Imperial Journal of Interdisciplinary Research (IJIR)
Vol-3, Issue-7, 2017
ISSN: 2454-1362, http://www.onlinejournal.in

In-Line Deduplication Technique using Multilevel Hash Indexing

Manali J Mahajan[1] & Chetan Shetty[2]
[1]Student, M.Tech Computer Science and Engineering, Ramaiah Institute of Technology, Bengaluru, India
[2]Assistant Professor, Science and Engineering, Ramaiah Institute of Technology, Bengaluru, India

Abstract: *The Cloud storage increases rapidly now-a-days, to improve the performance many techniques are upcoming, among that MapReduce become popular and efficient. Proposed system introduce Multilevel Hash Indexing technique to improve the MapReduce technique. The network traffic is a big issue in this digital communication world. In this system we proposed intermediate server to minimize the network traffic by means of buffering concepts. To make the system more efficient MapReduce technique is used. The intermediate server also called as community server is placed for each location, in general people on one location always access similar files, this is the key which reduce the network traffic, Since the blocks are store in intermediate server, it improves the performance of the system by reducing the time to access the file.*

Keywords: MapReduce, Community Server

1. Introduction

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into chunks.
While uploading the file from the client system to the cloud server, if simultaneously multiple clients uploads the file to the cloud, network traffic will be created if the single server has to do the map reducing as well as the cloud uploading process.
To reduce the network traffic as well as to do the map reducing with respect to the region of the client, by implementing online algorithm, we are redirecting of the client request to the respective region server.
In this paper, we solved the issue of data partition and aggregation for a MapReduce technique with an objective that is to minimize the network traffic considerably. In specific, we propose a distributed data storage algorithm for big data applications by decomposing the original large-files into several blocks that can be stored in community server to reduce the network traffic in parallel. Moreover, an online technique is designed to deal with the data partition and aggregation in a dynamic manner. Finally, We tested this with sample files and our experimental results demonstrate that this proposals can significantly reduce network traffic cost and performance.

1.1 System Model

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into chunks.
While uploading the file from the client system to the cloud server, if simultaneously multiple clients uploads the file to the cloud, network traffic will be created if the single server has to do the map reducing as well as the cloud uploading process.
To reduce the network traffic as well as to do the map reducing with respect to the region of the client, by implementing online algorithm, we are redirecting of the client request to the respective region server.

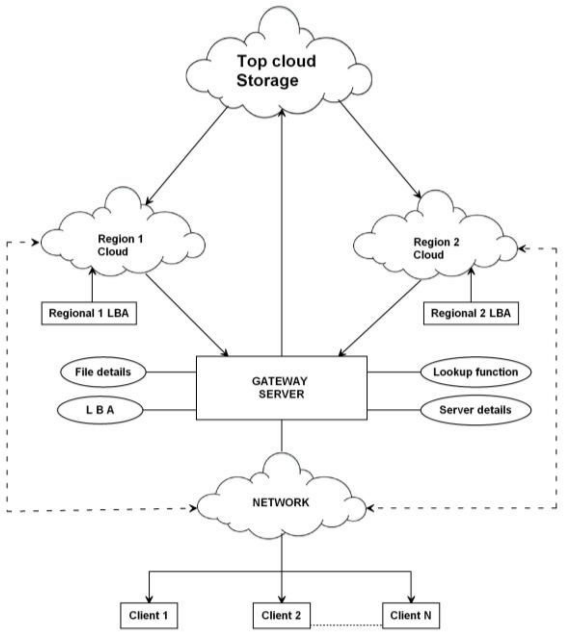


Figure. 1.1.1 Overview of the system

N number of clients may access the system to download their desire file from the cloud. All the request will go to Gateway system which uses the LoopkUp function and based on the client IP Address it will direct the request to corresponding Community Server (Regional Server). Community Server looks for the blocks in requested file using LBA (Logical Block Addressing) technique. Once the blocks are identified, it will check the blocks which are available in that community storage and which blocks are not available. It sends the request for the non available blocks to Top Cloud Storage and gets it. Once unavailable e blocks it receive, it fetch all the blocks for the request file, merge it and give to the client. The newly received blocks are store in community storage for further use. This process is shown in Fig. 1.1.1.

2 Previous Work

To solve the Big Data problem MapReduce technique is introduced, which is successful in reducing the storage space but it is not deal with the Network Traffic problem.

MapReduce

MapReduce is a core component of the Apache Hadoop software framework.

Hadoop enables resilient, distributed processing of massive unstructured data sets across commodity computer clusters, in which each node of the cluster includes its own storage. MapReduce serves two essential functions: It parcels out work to various nodes within the cluster or map, and it organizes and reduces the results from each node into a cohesive answer to a query.

- MapReduce is composed of several components, including:
- JobTracker -- the master node that manages all jobs and resources in a cluster
- TaskTrackers -- agents deployed to each machine in the cluster to run the map and reduce tasks
- JobHistoryServer -- a component that tracks completed jobs, and is typically deployed as a separate function or with JobTracker

To distribute input data and collate results, MapReduce operates in parallel across massive cluster sizes. Because cluster size doesn't affect a processing job's final results, jobs can be split across almost any number of servers. Therefore, MapReduce and the overall Hadoop framework simplify software development. MapReduce is available in several languages, including C, C++, Java, Ruby, Perl and Python. Programmers can use MapReduce libraries to create tasks without dealing with communication or coordination between nodes.

MapReduce is also fault-tolerant, with each node periodically reporting its status to a master node. If a node doesn't respond as expected, the master node re-assigns that piece of the job to other available nodes in the cluster. This creates resiliency and makes it practical for MapReduce to run on inexpensive commodity servers.

MapReduce in action

For example, users can list and count the number of times every word appears in a novel as a single server application, but that is time consuming.

3 Proposed Methodology

This system has three servers, Gateway Server & 2 Community Server. There are three cloud storage areas. Each community server has its own cloud storage space and one main cloud storage. Since this system has two community servers it has two cloud storage space.

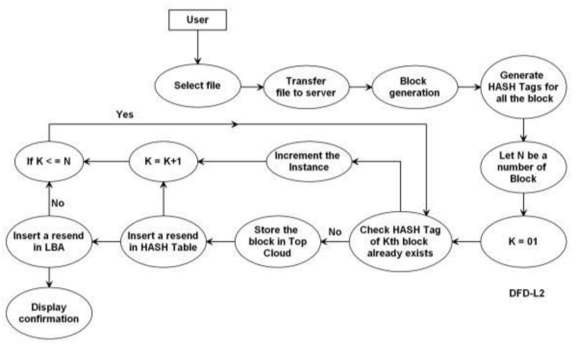

User
Select file
Transfer file to server
Block generation
Generate HASH Tags for all the block
Let N be a number of Block
K = 01
Check HASH Tag of Kth block already exists
No
Store the block in Top Cloud
Insert a resend in HASH Table
Insert a resend in LBA
Display confirmation
No
If K < = N
Yes
K = K+1
Increment the Instance
DFD-L2

Figure. 3.1 File Upload Process

When the user uploads a file it has to go to gateway server. Where the file is divided into blocks then for each block hash code is generated using MD5 algorithm. Using the generated hash code gateway server as to check whether the block is already present or not in block management system. If the block is present then the instance of the corresponding block will incremented and that block is not stored. If the block is not present then a new record is inserted with instance value 1. For all blocks in the file this process will be done. After this logical block address of the file will be determined and store in a file record. This process in shown in Fig. 3.1.

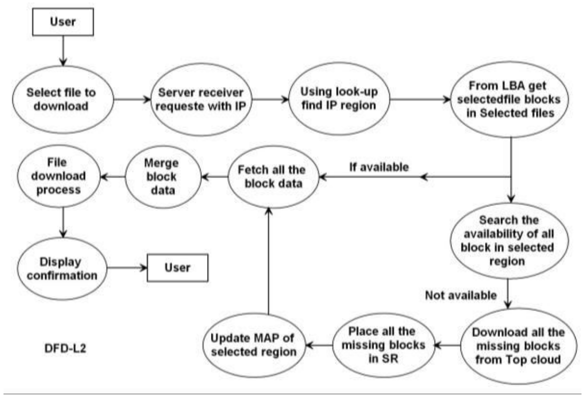

User
Select file to download
Server receiver requeste with IP
Using look-up find IP region
From LBA get selectedfile blocks in Selected files
If available
Fetch all the block data
Merge block data
File download process
Search the availability of all block in selected region
Display confirmation
User
Not available
DFD-L2
Update MAP of selected region
Place all the missing blocks in SR
Download all the missing blocks from Top cloud

Figure. 3.2 File Download Process

While user want to download a file he as to sent the request to the gateway server. Based on the IP address of the client node corresponding community server will be identified using IP Lookup function. Based on the file requested, LBA of the file is retrieved and sent to corresponding community server. The community server checks what the blocks available in its community storage and which are the blocks not available. The non available blocks are retrieved from top cloud storage and merge it and give to the client system. This process in shown in Fig. 3.2.

Multilevel Hash Indexing

While file is uploading it is divided into multiple blocks and for each block hashcode is generated. With the help of hashcode this system will check for the presence of the block in the cloud.

Consider there are huge numbers of blocks even though hashcode is 16 byte it took long time to compare and take decision whether block is presence or not.

To overcome this problem this system is introduced new technique multilevel hash indexing. In this technique the hashcode is divided into three blocks and each block is tested in corresponding hash level index which is showed in fig 3.3.

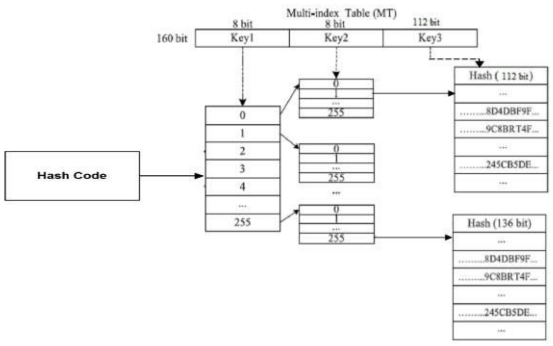
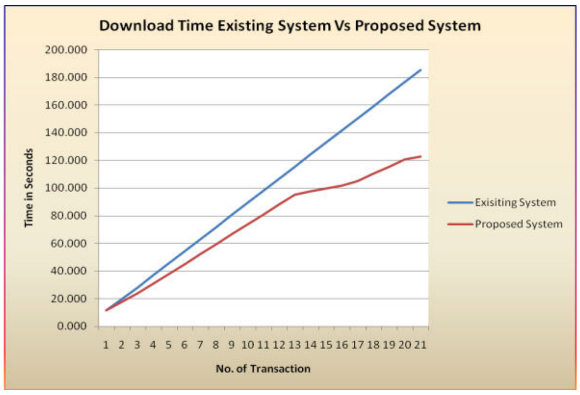


Figure. 3.3 Multilevel hash indexing

4 Experimental Results

System is tested with different number of users with more files uploaded and downloaded.

Graph 1 show the time comparison of the file download time for existing and proposed system. In the beginning the time difference is not much when the transactions are increased, the data in intermediate server getting filled so that the download time will be reduced considerably which in turn reduce the net work traffic.



Graph 1. File Download Time Comparision. Existing Vs Proposed.

5 Conclusion

In this paper we solved big issues in Bigdata storage system. Using Multilevel Hash Indexing technique over Mapreduce. To minimize the storage space, to minimize the network traffic this system has community servers, This system is developed in web based technology and tested with three servers. Where one is the gateway server and other two are community servers. This system tested with large number of files and the results shows there is a huge reduction in storage and there is a considerable time reduction in file transfer. This system can be improved for its security features as a feature work.

6. References

[1] J. Dean and S. Ghemawat, “Mapreduce: simplified data processing on large clusters,” Communications of the ACM, vol. 51, no. 1,pp. 107–113, 2008.
[2] W. Wang, K. Zhu, L. Ying, J. Tan, and L. Zhang, “Map task scheduling in mapreduce with data locality: Throughput and heavy-traffic optimality,” in INFOCOM, 2013 Proceedings IEEE.IEEE, 2013, pp. 1609–1617.
[3] F. Chen,M. Kodialam, and T. Lakshman, “Joint scheduling of processing and shuffle phases in

mapreduce systems," in INFOCOM, 2012 Proceedings IEEE. IEEE, 2012, pp. 1143–1151.

[4] Y. Wang, W. Wang, C. Ma, and D. Meng, "Zput: A speedy data uploading approach for the hadoop distributed file system," in Cluster Computing (CLUSTER), 2013 IEEE International Conference on. IEEE, 2013, pp. 1–5.

[5] T. White, Hadoop: the definitive guide: the definitive guide. " O'Reilly Media, Inc.", 2009.

[6] S. Chen and S. W. Schlosser, "Map-reduce meets wider varieties of applications," Intel Research Pittsburgh, Tech. Rep. IRP-TR-08-05, 2008.

[7] J. Rosen, N. Polyzotis, V. Borkar, Y. Bu, M. J. Carey, M. Weimer, T. Condie, and R. Ramakrishnan, "Iterative mapreduce for large scale machine learning," arXiv preprint arXiv:1303.3517, 2013.

[8] S. Venkataraman, E. Bodzsar, I. Roy, A. AuYoung, and R. S. Schreiber, "Presto: distributed machine learning and graph processing with sparse matrices," in Proceedings of the 8th ACM European Conference on Computer Systems. ACM, 2013, pp. 197– 210.

[9] A. Matsunaga, M. Tsugawa, and J. Fortes, "Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications," in eScience, 2008. eScience'08. IEEE Fourth International Conference on. IEEE, 2008, pp. 222–229.

[10] J. Wang, D. Crawl, I. Altintas, K. Tzoumas, and V. Markl, "Comparison of distributed data-parallelization patterns for big data analysis: A bioinformatics case study," in Proceedings of the Fourth International Workshop on Data Intensive Computing in the Clouds (DataCloud), 2013.

[11] R. Liao, Y. Zhang, J. Guan, and S. Zhou, "Cloudnmf: A mapreduce implementation of nonnegative matrix factorization for large scale biological datasets," Genomics, proteomics & bioinformatics, vol. 12, no. 1, pp. 48–51, 2014.

[12] G. Mackey, S. Sehrish, J. Bent, J. Lopez, S. Habib, and J. Wang, "Introducing map-reduce to high end computing," in Petascale Data Storage Workshop, 2008. PDSW'08. 3rd. IEEE, 2008, pp. 1–6.

[13] W. Yu, G. Xu, Z. Chen, and P. Moulema, "A cloud computing based architecture for cyber security situation awareness," in Communications and Network Security (CNS), 2013 IEEE Conference on. IEEE, 2013, pp. 488–492.

[14] J. Zhang, H. Zhou, R. Chen, X. Fan, Z. Guo, H. Lin, J. Y. Li, W. Lin, J. Zhou, and L. Zhou, "Optimizing data shuffling in dataparallel computation by understanding user-defined functions," in Proceedings of the 7th Symposium on Networked Systems Design and Implementation (NSDI), San Jose, CA, USA, 2012.

[15] F. Ahmad, S. Lee,M. Thottethodi, and T. Vijaykumar, "Mapreduce with communication overlap," pp. 608–620, 2013.

[16] H.-c. Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker, "Mapreduce- merge: simplified relational data processing on large clusters," in Proceedings of the 2007 ACM SIGMOD international conference on Management of data. ACM, 2007, pp. 1029–1040.

[17] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, J. Gerth, J. Talbot, K. Elmeleegy, and R. Sears, "Online aggregation and continuous query support in mapreduce," in Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. ACM, 2010, pp. 1115–1118.

[18] A. Blanca and S. W. Shin, "Optimizing network usage in mapreduce scheduling."

[19] B. Palanisamy, A. Singh, L. Liu, and B. Jain, "Purlieus: localityaware resource allocation for mapreduce in a cloud," in Proceedings of 2011 International Conference for High Performance Computing,
Networking, Storage and Analysis. ACM, 2011, p. 58.

[20] S. Ibrahim, H. Jin, L. Lu, S. Wu, B. He, and L. Qi, "Leen: Locality/fairness-aware key partitioning for mapreduce in the cloud," in Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on. IEEE, 2010, pp. 17– 24.

[21] L. Fan, B. Gao, X. Sun, F. Zhang, and Z. Liu, "Improving the load balance of mapreduce operations based on the key distribution of pairs," arXiv preprint arXiv:1401.0355, 2014.

[22] S.-C. Hsueh, M.-Y. Lin, and Y.-C. Chiu, "A load-balanced mapreduce algorithm for blocking-based entity-resolution with multiple keys," Parallel and Distributed Computing 2014, p. 3, 2014.

[23] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears, "Mapreduce online." in NSDI, vol. 10, no. 4, 2010, p. 20.

[24] J. Lin and C. Dyer, "Data-intensive text processing with mapreduce," Synthesis Lectures on Human Language Technologies, vol. 3, no. 1, pp. 1–177, 2010.

[25] P. Costa, A. Donnelly, A. I. Rowstron, and G. O'Shea, "Camdoop: Exploiting in-network aggregation for big data applications." In NSDI, vol. 12, 2012, pp. 3–3.