Secure Data Deduplication in Hadoop Distributed File Storage System

Naresh Kumar
Assistant prof. Department of Computer Science and Engineering, University Institute of Engineering and Technology (UIET), Kurukshetra University, Kurukshetra (KUK), 136119, India

Shalini
Department of Computer Science and Engineering, University Institute of Engineering and Technology (UIET), Kurukshetra University, Kurukshetra (KUK), 136119, India

Abstract – Big data is a collection of large amount of data that contains duplicated items collected from various sources. To detect duplicated data from big data is a serious issue. In this paper secure deduplication mechanism has been proposed for big data storage. In proposed work, fixed size chunking mechanism is used to create chunks of a large file and SHA-2 algorithm is used to generate secure hashes of chunks and on the basis of these hashes deduplication has been done. The proposed mechanism has been implemented using Hadoop and analyzed using chunk time and hash time.

Index Terms – Big Data, Deduplication, Fixed Size Chunking (FSC), Secure Hash Algorithm (SHA) and Message Digest (MD) and Hash function.

1. INTRODUCTION

In big data storage, large amount of data is stored in the form of Giga Bytes (GB) and Tera Bytes (TB). The data stored in big data is in unstructured form means data without any format. This large amount of data is collected from different sources that were increases duplicity in data [1]. To find duplicate data and remove it from big data storage is a difficult problem also managing unstructured data or converting it into structured form is difficult problem to solve. To solve above mentioned problems, lots of techniques exists named File-level chunking, Block-Level Chunking. Block-Level Chunking further divided into Fixed-Size Chunking and Variable Size Chunking to find duplicate data [2].

1.1 File-Level Chunking

File-level chunking was also known as Whole File Chunking, It considered all entire file as a single file and create chunk of entire file rather than break the file into multiple chunks, here single index was created for all the file and this process was helpful to save the storage space. The drawback was this technique was that if small changes are required in existing file then no provision for that user need to change complete file [3].

1.2 Fixed Size Chunking

In Fixed -sized chunking, the file was divide into fixed size block, set their boundary values on the basis of offsets. This method overcomes the drawback of File Level chunking technique. Suppose user wants to alter a few bytes then only altered chunks need to re-index and move to the backup location. The drawback of this method is byte shifting problem mans which bytes will be shifted upward or downward while data will be altered [4].

1.3 Variable Sized Chunking

Variable-Sized chunking method splits the file into multiple chunks according to the size or content of file. The drawback of this technique was that when chunks are created dynamically then their indexes are also generated dynamically and if alteration is required in chunks then same alteration is also needed in indexes that were increase extra overhead and also increases total processing time [5].

On the basics of chunking techniques only data is divided into different small chunks and theses chunks only helps to find duplicate contents but there is no security mechanism, for this some security mechanism such as MD-5, SHA-1, SHA-2 were needed to get some secure hash value of chunk but in SHA-1 technique security level is week because it takes only 160bits to generate hash values also it takes more rounds to generate secure hash value.MD-5 takes more time to generate secure hash value and it takes 512 bits for generation of secure hash value this was increases overall process creation time. So to overcome drawback of both techniques in this paper SHA-2 is used to generate secure hash value [6] [7].

In this paper secure data deduplication mechanism for finding duplicate chunks is to be proposed in which on the basis of secure hash value duplicate chunks has been detected and only unique chunks will be stored in HDFS. The proposed mechanism will generate chunks and hash in less time in perspective of existing technique [8] [9].

2. RELATED WORK

Sun *et al*. [10] proposed an information chunk system for No Sql data bases in view of the property structure tree.

Xia *et al*. 2014 [11] displayed DARE, a deduplication-mindful, low-overhead similarity discovery and disposal conspire for delta compression on the highest point of deduplication on reinforcement datasets.

Wu *et al*. [12] proposed the INS to process not just file compression, chunk coordinating, information de-duplication, ongoing input control, IP data, and occupied level record checking, yet additionally file storage, upgraded hub determination, and server stack adjusting. By compacting and apportioning the files as indicated by the chunk size of the cloud file framework, they diminish the information duplication rate. The handled files were encoded into the mark by MD5 fingerprint for the INSs to coordinate, file, assign to the storage servers, and give important transferring data to the customers.

Kumar and Shalini [13] provides evaluation of different chunking and deduplication techniques with different parameters and they also provides comparative analysis between different hashing techniques such as SHA and MD5.

Lu *et al.*[14] proposed efficient Key Value store on flash with a Bloom Filter based index structure called Bloom Store that not only assures an extremely low amortized RAM overhead per KV pair by keeping a flash-page sized data buffer and a very small sized BF buffer per Bloom Store instance in RAM, but also achieves a high lookup/insertion throughput by reducing the maximum number of flash page reads with key-range partitioning; by buffering multiple BFs per Bloom Store instance in RAM to reduce the BF-containing flash page reads and writes.

Yan *et al.* [15] proposed a practical scheme to manage the encrypted big data in cloud with deduplication based on ownership challenge and PRE. Proposed scheme could flexibly support data update and sharing with deduplication even when the data holders were offline.

Fu *et al.* [16] proposed, an application-aware scalable inline distributed deduplication framework in cloud environment, to meet this challenge by exploiting application awareness, data similarity and locality to optimize distributed deduplication with inter-node two-tiered data routing and intra-node application-aware deduplication.

3. PROPOSED WORK

In Big data, there are lots of duplicate data and to remove this duplicate data in secure manner various techniques are used like SHA-1, SHA-2, MD-5, etc. In proposed mechanism SHA-2 algorithm is used to find duplicate data from given real data set in secure manner.

3.1 Proposed Framework

In proposed mechanism, real dataset is downloaded from the "data.World" website. This dataset is loaded into memory for further processing of data. This dataset is divided into small blocks called chunks by using fixed chunking algorithm. In Fixed size chunking algorithm, all the files are divided into fixed size chunks which have fixed boundaries of 1MegaByte. After that some unique value of chunks are created through SHA-2 algorithm. These unique values are called hash values. Before storing these hash values in HDFS, we compare the hash value of every chunks separately if hash value of two chunks are same then it means there is a duplicate data and if it is not same then it means every chunk has different value and these chunks are stored in HDFS.

3.2 Proposed Algorithm

Algorithm. Every hash value of each chunk is compared with hash value of another chunk and if hash values of both chunks are same then there is duplicate data presented in real dataset then discard this duplicate data and that data is not stored in HDFS, otherwise chunk with different hash value are stored in HDFS.

Description of Proposed Algorithm

In the proposed algorithm, real data set is taken from the 'data. World' website and then this data is divided into different fixed sized chunks by applying fixed size chunking algorithm and then unique hash value of these chunks are generated through SHA-2

3.3 Fixed Size Chunking Algorithm

In Fixed size chunking algorithm, file is divided into fixed size chunks which have fixed boundaries of 1 Megabytes. If there will be few changes in file then only that chunks are to be reindexed instead of whole file and only that chunks are moved to the backup location. The fixed size chunking algorithm, first provide input of data set and then set the size of chunks. Load the file into memory which generates the chunks in bytes form and passed these chunks to next algorithm for further processing.

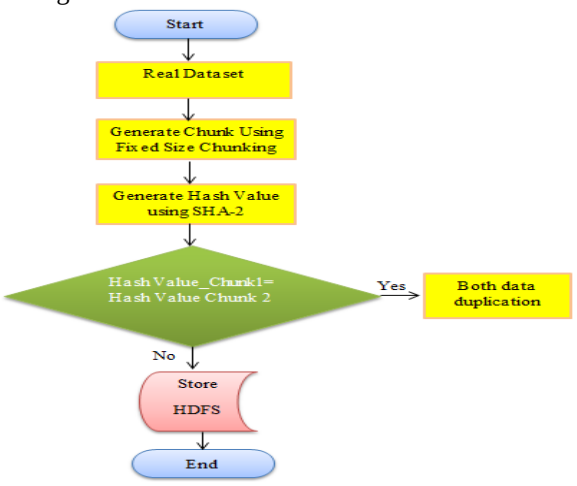

Figure 3.2: Flow Chart of Propose Algorithm

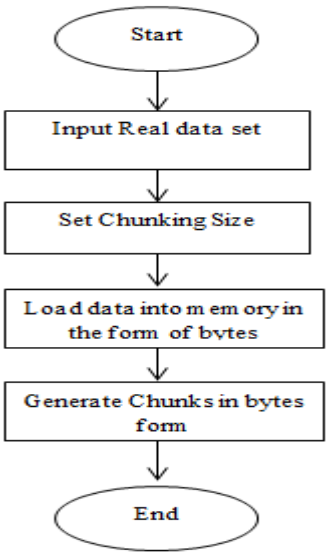

Figure.3.3 Working Flow Chart of Fixed size Chunking Algorithm

3.4 SHA algorithm

In this algorithm, the output of fixed level chunking algorithm is taken as input in form of bytes.These bytes input are converted into hexadecimal numbers to perform the XOR operation by adding some random functions to generate a key for generating the secure hash value of chunks. The outputted secure hash value is used for checking the data duplicity in chunks.

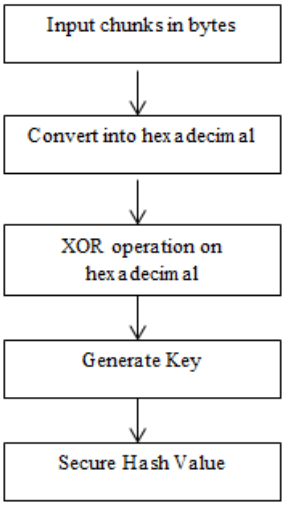

Figure 3.4 Flow Chart of SHA Algorithm

4. RESULTS AND ANALYSIS

Hadoop is used to implement proposed framework. It is open source software. Hadoop provides Hadoop distributed framework system to store data and process it mapper and reducer is used.

Metrics used:

4.1 Chunk time: Chunk time is used to measure the amount of time taken to perform chunk operations.

4.2 Hash time: It is the total time taken to perform hash values of each chunk.

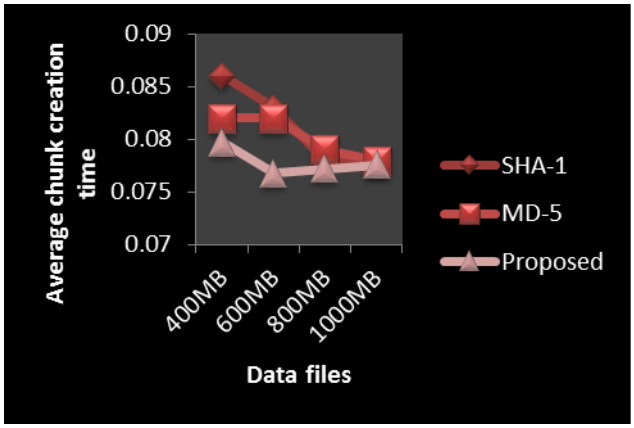

Figure 4 Average Chunk Creation Time

Figure 4 shows average chunk creation time of propose technique and existing SHA-1 technique. Here in this figure X-axis show data size which is varied from 400MB to 1000MB results show that proposed mechanism has less average chunk creation time as compared to existing techniques.

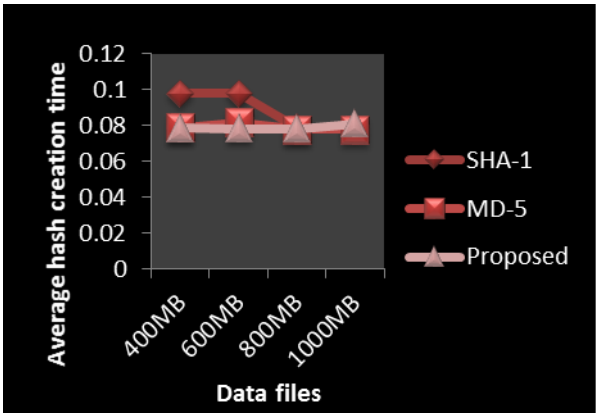

Figure 5 Average Hash creation time

Figure 5 shows average hash creation time of propose technique and existing SHA-1 technique, results show that proposed mechanisms have less average hash creation time as compared to existing technique.

5. CONCLUSION

Deduplication is used to find duplicate contents from large amount of data storage. Detection of duplicate content is a challenging task. To solve this task this paper provides secure

data deduplication framework for big data storage. To implement proposed framework Hadoop tool is used. The proposed framework is compared with existing algorithms such as MD-5 and SHA-1. In proposed framework average chunk creation time is low as compare to exiting security algorithms also proposed work has low hash time as compare to other security algorithms. In future continue working on it and enhance proposed framework to get less execution time also try to apply proposed framework in some real life applications.

REFERENCES

[1] B. Mao, H. Jiang, S. Wu, and L. Tian, "Leveraging data Deduplication to improve the performance of primary storage systems in the cloud", IEEE Transaction on Computers , vol. 65, issue. 6, pp. 1775–1788, 2016.

[2] J. Hur, D. Koo, Y. Shin, and K. Kang, "Secure Data Deduplication with Dynamic Ownership Management in Cloud Storage", IEEE Transactions on Knowledge Data Engineering., vol. 28, issue. 11, pp. 3113–3125, 2016.

[3] S. Luo, G. Zhang, C. Wu, S. Khan, and K. Li, "Boafft: Distributed Deduplication for big data storage in the cloud", IEEE Transaction on Cloud Computing., vol. pp, issue. 99, pp. 1–1, 2015.

[4] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-Locked Encryption and Secure Deduplication, in Advances in Cryptology", – EUROCRYPT 2013, T. Johansson and P. Nguyen, Editors, Springer Berlin Heidelberg. pp. 296-312, 2013.

[5] M .Bellare, S. Keelveedhi, and T. Ristenpart, "DupLESS: server-aided encryption for deduplicated storage", in Proceedings of the 22nd USENIX conference on Security., USENIX Association: Washington, pp. 179-194, 2013

[6] D. Harnik., B. Pinkas, and A. Shulman-Peleg, "Side channels in cloud services, the case of deduplication in cloud storage", IEEE Security & Privacy, vol.8, issue.6, pp. 40-47, 2010.

[7] Z. Yan, W. Ding, X. Yu, H. Zhu, and R. H. Deng, "Deduplication on Encrypted big data in cloud", IEEE Transaction on Big Data, vol. 2, issue 2, pp. 138–150, 2016.

[8] J. Wang, Z. Zhao, Z. Xu, H. Zhang, L. Li, and Y. Guo, "I-sieve: An inline high performance deduplication system used in cloud storage", Tsinghua Science and Technology, vol. 20, issue 1, pp. 17–27, 2015.

[9] Deepu. S R, Bhaskar. R, and Shylaja. B S, "Performance Comparison Of Deduplication Techniques For Storage In Cloud Computing Environment", Asian J. of Comput. Sci. Inf. Technol., vol. 4, issue.5, pp. 42–46, 2014.

[10] S. Sun, Y. Shi, S. Zhang and L. Cui, "A Privacy Protection Mechanism for No Sql Database Based on Data Chunks", IEEE Trust Com-Big Data SE-ISPA, pp.829-836, 2016.

[11] W. Xia, H. Jiang, D. Feng and L. Tian, "Combining Deduplication and Delta Compression to Achieve Low-Overhead Data Reduction on Backup Datasets", Data Compression Conference, pp. 203-212, 2014.

[12] Tin-Yu Wu, J. Shyang Pan and C.-Fan Lin, "Improving Accessing Efficiency of Cloud Storage Using De Duplication and Feedback Schemes", IEEE Systems Journal, vol 8 ,Issue 1, pp. 208-218, 2014.

[13] N. Kumar and Shalini, "Estimation of Secure Data Deduplication in Big Data", International Journals of Advanced Research in Computer Science and Software Engineering, vol.7,issue 6, pp. 702-705,2017.

[14] G. Lu, Y. Jin Nam and D. H.C. Du, "BloomStore: Bloom-Filter based Memory-efficient Key-Value Store for Indexing of Data Deduplication on Flash", IEEE 28 Symposium on Mass Storage Systems and Technologies San Diego, CA, USA, pp.1-11, 2012.

[15] Z. Yan, W. Ding, X. Yu, H. Zhu, and Robert H. Deng, "Deduplication on Encrypted Big Data in Cloud", IEEE Transactions on Big Data, vol. 2, issue. 2, pp.138-150, 2016.

[16] Y. Fu, N. Xiao, H. Jiang, G. Hu, and W. Chen, "Application-Aware Big Data Deduplication in Cloud Environment", IEEE Transactions On Cloud Computing, vol. pp, issue 99, pp.1-14, 2017.