOE, Maharashtra, India

Abstract- *In the previous couple of decade, we've seen associate degree info explosion era and for that giant quantity of distributed knowledge being keep. 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 implement an approach to flexibly replicate the data files. predictive analysis is used for set the replication factor. The frequent access files are often replicated consistent 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.

I. 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 approach that systematically replicates 3 copies of every data file for improve the reliability. This approach is widely used for accomplish fault tolerance problems. 3 copies of every file is 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 fetch using a good range of devices, like tablets, personal computers and smart phones. organizational data is stored on 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 an 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 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 with Existing system

In distributed system, the numbers of node are connected within the network. every node has some hardware configuration. The configuration depends on the processor of system, size of RAM, size of memory or storage capability. when specific file get access by users in great deal, then system get overloaded. because of restricted configuration of nodes, they're powerless to handle large amount of traffic[4]. As a result, the nodes get overloaded, so it may be goes in waiting state. Some time, 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 existing system.

1.3 Proposed System

In our project work, we implement the flexible replication system in HDFS to beat the drawbacks of 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 block. Those files are access using the HDFS logging system. the {data|the info|the information} files are act as a training data in system. Size of those data may be in terms of megabyte (MB), gigabyte (GB), terabyte (TB) or petabyte (PB). training data is send to monitoring system and user interface system. monitoring system is employed to keep watch or take the feedback of system. Timer is employed for set counting for explicit period in terms of minute. therein fundamental quantity, access potential of every file is calculated. once time is over, the calculated access potential is send 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 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

- 1) Start server systems
- 2) Upload file in HDFS and create entry in database
- 3) Set countdown
- 4) Calculate Access Potential
- 5) Calculate replication factor
- 6) Update database
- 7) Set new replication for each file
- 8) 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 get total 72 hits in 4 minute, the access potential for that file is 72 for four minute of 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 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 node is rely 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 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,

$$\text{No_hits_per_min} = (\text{no. of hits in time T} / \text{time in minute}) = H / T$$

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

Number of replications is,

$$\text{No_replication} = (\text{No_hits_per_min} / C)$$

According to 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.

$$\text{New replication factor} = 3 + \text{No_replication}$$

New replication factor denote 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.

II. NOTATIONS

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

Table -1: The Notations used in this Paper

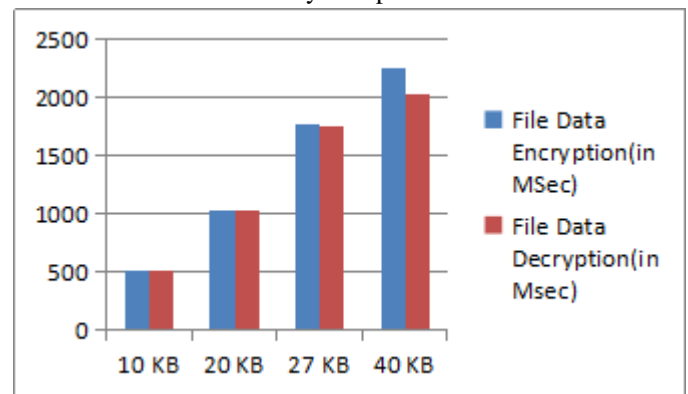
| Notations used in this paper | |
|------------------------------|----------------------|
| Acronym | Description |
| C | Capability of System |
| T | Time |
| H | No of Hits |

III. 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 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.

Table -2: System performance



IV. CONCLUSIONS

In order to improve the availability of HDFS by enhancing the data locality, our contribution 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 the 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 on the characteristics of the file operations in HDFS, our uniqueness is to create an flexible solution to advance the Hadoop system.

V. ACKNOWLEDGEMENT

A very firstly I gladly thanks to my project guide Dr. M. B. Vaidya, for her valuable guidance for implementation of proposed system. I will forever remain a thankful for their excellent as well as polite guidance for preparation of this report. Also I would sincerely like to thank to HOD of computer department Mr. R. L. Paikrao and other staff for their helpful coordination and support in project work

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 mapreduce 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] G. Kousiouris, G. Vaadis, and T. Varvarigou, "Enabling proactive data management in virtualized hadoop clusters based on predicted data activity patterns." in P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference on, Oct 2013, pp. 18.
- [6] A. Papoulis, Signal analysis. McGraw-Hill, 1977, vol. 191.
- [7] Q. Wei, B. Veeravalli, B. Gong, L. Zeng, and D. Feng, "Cdrm: A cost-effective dynamic replication management scheme for cloud storage cluster" in Cluster Computing (CLUSTER), 2010 IEEE International Conference on, Sept 2010, pp. 188196.
- [8] C. L. Abad, Y. Lu, and R. H. Campbell, "Dare: Adaptive data replication for efficient cluster scheduling." in CLUSTER. IEEE, 2011, pp. 159168.
- [9] 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.
- [10] 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.
- [11] 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.
- [12] A. Duminuco and E. Biersack, "Hierarchical codes: How to make erasure codes attractive for peer-to-peer storage systems," in Peer-to-Peer Computing, 2008.P2P08. Eighth International Conference on. IEEE, 2008, pp. 8998.
- [13] 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 the Twenty-Third ACM Symposium on Operating Systems Principles. ACM, 2011, pp. 143157.
- [14] B. Fan, W. Tantisiriroj, L. Xiao, and G. Gibson, "Diskreduce: Replication as a prelude to erasure coding in data-intensive scalable computing," Parallel Data Laboratory, Carnegie Mellon University, Pittsburgh, 2011.
- [15] S. B. Wicker and V. K. Bhargava, Reed-Solomon codes and their applications. John Wiley Sons, 1999.
- [16] 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.
- [17] A. Datta and F. Oggier, "Redundantly grouped cross-object coding for repairable storage," in Proceedings of the Asia-Pacific Workshop on Systems. ACM, 2012, p. 2.

- [18] Y. Chen, S. Alspaugh, and R. Katz, \Interactive analytical processing in big data systems: A cross-industry study of map reduce workloads," Proceedings of the VLDB Endowment, vol. 5, no. 12, pp. 18021813, 2012.
- [19] Z. Guo, G. Fox, and M. Zhou, \Investigation of data locality in map reduce," in Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012). IEEE Computer Society, 2012, pp.419426.
- [20] What is ganglia?" <http://ganglia.sourceforge.net/>, accessed: 2015-08-13.
- [21] 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>
- [22] R. Gallager, Stochastic Processes: Theory for Applications. Cambridge University Press, 2013. [Online]. Available:<http://books.google.co.kr/books?id=CGFbAgAAQBAJ>
- [23] 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.
- [24] 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/>
- [25] 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.
- [26] E. G. Tsionas, \Maximum likelihood estimation of stochastic frontier models by the fourier transform." Journal of Econometrics, vol. 170, no. 1, pp. 234248,2012.
- [27] 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.
- [28] T. T. Cai, T. Liang, and H. H. Zhou, \Law of log determinant of sample covariance matrix and optimal estimation of differential entropy for high-dimensional gaussian distributions." CoRR, vol. abs/1309.0482, 2013.
- [29] P. Sollich and C. K. I. Williams, \Understanding gaussian process regression using the equivalent kernel." in Deterministic and Statistical Methods in Machine Learning, ser. Lecture Notes in Computer Science, J. Winkler, M. Niranjana, and N. D. Lawrence, Eds., vol. 3635. Springer, 2004, pp. 211228. [Online]. Available:<http://dblp.unitrier.de/db/conf/dsmml/dsmml12004.html>.