International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 5 Issue: 5 1045 – 1052

A Survey on Data Deduplication

Shubhanshi Singhal
M.Tech., (CSE Deptt.)
University Institute of Engineering &Technology
Kurukshetra, India
Shubhanshi17@gmail.com

Naresh Kumar
Assistant Processor (CSE Deptt.)
University Institute of Engineering &Technology
Kurukshetra, India
naresh_duhan@rediffmail.com

***Abstract*—** Now-a-days, the demand of data storage capacity is increasing drastically. Due to more demands of storage, the computer society is attracting toward cloud storage. Security of data and cost factors are important challenges in cloud storage. A duplicate file not only waste the storage, it also increases the access time. So the detection and removal of duplicate data is an essential task. Data deduplication, an efficient approach to data reduction, has gained increasing attention and popularity in large-scale storage systems. It eliminates redundant data at the file or subfile level and identifies duplicate content by its cryptographically secure hash signature. It is very tricky because neither duplicate files don't have a common key nor they contain error. There are several approaches to identify and remove redundant data at file and chunk levels. In this paper, the background and key features of data deduplication is covered, then summarize and classify the data deduplication process according to the key workflow.

Keywords- *Deduplication, Chunking, Hashing, CDC.*

*****

I. INTRODUCTION

With the enormous growth of digital data, the need of storage is continuously increasing. As the amount of digital information is increasing, the storage and transfer of such information is becoming a very challenging task [1]. The new kind of storage that is gaining much attention in current scenario is cloud storage. One side, the digital data is increasing, other side backup problem and disaster recovery are becoming critical for the data centres. The three quarter of digital information is redundant by a report of Microsoft research. This massive growth in storage environment is balanced by the concept of deduplication. Data deduplication is technique to prevent the storage of redundant data in storage devices. Data deduplication is gaining much attention by the researchers because it is an efficient approach to data reduction. Deduplication identifies duplicate contents at chunk-level by using hash functions and eliminates redundant contents at chunk level. According to Microsoft research on deduplication, in their production primary and secondary memory are redundant about 50% and 85% of the data respectively and should be removed by the deduplication technology.

The two important aspects to evaluate the deduplication system are:

i) The deduplication ratio and ii) performance.

Deduplication can be performed at chunk level or file level. Chunk-level deduplication is preferred over file level deduplication because it identifies and eliminate redundancy at a finer granularity. Chunking is one of the main challenges in the deduplication-system. Efficient chunking is one of the key elements that decide the overall deduplication performance.

The chunking step also play a significant impact on the deduplication ratio and performance. The chunking and the chunk fingerprinting are the steps that are CPU intensive whereas fingerprint indexing and querying as well as data storing and management step require a lot of memory and disk resources. S. He et al. improve the performance of fingerprint indexing, querying, data storing and management steps by reducing the memory and disk resources consumption [2].

An effective chunking algorithm should satisfy two properties. Chunks must be created in a content-defined manner and all chunk sizes should have equal probability of being selected. Chunk level deduplication scheme divides the input stream into chunks and produce hash value that identifies each chunk uniquely. Chunking algorithm can divide the input data stream into chunk of either fixed size or variable size. If the fingerprints are matched with previously stored chunk, then these duplicated chunks are removed otherwise these are stored.

In this survey, section 2 covers the Deduplication Tool. In section 3, Data Deduplication is presented in details. Section 4 discusses Deduplication Storage System. In section 5, Encryption in Deduplication and Chunking Techniques are discussed. Section 6 covers the Hash Mechanism, section 7 represents Routing Strategies. In section 8, its Advantages and in last, paper is concluded with Conclusion in section 9.

II. DEDUPLICATION TOOL

The word Big data is introduced because the growth of data is drastic and its processing cannot be achieved using traditional data processing applications. The large volume of

 ISSN: 2321-8169
1045 – 1052

data- both structured and unstructured is termed as Big data [3]. For example: Billions of photos are daily shared on facebook from its users. Now a day, the availability of data is increasing day by day. Everyone has atleast 1TB data storage capacity. The characteristics of Big data is shown in figure 1.

A. Hadoop

Hadoop[4]is an open source java based programming framework. It was created by computer scientist Doug Cutting and Mike Cofarella in 2006. It is emerged as a foundation for big data processing tasks such as scientific analytics, business planning etc. It provides the parallel processing capability to handle big data. It is composed of four functional modules. Figure 2 shows the modules of Hadoop. Hadoop kernel provides the framework of essential libraries required by other Hadoop module. Hadoop Distributed File System (HDFS) is capable of storing data across thousands of servers. It was developed using distributed file system. It holds very large amount of data distributed across multiple machines. These files are stored in redundant fashion across multiple system to prevent system from data losses. Hadoop Yet Another Resource Negotiator (YARN) provides resource management and scheduling for user applications. Hadoop Map Reduce is a YARN based system for parallel processing of large data sets.

III. DATA DEDUPLICATION

Data deduplication is the process of eliminating the redundant data[5][6].It is the technique that is used to track and remove the same chunks in a storage unit. It is an efficient way to store data or information. The process of deduplication is shown in figure 3. Deduplication is generally divided into three parts i.e. Data unit based, Location based and Disk placement based. The classification of deduplication is shown in figure 4. It is preferably used for platform virtualization and backup server because at both the places many identical copies are used and produced.

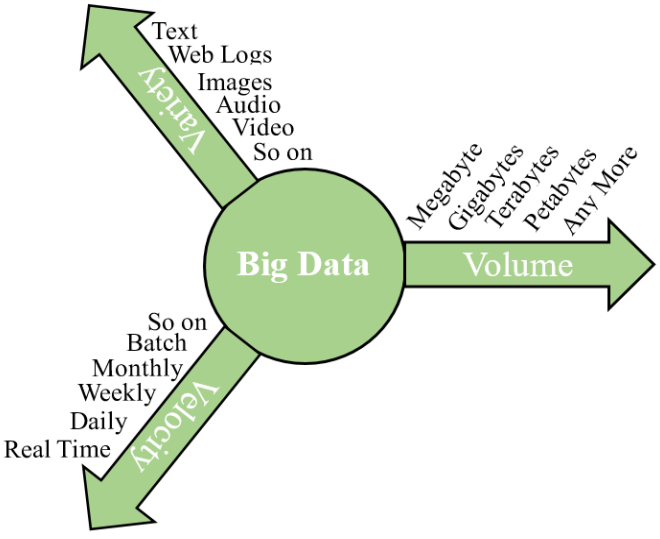

Figure 1. Characteristics of Big Data

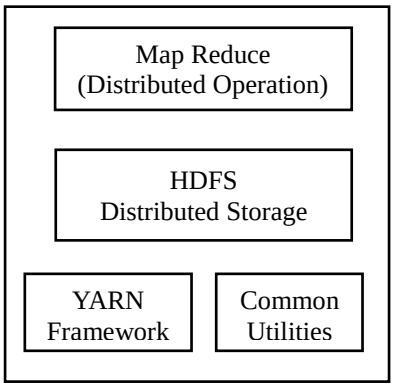

Figure 2. Modules of Hadoop

Data deduplication strategies are classified into two parts i.e. File level deduplication and Block (chunk) level deduplication. In file level deduplication, if the hash value for two files are same then they are considered identical. Only one copy of a file is kept. Duplicate or redundant copies of the same file are removed. This type of deduplication is also known as single instance storage (SIS). In block (chunk) level deduplication, the file is fragmented into blocks (chunks) then checks and removes duplicate blocks present in files. Only one copy of each block is maintained. It reduces more space than SIS. It may be further divided into fixed length and variable length deduplication. In fixed length deduplication, the size of each block is constant or fixed whereas in variable length deduplication, the size of each block is different or not fixed.

In location based deduplication, deduplication process is performed on different location. It is further divided into two parts i.e. Source based duplication and Target based deduplication. In source based deduplication, the deduplication is performed on client side or where the data is created. Before the transmission of actual data to the target, deduplication is performed. It reduces the huge amount of the backup data that would be sent through network. In this process, duplicate data is identified before the transmission over the network. It creates extra burden on the CPU but required fewer bandwidth for transmission and reduces the load on the network. It is further divided into two parts i.e. Local chunk deduplication and Global chunk deduplication. In

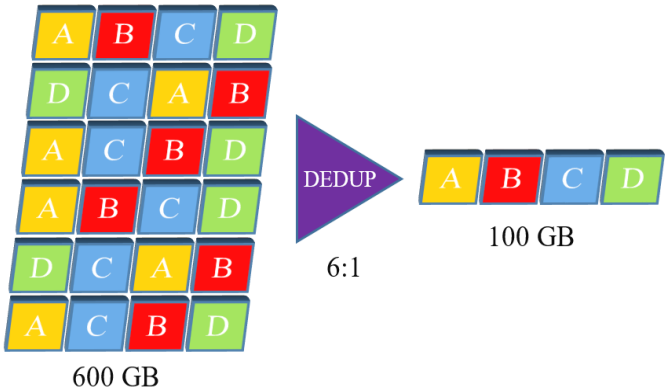

Figure 3. The Deduplication Process

ISSN: 2321-8169
1045 – 1052

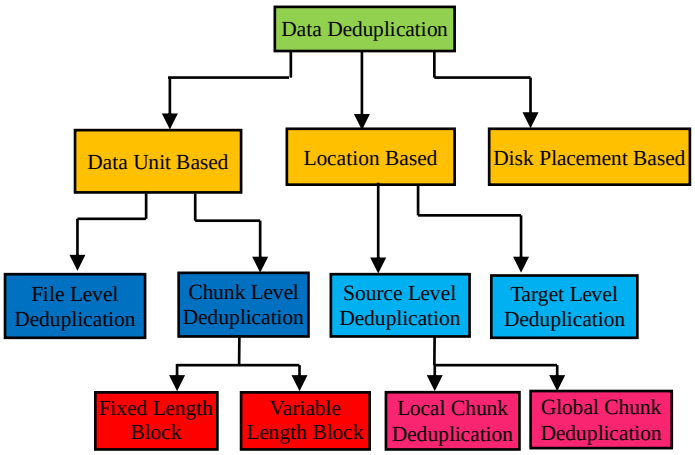

Figure 4. Classification of Deduplication Process

the local chunk level, redundant data is removed before sending it to the destination. In the global chunk level, redundant data is removed at global site for each client.

In target level, deduplication process is performed through mobile agents at the backup storage. The mobile agent tracks the redundancy at backup server and then only the unique data blocks are stored to the disk. The deduplication is performed on target i.e. data storage centre and the source is fully unaware about what is going at target. It does not reduce the amount of data transmitted. its main advantage is that clients are saved from unwanted overhead of the deduplication process.

Disk placement based deduplication is based on how data is to be placed on disk. Forward reference or backward reference are techniques that are used for this process. In forward reference, recent data chunks are maintained and all the old data chunks are associated to the recent chunks through pointers. Backward reference introduces the more fragmentation for the past data chunks.

A. Data Deduplication- performance evaluator[5]

The efficiency or performance of any data deduplication application can be effectively measured by:

1. Dedupe ratio where dedupe ratio = size of actual data / size of data after deduplication
2. Throughput (Megabytes of data deduplication / Second)

B. Algorithm/ Working Strategy [5]

The deduplication process removed blocks that are not unique. The working strategy of Deduplication process is shown in figure 5. The following steps are used for this process.

1. Divide the input data into chunks or blocks.
2. Hash value for each block needs to be calculated.
3. The values that are generated is used to check whether the blocks of same data is present in another stored block data.
4. If duplicate data is found then the reference to be created in database.
5. Based on the results, the duplicates data is eliminated. Only a unique chunk is stored.

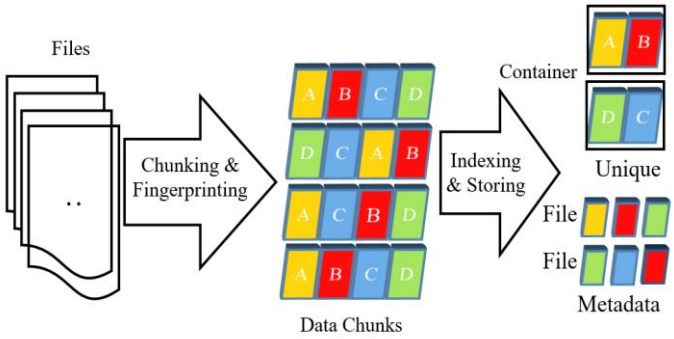

Figure 5. Deduplication workflow

IV. DEDUPLICATION STORAGE SYSTEM [7]

There are several deduplication storage systems that are preferred for different storage purposes.

Venti [8] is a type of network storage system. The deduplication technique followed by this storage system is based on the identification of identical hash values of the data blocks so that it reduces the overall usage of the storage space. It follows the write once policy to avoid collision of the data. This system is not efficiently deal with a vast amount of data and does not offer scalability.

HYDRAstor[9] is highly scalable secondary storage solution. It is based on decentralized hash index for the grid of storage nodes at the backend and traditional file interface at front end. It efficiently organizes large scale, variable size and content addressed, highly resilient data blocks with the help of Directed Acyclic Graph.

Extreme Binning [10] uses paralleled deduplication approach which aims at a non-traditional backup workload. It prefers the usage of the file similarity over the locality. The mechanism involves arranging similar data files into bins and removes duplicated chunks from each bin. It also keeps only primary index memory to reduce RAM assumption.

MAD2 [11] is famous and widely used for its accuracy. It provides an accurate deduplication backup service which primarily works on both at the file level and at the block level. The methodology involves the following techniques – a hash bucket matrix, a bloom filter array, a distributed Hash table based load balancing and dual cache, in order to achieve the desired performance.

Duplicate Data Elimination (DDE) [12] exactly calculates the corresponding hash values of the data blocks before the actual transformation of data at the client side. It works on the combination of copy-on-write, lazy updates & content hashing to identify and coalesce identical blocks of data in SAN system.

V. ENCRYPTION IN DEDUPLICATION [13]

Deduplication works by removing the redundant blocks, files or data. In whole file hashing, an entire file is first sent to hashing function. The preferred hashing functions are MD5 and SHA-1. Its result is a cryptographic hash which forms the

 ISSN: 2321-8169
1045 – 1052

basis for the identification of entire duplicate file. The whole file hashing has fast execution with low computation and low metadata overhead. But it prevents from matching two files that only differ by a byte of data. Subfile hashing divides the file into a number of smaller pieces before the actual data deduplication. The two types, fixed-size chunking and variable length chunking is used for dividing a file. In fixed size chunking, a file is divided into a number of static or fixed sized pieces called 'Chunks' whereas in variable length chunking, a file is broken down into chunks of varying length. The cryptographic hash function is applied to the broken pieces of file to calculate the 'Chunk Identifier'. The chunk identifier is the key parameter in locating redundant data.

A. *Delta Encoding (DE)* [14]

It is used to calculate the difference between a target object and a source object. Let block A is the source object and B is the target object then DE of B is the difference between A and B that is unique to B and not A. The DE is mostly used when there is a high similarity between the two items or blocks. By this way, the storing of difference would take less space than storing the non-duplicate block.

B. *Chunking based Data Deduplication*[15]

In this method, each file is firstly divided into a sequence of mutually exclusive blocks, called chunks. Each chunk is a contiguous sequence of bytes from the file which is processed independently. Various approaches are discussed in details to generate chunks.

1. *Static Chunking* [16]

The static or fixed size chunking mechanism divides the file into same sized chunks. Figure 6 shows the working of static size chunking. It is the fastest among other chunking algorithms to detect duplicate blocks but its performance is not satisfactory. It also leads to a serious problem called "Boundary Shift". Boundary shifting problem arises due to modifications in the data. If user inserts or deletes a byte from the data, then static chunking will generate different fingerprints for the subsequent chunks even though mostly data in file are intact.

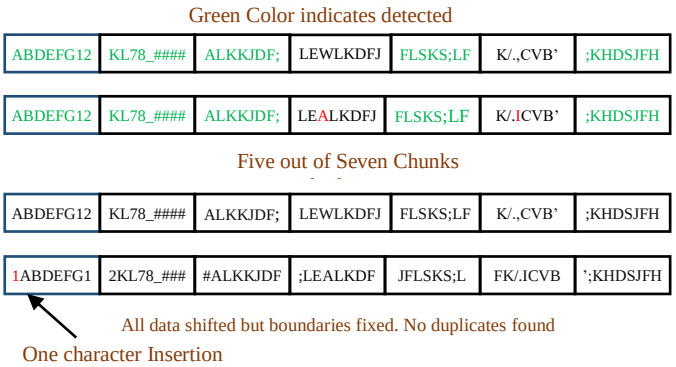

Figure 6. Fixed Size Chunking

2. *Content Defined Chunking*[16]

To overcome the problem of boundary shift, content defined chunking (CDC) is used. CDC reduces the amount of duplicate data found by the data deduplication systems, instead of setting boundaries at multiples of the limited chunk size. CDC approach defines breakpoints where a specific condition becomes true. The working of CDC is shown in figure 7. In simple words, It can be said that CDC approach determine the chunk boundaries based on the content of the file. Most of the deduplication systems use the CDC algorithm to achieve good data deduplication throughput. It is also known as Variable size chunking. It uses the Rabin fingerprint algorithm for hash value generation. It takes more processing time than the fixed size chunking but has better deduplication efficiency.

3. *Basic Sliding Window (BSW)* [17]

BSW uses the prototype of the hash-breaking (non-overlap) approach. The three main parameters needed to be pre-configured are: a fixed size of window W, an integer divisor – D, and an integer remainder – R, where R < D. In BSW method, on every shifts in the window, fingerprint will be generated and check for the chunk boundary. It is shown in figure 8.

4. *Improved Sliding Window* [18]

It has the advantage of fixed chunk size k byte. If the fingerprints of new chunk have same hash value, then consider it as redundancy and move the window forward over this data chunk otherwise assumption is made that the chunk may be duplicate that has been offset changed by insertion and deletion.

5. *Small Chunk Merge (SCM)* [19]

In this algorithm, chunk boundary is decided by sliding window according to the match of content of data. Repeatedly merge chunks with adjacent chunk whose size is less than a threshold value. By merging chunks in this way, we can eliminate chunk size variance problem.

6. *Breaking Large Chunk into Fixed Size Sub chunks* [19]

In this, first basic sliding window is used to decide the chunk boundary according to match of content of data then for each chunk whose size is bigger than threshold value. Break it

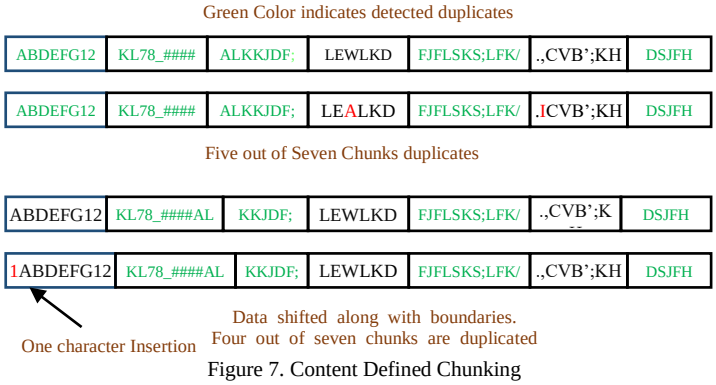

Figure 7. Content Defined Chunking

 ISSN: 2321-8169
1045 – 1052

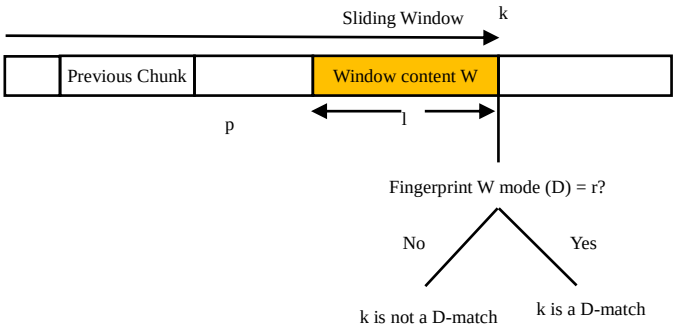

Figure 8. Basic Sliding Window

into the sequence of sub chunks of fixed size where the size of each chunk is less than or equal to threshold value. It reduces the variability of chunk size when the large chunks are broken into fixed size chunks but then boundary shifting problem is reintroduced.

7. Two Divisor [20]

BFS algorithm fails to improve due to boundary shifting problem. To overcome this problem two divisor algorithm is introduced. Three terms are used in the case of two divisor algorithm: K – Chunk Boundary, D –Main Divisor, D'– Backup Divisor

8. Two Threshold Two Divisor Chunking (TTTD) [19]

TTTD algorithm uses the same idea as the BSW algorithm but four parameters the Minimum Threshold, the Maximum Threshold, the Main Divisor, and the Second Divisor are used. Minimum Threshold is used to eliminate the vary large size chunks and the Maximum Threshold is used to eliminate very small-sized chunks. The main divisor determines the breakpoint and the second divisor value is the half of the main divisor. The second divisor finds the breakpoint when the main divisor cannot find it. The breakpoint found by second divisor are larger and very close to maximum threshold. The working of TTTD is shown in figure 10.

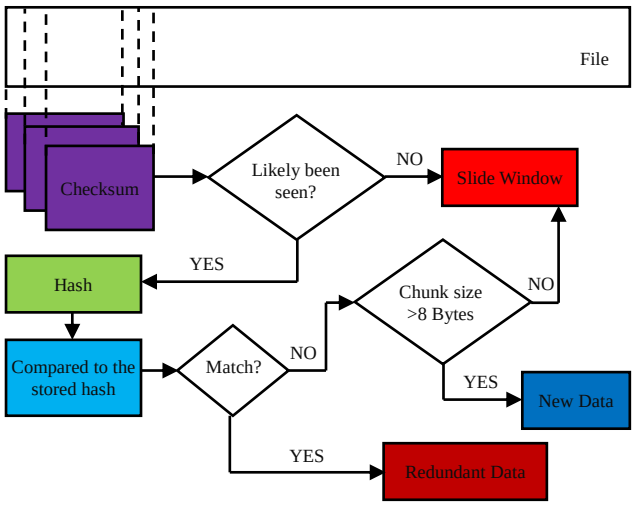

Figure 9. Improved Sliding Window

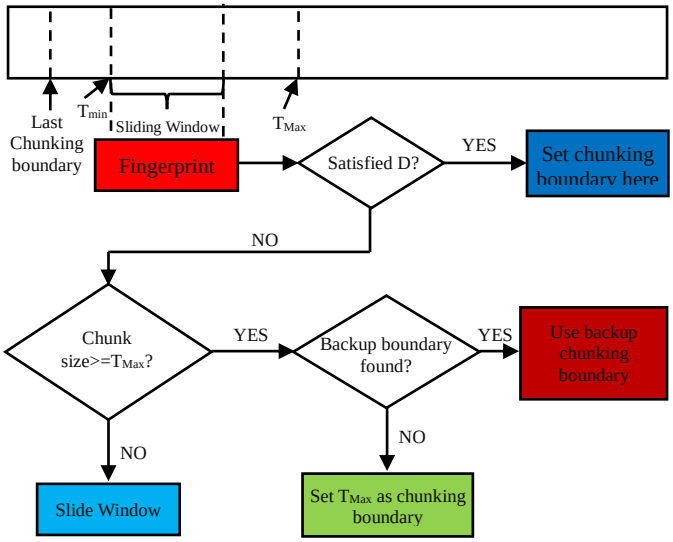

Figure 10. Two Threshold Two Divisor

9. Two Threshold Two Divisor – S [21]

This method overcomes the drawback of TTTD. It introduces a new parameter S which is the average of the maximum threshold and minimum threshold. It finds the chunks of average size. When the average threshold value is reached then both primary and secondary divisor shrink to half. Primary and secondary divisor get restored after chunking is completed.

10. Byte Index Chunking [22]

Byte Index Chunking is source based deduplication network file system. In source based network file system deduplication is performed on client side. Index table is exchanged between source and server until the non-deduplicated blocks are transferred for storage. There may be

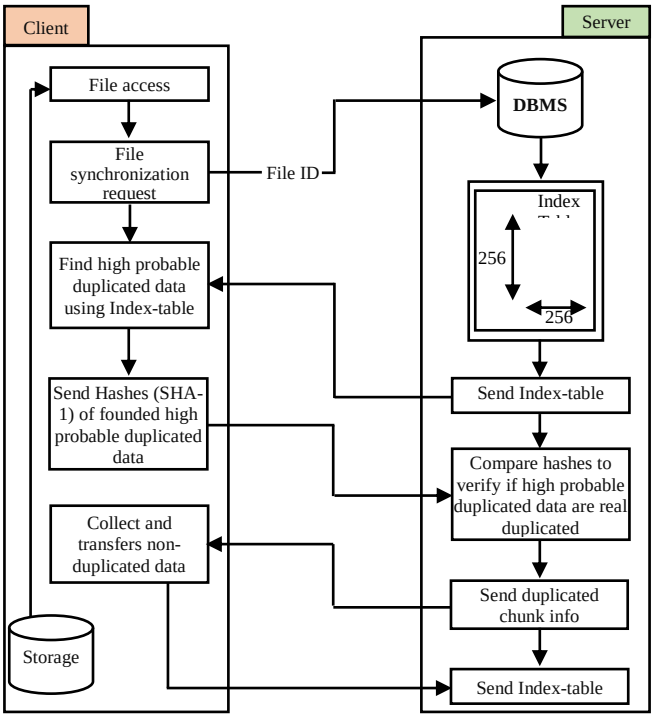

Figure 11. Byte Index Chunking

 ISSN: 2321-8169
1045 – 1052

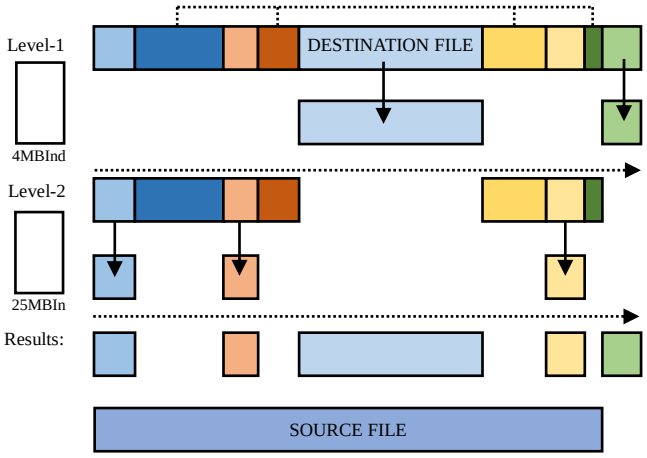

Figure 12. Multi Level Byte Index Chunking

occasion when a small of these chunks are not proven to be deduplicated then use lookup process to find duplicated chunks. Figure11 shows the working of Byte Index Chunking.

11. Multi-level Byte Index Chunking [23]

If the chunks of large size are set, then it will consume less time in processing but deduplication speed decreases and vice-versa. In multilevel byte index chunking entire file is divided into multiple size of chunks according to the size of file. If the file size is lower than 5GB then file deduplication process is done using 32KB sized index table. If the file size is more than 5GB then after retrieving data using 4MB chunk size, then start retrieving process using 32 KB chunk size index table. It is much better than fixed size chunking and CDC. The working of Multi-level Byte Index Chunking is shown in figure 12.

12. Frequency Based Chunking Algorithm [24]

FBC approach is a hybrid chunking algorithm that divides the byte stream according to the chunk frequency. First, it identifies the fixed size chunks with high frequency using bloom filter then the appearance is checked for each chunk in the bloom filter. If the chunk exists in bloom filter, then it is passed through the parallel filter otherwise record it in one of bloom filter then count the frequency for each chunk from parallel filter. Its working is shown in figure 13.

13. Improved Frequency based Chunking [25]

It overcomes the drawback of FBC algorithm. In this, CDC algorithm is used to chop the byte stream into chunks. This process leads to increase in amount of metadata and it does not allow re-chunking when the chunk size is fixed.

14. Bimodal Content Defined Chunking (BCDC) [26]

A major disadvantage of CDC is that the chunks of very small size or large size are produced. The chunks of small size leads to more CPU overhead and large size leads to decrease deduplication rate. To overcome this problem, Bimodal CDC algorithm is introduced. It is one of the appropriate chunking algorithm that decides whether to select the chunks of very small size or large size. BCDC provides the functionality of

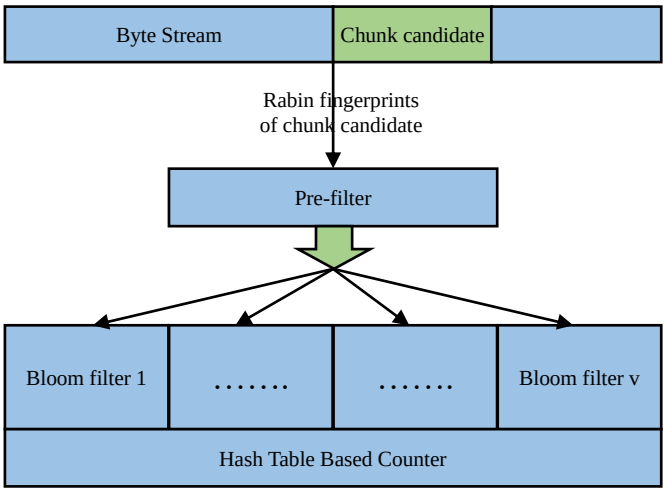

Figure 13. Frequency Based Chunking

merging the small chunks with adjacent chunks thus avoids the very small size chunks to reduce the CPU overhead.

15. Multimodal Content Defined Chunking (MCDC) [27]

In MCDC, the chunk's size is decided according to compressibility information. Large files with low compressibility are barely modified for large chunks and files with high compressibility result into chunks of small size. In this first of all, the data objects are divided into the fixed size chunks then the compression ratio is estimated of fixed size chunks. After this estimation, mapping table is created which maps compression range to chunk size.

16. Asymmetric Extremum (AE) CDC [28]

To address the challenges of CDC algorithm, a new algorithm called as Asymmetric Extremum CDC is employed. This method employs an asymmetric sliding window to define cut points. An asymmetric sliding window effectively finds the local extreme value to speed-up CDC. It resolves the low chunking throughput that renders the chunking stage deduplication performance bottleneck.

17. Fast CDC [29]

This approach eliminates the chunks of very small size. It is 10 times faster than Rabin based CDC and 3 times faster than AE based approach. Instead of using Rabin fingerprint hash mechanism, it generates the hash value for each byte. It normalizes the chunk size distribution to small specified region. The key idea behind the performance of fast CDC is use of combination of three techniques: 1. Optimizing hash judgement 2. Subminimum chunk cutpoint skipping 3. Normalized Chunking. The modules of Fast CDC is shown in figure 14.

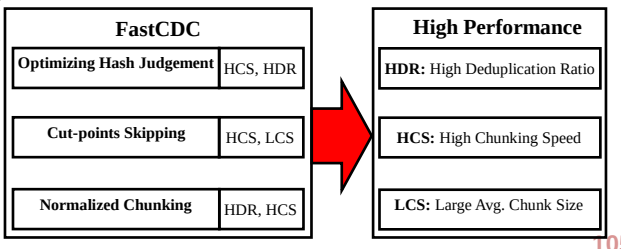

Figure 14. FastCDC

 ISSN: 2321-8169
1045 – 1052

VI. HASHING MECHANISM FOR DEDUPLICATION [30]

Hash collisions are potential problem with deduplication. The hash number for each chunk is generated using an algorithm. Some commonly used hashing algorithms are Rabin's algorithm, Alder-32, Secure Hash Algorithm – 1 (SHA-1) and Message Digest algorithm (MD-5). Advanced Encryption Standard (AES) is rarely used for this purpose. The encrypted data is out-sourced to the cloud environment. Rabin's fingerprinting scheme generate fingerprints using polynomials. It creates cryptographic hash of each block in a file. Rabin's fingerprint is an efficient rolling hash, because the computation of the Rabin fingerprint of Block B can reuse some of the computation of Rabin fingerprint of Block A when block A and B overlap.

Alder-32 was invented in 1995. It calculates two 16-bit checksums A and B and concatenating their bits into a 32-bit integer. It can be forged easily and unsafe for protecting against intentional modifications. MD5 is more secure than Alder-32.The abbreviation 'MD' stands for "Message Digest." MD5 process a variable length message into a fixed-length output of 128 bits. SHA-1 produces a 160-bit hash value. It is impossible to recreate a message from a given fingerprint. It generates a collision free hash value.

VII. ROUTING [31]

Cluster deduplication mainly consist of -Backup client, Metadata management, Deduplication server node. There are two types of routing mechanisms which are used to route the data packet i.e. stateless routing mechanism and state-full routing mechanism. In stateless routing the hash table is used to determine the path to server node. Statefull routing requires the information of deduplication server node to route the packet to the server node that has the same packet.

A. *Extreme Binning*[10]

It uses the stateless routing mechanism to find the deduplication server node. It selects the minimum of fingerprint then route the data packet. It achieves the low deduplication throughput but it cannot perform the load balancing well.

B. *Dedupe*[6]

To overcome the problem of load balancing Dedupe technique is developed. It selects the deduplication server node and check the similarity of chunk before sending the chunk into server node. It achieves high deduplication rate and does the load balancing well but there is a more communication overhead.

C. *AR Dedupe* [32]

To overcome the problem of load balancing and communication overhead AR dedupe technique is developed. In AR Dedupe, routing server is used. Hash table including hash information and id of deduplication server node of this superchunk is stored on routing server then superchunk is routed according to this id. AR Dedupe mainly consist of four parts backup client, metadata server, deduplication server node, routing server.

D. *Boafft* [33]

Boaftt is used in cloud storage for deduplication. The working of boafft is explained in following steps: First, it uses the routing algorithm to determine the address of node. Then it maintains the indexing in data server so as to avoid the duplicate read-write. In last, the deduplication is improved with the help of cache hot fingerprint. Boaftt uses the hash function or Jaccard coefficient to measure the similarity between datasets.

Client divides the data stream into smaller chunks. Fingerprints are created for each chunk separately and organize these chunks into super blocks. Boaftt finds the routing address for these chunks as the feature fingerprint by interacting with meta data server and route the superblock to appropriate data server. Fingerprints are stored on data server. Boafft uses the container to store fingerprint and organize the fingerprint in similarity index then boafft compare the superblock with similarity index. It avoids the large amount of input output operation. The working of Boafft is shown in figure 16.

VIII. DATA DEDUPLICATION: ADVANTAGES

Data deduplication optimizes the storage and saves a lots of money by reducing the storage and bandwidth cost. This technique needs less hardware to store the data because the storage efficiency is increased that reduces the hardware cost too. It reduces the backup costs because buying and maintaining less storage will return us with the faster returns. If effectively increases the network bandwidth and improves the network efficiency.

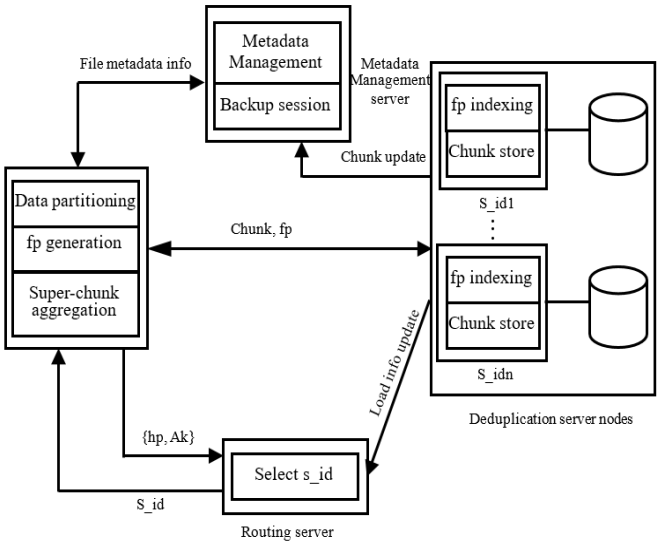

Figure 15. ARDedupe

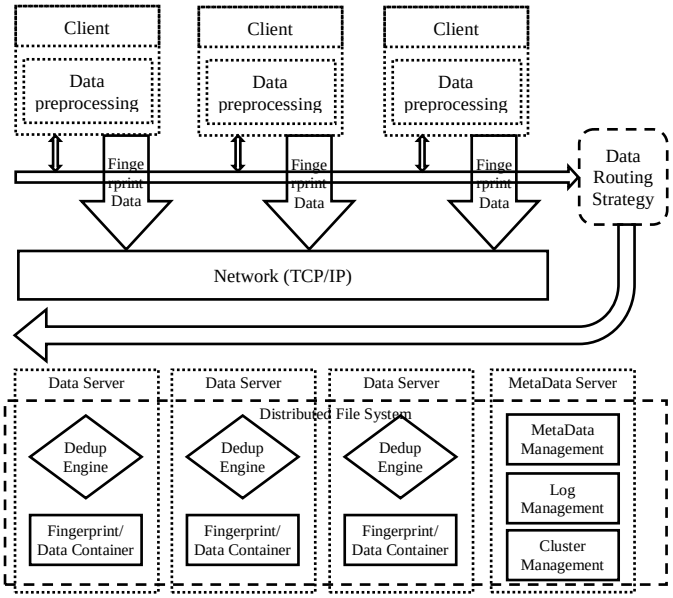

Figure 16. Boafft

IX. CONCLUSION

This paper mainly focuses on the Data Deduplication, its Chunking Techniques, and Storage Systems. The main aim of data deduplication is to remove redundant data and maintain unique copy of data files. Data deduplication is an emerging trend in market and good alternate of data compression. It is one of the intelligent storage saving mechanism. Data deduplication well manages the storage spaces and improves the network bandwidth which is required for transferring data.

References

[1] B. Cai, Z. F. Li and W. Can, "Research on Chunking Algorithms of Data Deduplication," in *International Conference on Communication, Electronics and Automation Engineering*, Berlin, Heidelberg, 2012.

[2] S. He, C. Zhang and P. Hao, "Comparative study of features for fingerprint indexing," in *16th IEEE International Conference on Image Processing (ICIP)*, Cairo, 2009.

[3] H. Fang, Z. Zhang, C. J. Wang, M. Daneshmand, C. Wang and H. Wang, "A survey of big data research," *IEEE Network*, vol. 29, no. 5, pp. 6-9, september 2015.

[4] M. Panda and R. Sethy, "Big Data Analysis using Hadoop: A Survey," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 5, no. 7, pp. 1153-1157, july 2015.

[5] J. Malhotra and J. Bakal, "A survey and comparative study of data deduplication techniques," in *International Conference on Pervasive Computing (ICPC)*, Pune, 2015.

[6] D. Meyer and W. Bolosky, "A Study of Practical deduplication," in *9th USENIX conference on File and Storage Technologies (FAST' 11)*, SAN JOSE, California, 2011.

[7] D. Bobbarjung, S. Jagannathan and C. Dubnicki, "Improving Duplicate Elimination in Storage Systems," *ACM Transactions on Storage (TOS)*, pp. 424-428, 2006.

[8] Q. Sean. and S. Dorward, "Venti: A new approach to archival storage," Bell Labs, Lucent Technologies.

[9] C. Dubnicki, L. Gryz, L. Heldt, M. Kaczmarczyk, W. Kilian, P. Strzelczak, J. Szczepkowski, C. Ungureanu and M. Welnicki, "HYDRAstor: A Scalable Secondary Storage," in *7th USENIX Conference on File and Storage Technologies (FAST 09)*, SAN FRANCISCO, California, 2009.

[10] D. Bhagwat, K. Eshghi, D. D. E. Long and M. Lillibridge, "Extreme Binning: Scalable, parallel deduplication for chunk-based file backup," in *IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems*, LONDON, 2009.

[11] J. Wei, H. Jiang, K. Zhou and D. Feng, "MAD2: A scalable high-throughput exact deduplication approach for network backup services," in *IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, Incline Village, NV, 2010.

[12] B. Hong, "Duplicate Data Elimination in a SAN File System," in *21st International Conference on Massive Storage Systems and Technologies (MSST)*, College Park, MD, 2004.

[13] K. Akhila, A. Ganesh and C. Sunitha, "A Study on Deduplication Techniques over Encrypted Data," in *Fourth International Conference on Recent Trends in Computer Science & Engineering*, Chennai, Tamil Nadu, India, 2016.

[14] B. Song, L. Xiao, G. Qin, L. Ruan and S. Qiu, "A Deduplication Algorithm Based on Data Similarity and Delta Encoding," in *International Conference on Geo-Informatics in Resource Management and Sustainable Ecosystems*, Hong Kong, China, 2016.

[15] W. Xia, H. Jiang, D. Feng, F. Douglis, P. Shilane, Y. Hua, M. Fu, Y. Zhang and Y. Zhou, "A comprehensive study of the past, present, and future of data deduplication," *Proceedings of the IEEE*, vol. 104, no. 9, pp. 1681--1710, 2016.

[16] A. Li, S. Jiwu and L. Mingqiang, "Data Deduplication Techniques," *Journal of Software*, vol. 2, no. 9, pp. 916-929, 2010.

[17] W. Hsu and S. Ong, "System and Method for Dividing Data into Predominantly Fixed-sized Chunks So That Duplicate Data Chunks May be Identified". US Patent Patent US7281006B2, 207.

[18] C. Bo, Z. F. Li and W. Can, "Research on chunking algorithms of data de-duplication," in *International Conference on Communication, Electronics and Automation Engineering*, Berlin Heidelberg, 2012.

[19] K. Eshghi and H. K. Tang, "A framework for analyzing and improving content-based chunking algorithms," Hewlett-Packard Labs, 2005.

[20] A. Venish and K. Siva Sankar, "Study of Chunking Algorithm in Data deduplication," in *international conference on soft computing systems*, Chennai, 2015.

[21] T. Moh and B. Chang, "A Running Time Improvement for Two Threshold Two Divisors Algorithm," in *ACM 48th Annual Southeast Regional Conference*, MS, USA, 2010.

[22] I. Lkhagvasuren, J. So, J. Lee, C. Yoo and Y. Ko, "Byte-index chunking algorithm for data dedupli-cation system," *International Journal of Security and its Applications*, vol. 7, no. 5, pp. 415-424, Oct 2013.

[23] I. Lkhagvasuren, J. So, J. Lee and Y. Ko, "Multi-level Byte Index Chunking Mechanism for File Synchronization," *International Journal of Software Engineering and Its Applications*, vol. 8, no. 3, pp. 339-350, 2014.

[24] G. Lu, Y. Jin and D. Du, "Frequency Based Chunking for Data De-Duplication," in *IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Miami Beach, FL, 2010.

[25] Y. Zhang, W. Wang, T. Yin and J. Yuan, "A Novel Frequency Based Chunking for Data Deduplication," *Applied Mechanics and Materials*, vol. 278, pp. 2048-2053, 2013.

[26] E. Kruus, C. Ungureanu and C. Dubnicki, "Bimodal Content Defined Chunking for Backup Streams," in *8th usenix conference on file and storage technologies (FAST-10)*, SAN JOSE, California, 2010.

[27] J. Wei, J. Zhu and Y. Li, "Multimodal Content Defined Chunking for Data Deduplication," Huawei Technologies, 2014.

[28] Y. Zhang, D. Feng, H. Jiang, W. Xia, M. Fu, F. Huang and Y. Zhou, "A Fast Asymmetric Extremum Content Defined Chunking Algorithm for Data Deduplication in Backup Storage Systems," *IEEE Transactions on Computers*, vol. 66, no. 2, pp. 199-211, Febuary 2017.

[29] W. Xia, Y. Zhou, H. Jiang, D. Feng, Y. Hua, Y. Hu, Q. Liu and Y. Zhang, "FastCDC: A Fast and Efficient Content-defined Chunking Approach for Data Deduplication," in *2106 USENIX Conference on Usenix Annual Technical Conference*, Berkeley, CA, USA, 2016.

[30] G. Raj, "Deduplication Internals – Hash based deduplication : Part-2," [Online]. Available: https://pibytes.wordpress.com/2013/02/09/deduplication-internals-hash-based-part-2/.

[31] J. Paulo and J. Pereira, "A survey and classification of storage deduplication systems," *ACM Computing Surveys (CSUR)*, vol. 47, no. 1, 2014.

[32] Y. Xing, N. Xiao, F. Liu, Z. Sun and W. He, "AR-dedupe: An efficient deduplication approach for cluster deduplication system," *Journal of Shanghai Jiaotong University (Science)*, pp. 76-81, 2015.

[33] S. Luo, G. Zhang, C. Wu, S. Khan and K. Li, "Boafft: Distributed Deduplication for Big Data Storage in the Cloud," *IEEE Transactions on Cloud Computing*, vol. 99, pp. 1-1, 2015.