ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 8(26), DOI: 10.17485/ijst/2015/v8i26/80754, October 2015* ISSN (Online) : 0974-5645

Framework of Data Deduplication: A Survey

A. Venish[1*] and K. Siva Sankar[2]

[1]Computer Science and Engineering, Noorul Islam University, Kumarakoil – 629180, Tamil Nadu, India; venish07@gmail.com

[2]Information Technology, Noorul Islam University, Kumarakoil – 629180, Tamil Nadu, India; sivasankarniu@gmail.com

Abstract

To understand the concept and framework of de-duplication process along with application, various methods and technologies involved during the each level implementation of this process and about the Limitations. Different chunking algorithm such as fixed, variable and content aware chunking methods are used to decide the chunk size of deduplication. To avoid the single point failure in the distributed system, cluster model with parallel process can be used. Comparing with and without deduplication, we can save up to 75% of storage space in the backup system by using with deduplication. Digital data increase happens in all cloud deployment models and this requires more storage capacity, more costs, manpower and more time to handle data information like backup, replication and disaster recovery, more bandwidth utilization in transmitting the data across the network. If we handle the data effectively like remove the redundant data before storing into the storage device we can avoid data handling overhead and we can improve the system performance. By using data deduplication concept we can achieve the above results. The variable chunking method including content aware process yields good throughput in the deduplication system compare with fixed and whole file chunking method. Inline process avoids extra required spaces and increase system performance. Cluster model eliminates single point failure in the distributed deduplication system. Deduplications save more storage spaces, avoid tohandle unnecessary data over head and provides less resources utilization with minimal cost.

Keywords: Chunking, Cluster Deduplication, Data Deduplication, Duplicate Detection, Fingerprint Calculation

1. Introduction

Basically Data has been classified into two types namely ‘structured data’ and ‘unstructured data’ which are playing a major role in the recent trend. Normally the structure data can be easily organized includes website log data, customer call detailed records etc., Due to the rapid increase of social media usage and mobile usage the unstructured data cannot be easily organized includes blog data, social media interaction data, videos etc., So, unstructured data should be managed in a cost effective way. Today in IT budgets, on an average of 13% of the money being invested on storage capacity[1]. Data to grow more quickly says IDC’s Digital Universe study[2]. These impacts create more problems, like degradation of performance, compromise of quality, and more operational costs. So in order to overcome the above problems and handle the data growth in a good manner we need a specialized system where the concept of Deduplication is derived.

Deduplication technology looks into data either at a block level (sub file) or file level. The incoming data is split into smaller fixed or variable blocks or segments. Each of these smaller blocks is given a unique identifier which is created by several hashing algorithms or even a bit by bit comparison of the block. Common algorithms used for this process are MD5 or SHA-1.

Also content aware logic, which considers the content type of the data and finalize the size of block and boundaries.

As the deduplication system processes data, it compares the data to the already identified blocks and stores in its database. If a block already exists in the database, the new redundant data is discarded and a reference to the existing data is inserted into the repository. If the block

** Author for correspondence*

contains new, unique data, then the block is inserted into the data store (file system), and a reference is added to that block in the de-dupe database. The primary benefit of deduplication is that it greatly reduces storage capacity requirements, drives several other advantages like lower power consumption, lower cooling requirements, longer disk-based retention of data (faster recoveries), and disaster recovery.

The data partitioning, finger print calculation, finger print lookup and write the Finger prints into the database are the four major processes involved in the concept of deduplication.

2. Deduplication Framework

The deduplication framework has four important steps including data partitioning and extraction, finger print calculation and lookup, comparing the finger prints and write the finger prints into the database.

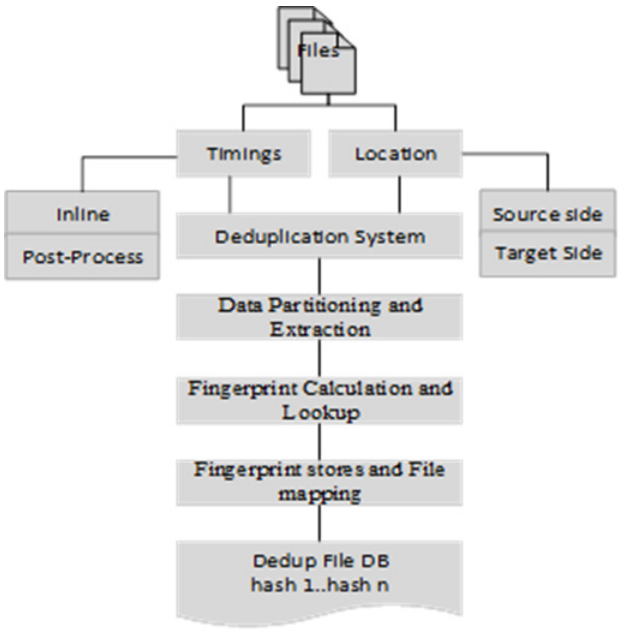

Figure 1. Framework of deduplication system.

Figure 1, explains the framework of deduplication. Deduplication process can be classified based on the timings, and location. Based on the timings deduplication can happen on the Post Process or Inline. In the Post Process techniques first data's are stored into the disk then it removes the redundancy data whereas Inline method first removes the redundancy data then stores it into the disk. The main advantages of the post process method is the deduplication algorithm completely understands the data which already stored in the storage system but this method consumes more disk spaces as it stores the complete data before start the deduplication. Compared with the post process method, the Inline method is adapting in to the most recent backup deduplication process, because extra disk space is not required as it is removing the redundant data on the fly.

Based on the location, deduplication can happen on the source side or target side. In the source side, redundant data will be detected and it will be moved to the target side, whereas target side deduplication redundant data will be detected on the target side not on the source side. There are several studies carried out to understand and analyze the above four steps. Using file level, static and content aware chunking Mandagere et al.[3] presented a first improved deduplication results with personal computer storage system, Meister and Brinkman analyze the deduplication study with collage documents as well as the unstructured data[4] and they explained how the file format make effects on the deduplication system. Karthigha[13] studied various methods to remove the deduplication in database. Also various studies presented deduplication results using different platforms such as Microsoft[5], EMC Data Domain installations[6] and virtual machine[7].

In the below sections we have discussed each and every steps of deduplication framework in detail.

2.1 Data Partitioning and Extraction

In the first step, the incoming data's are partitioning or dividing using the chunking algorithm. The chunk size has implications on number of chunk entries and the hash lookup so it can decide the deduplication ratio and the performance of the system[8]. Dividing data's into smaller chunks or segments are happening into two different major levels such as File-Level deduplication and Block-Level deduplication.

In the File-Level deduplication, the hash value will be created for each file using cryptographic hash algorithm such as MD or SHA-1[5,10,14] and the same value will be stored in the hash table. As it is creating only one hash value for each file, it takes minimal time to look up the hash value from the hash table but it fails and provides more duplicated date if the sub set of the files changes by only a single byte also we can expect more delay when it handles a large file. In the Block-Level deduplication the data's are divided into small chunks, each chunk will be given a hash value and the same will be stored into the index table. This method is very suitable for the distributed file system but the drawback of the system is, as it is creating more hash value, it is taking more memory space to store the hash index and more time to look up the hash index[15]. The division of chunks can be done on the basis of Algorithms called 'Fixed-Size Chunking (FSC)' and 'Variable-Size Chunking (VSC)'. The Fixed-Size Chunking divides the input file into equal size (4k, 8k, 16k etc.)[16,17] regardless of the content of the file. This method is taking less time to divide the file. Also it consumes less power. The drawback of this method is if file changes in a single byte, the entire boundary will be shifted and result in a new version of the file having very less duplicate chunks.

Venti[17] and DDFS[12] used the fixed size chunking method using SHA-1 cryptographic hash algorithm. The average chunk size used in the enterprise storage system is between 4KB and 16KB[17,18].

The Variable-Size Chunking divided the data into variable size chunks and the boundaries are finalized based on the content of the file not on the offset of the file by the finger print algorithm such as sliding window approaches[26] rolling hashes[27] Rabin fingerprints[25], and bimodal chunking[28]. This method overcomes the boundary shifting problem which is occurring in the fixed size chunking method.

Delta encoding is a one of the variable size chunking approach will record the change between a source file and a target file. As it stores only changed byte on storage, it will not store the file when files are identical or just small changes between almost same files. But it can be only performed on a pair of files as well as it has to remember the file or chunks that are used for delta encoding.

Basic Sliding Window (BSW)[31] which is fully depends on the finger print algorithm to find out the chunk boundaries. It divides the file if the break condition matches. But this method fails to provide the guaranteed maximum and minimum chunk size. But 'Two Threshold Two Divisor (TTTD)' chunking method[32] overcome basic sliding window problem by not producing chunks smaller than a certain threshold. Kruus et al.[28] uses chunks of two different size targets to eliminate the minimum and maximum chunk size problem.

Content defined chunking is another form of variable chunking method. Using this algorithm Spring et al.[19] adopts Border's[20] method to identifying redundant network traffic. Muthitacharoen et al.[21] presented a approach to remove the redundant data in low bandwidth network file system is helping to reduce the bandwidth, network traffic and uses less storage system. You et al.[22] proposed data removal in an archival storage system. TAPER[23] and REBL[40] combined CDC and Delta encoding chunking algorithm for directory synchronization.

2.2 Fingerprint Calculation and Lookup

After data chunking has been done, the finger print creation and lookup has to be done with existing stored hash value to remove the de-duplication. Generally the identification of duplicate can be done by comparing data hash value or files bit by bit. These methods provide correct accuracy but taking some additional time. When we use the hash algorithm to find out the duplicate, the hash collision would be increased which depends on the hash algorithm. Thus choosing the hash algorithm is very important at this stage. Various studies carried out to provide the hash algorithm performance and its drawbacks.

The following Finger print list gives an overview over the existing well-known mechanisms and their characteristics:

Rabin fingerprints[25] using random polynomials over a finite field to calculate the hash value. Rolling Adler-32 uses a rolling hash function which is based on the Adler-32 checksum algorithm[29]. 'Pretty Light and Intuitive (PLAIN)' fingerprinting algorithm presented by Spiridonov et al.[30] useful for Low Bandwidth File System. When we come to the Fingerprint lookup, it is affecting I/O performance in the large-scale system. To improve the hash lookup efficiency there are various techniques proposed.

First To avoid the disk I/O when looking up the index, the more used techniques is a main memory filter called the 'Bloom filter'[33]. Hamilton et al. presented a new method to reduce the overhead of fingerprint lookup. Here he maintains fingerprints in hierarchical manner where a parent node's fingerprints arethe hash value of the fingerprints of the child nodes[34].

Second to reduce the number of fingerprints used in the comparison Lillibriged et al.[11] created Sparse Index that contains sampling chunks and reference of the chunks. It can reduce the memory occupation overhead and improve the deduplication ratio. Bobbarjung et al, proposed new concept called fingerdiff[24]. It maintains chunks in a hierarchical way and maintain group of chunks as a single unit using small size chunk e.g., 1KB. But it fails to work when chunk size change significantly. Aronovich et al.[35] proposed a method which is comparing and duplicating data based on the similarity with the existing data for that they maintained information summary in larger unit. Also Bhagawat et al.[9] proposed similarity based comparing the duplicate data.

Third to reduce the disk overhead in fingerprint lookup Zhu et al. he proposed a technique called SISL, where they simply append the incoming fingerprints at the end of existing table[12]. Spyglass[36] is a file metadata search system uses a hierarchical partitioning of the namespace

Fourth aspect is to enhance reliability of de-duplicated data, Liu et al.[37] proposed combination of fixed and variable size chunks yield better result in the deduplication reliability. Bhagawat et al.[38] explained applying different levels of replication for each chunk also provide good reliability into the duplication system. Efstathopoulos et al.[39] created grouped mark-and-Sweep techniques which is overcome the chunk garbage collection issue.

Comparing the finger prints and writes the finger prints into the database are last steps in the deduplication process. The most important point to remember is when we deal with single node deduplication system, the failover of single node possibility is very high, and hence the system fails to provide the accurate deduplication so we need some methodology to overcome this single node failure issue. Also to handle large amount of data, the distributed system with cluster model is very important.

2.3 Cluster Model

From recent growth of data it's apparent that a single-node deduplication system will not fulfil the deduplication ratio and scalability. A single node de-duplication system also can become a single point of failure. In this scenario, if the de-duplication server/appliance fails, it's catastrophic. This problem has led to the development of clustered deduplication systems in cloud domain which are designed with an intension to provide uninterrupted services.Some available cluster model based deduplication concepts have been proposed earlier.

Douglis et al.[42] present a load balancing/backup assignment model in which the backups are assigned to deduplication instances so that the intersection and similarity between backup runs is increased. DeDe[43] presented a method which each host creates and stores content summary and exchanging between other nodes. DEBAR[44] proposed a cluster with post processing method. 'HYDRAstore'[10] is a cluster model which is using Distributed Hash Table (DHT) and detects duplicate data. HANDS[46] use grouping techniques which is reducing main memory index lookup. It is considering two features like namely & time stamp where SAM[45] proposed 4 features like file locality, time stamps, size and type. Droplet[47] also trying to solve the single-node failure. It stores all chunk index into inside the RAM and sends input data segments into multiple storage nodes. Longxiang Wang[48] method introducing virtual memory cache replacement techniques instead of indexing table to expand the storage capacity without disturbing main memory. Can Wang[49] proposed hierarchical indexing structure which is not depending on the data locality but based on the similarity of the file. Guohua Wang[50] presented a structure where all nodes can do chunk level deduplication parallel. Hpod[51] focus on the routing strategy which is main factor for deduplication performance, the fault tolerance and cluster system stability. Kaiser[18] explain exact data deduplication cluster method using locality and load balancing techniques.

3. Deduplication Result

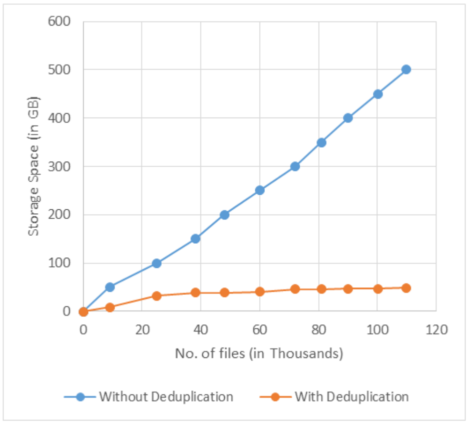

Figure 2. Comparison of without deduplication and with deduplication.

Using deduplication concept in the backup storage system we can save up to 75% storage space. Figure 2, compares without deduplication and with deduplication space savings efficiency[41]. If we store 500 GB data without deduplication, the traditional storage system requires 500 GB storage space but using with deduplication we need only 40 – 50 GB storage space instead of 500 GB. So we can save lot of spaces using the deduplication.

4. Conclusion

Basically the deduplication efficiency is measured by detection of deduplication and the cost saving parameter. The scalability is measured by the large amount of data handled by the deduplication system. The throughput will be measured by the rate at which data can be transferred in and out of the system. So a good deduplication system should achieve the above three factors without fails.

In our study we have explained the complete framework of the deduplication. Inline deduplication method avoids the extra storage space compare than the Post Process method. The data partitioning and extraction is very crucial steps in the deduplication process, so the choosing of chunking algorithm can decide the entire deduplication performance as well as the correct chunk size can improve the deduplication ratio and the throughput. There are various studies to create the fingerprint value for each chunk and enormous effort to reduce the fingerprint lookup timings. Also when we handle the huge amount of data in the cloud storage the distributed system with cluster concept can avoid the single point failure. The various cluster models has been discussed and effective way of chunk routing and parallel distribution of chunks to all the nodes can improve the data skew. Further research can help how to tune up critical nodes configuration which will increase the data skew.

In the deduplication process a number of factorsneed to be taking care to improve the total performance. Hash collision, false positive ratio, I/O processing, main memory utilization and data routing and distributing in the cluster model can affect the overall deduplication performance.

5. References

1. Stacy C. Big data storage doesn't have to break the bank. 2015 Mar 14. Available from: http://www.computerworld.com/s/article/9242951/Big_data_storage_doesn_t_have_to_break_the_bank?taxonomyId=19&pageNumber=2
2. Antony A. Data to grow more quickly says IDC's Digital Universe study. 2015 Mar 14. Available from: http://www.computerweekly.com/news/2240174381/Data-to-grow-more-quickly-says-IDCs-Digital-Universe-study.
3. Mandagere N, Zhou P, Smith MA, Uttamchandani S. Demystifying data deduplication. Proceedings of the Middleware Conference Companion'08, 2008. p.12–7.
4. Meister D, Brinkmann A. Multi-level comparison of data deduplication in a backup scenario. Proceedings of the 2nd Israeli Experimental Systems Conference (SYSTOR)'09; 2009. p.1–12.
5. Meyer DT, Bolosky WJ. A study of practical deduplication. ACM Transactions on Storage. 2012; 7(4):1–20.
6. Wallace G, Douglis F, Qian H, Shilane P, Smaldone S, Chamness M, Hsu W. Characteristics of backup workloads in production systems. Proceedings of the 10th USENIX Conference on File and Storage Technologies (FAST); 2012. p. 1–41.
7. Jayaram KR, Peng C, Zhang Z, Kim M, Chen H, Lei H. An empirical analysis of similarity in virtual machine images. Middleware'11 Proceedings of the Middleware 2011 Industry Track Workshop; 2011. p. 1–6.
8. Kiswany S, Ripeanu M, Vazhkudai SS, Gharaibeh A. stdchk: A checkpoint storage system for desktop grid computing. Proceedings of the 28th International Conference on Distributed Computing Systems (ICDCS); Beijing. 2008. p. 613–24.
9. Bhagwat D, Eshghi K, Long DDE, Lillibridge M. Extreme Binning: Scalable, parallel deduplication for chunk-based file backup. IEEE International Symposium on Modeling. Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '09); London, UK. 2009 Sep. p. 1–9.
10. Dubnicki C, Gryz L, Heldt L, Kaczmarczyk M, Kilian W, Strzelczak P, Szczepkowski J, Ungureanu C, Welnicki M. Hydrastor: Ascalable secondary storage. Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST); San Francisco, CA. 2009 Feb. p. 197–210.
11. Lillibridge M, Eshghi K, Bhagwat D, Deolalikar V, Trezise G, Camble P. Sparse indexing: Large scale, inline deduplication using sampling and locality. Proceedings of the 7th SENIX Conference on File and Storage Technologies (FAST); San Francisco, CA. 2009 Feb. p. 111–23.
12. Zhu B, Li K, Patterson H. Avoiding the disk bottleneck in the Data Domain deduplication files system. Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST). 2008 Feb. p. 1–14.
13. Anand KS. A Survey on Removal of Duplicate Records in Database. Indian Journal of Science and Technology. 2013; 6(4):4306–11.
14. Ungureanu C, Atkin B, Aranya A, Gokhale S, Rago S, Cakowski G, Dubnicki C, Bohra A. Hydrafs: A high-throughput file system for the Hydrastor content-addressable storage system. Proceedings of the 8th USENIX Conference on File and Storage Technologies. 2010. p. 225–38.
15. Bolosky W, Corbin S, Goebel D, Douceur J. Single instance storage in Windows 2000. Proceedings of the 4th USENIX Windows Systems Symposium; 2000. p. 13–24.
16. Kubiatowicz J, et al. Oceanstore: An architecture for global store persistent storage. Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems; 2000. p. 190–201.
17. Quinlan S, Dorwards S. Venti: A new approach to archival storage. Proceedings of USENIX Conference on File and Storage Technologies. 2002. p. 89–102.

18. Kaiser J, Meister D, Brinkmann A, Effert S. Design of an exact data deduplication cluster. Proceedings of the IEEE Conference on Mass Storage Systems and Technologies (MSST); San Diego, CA. 2012. p.1–12.
19. Spring NT, Wetherall D. A Protocol-Independent Technique for Eliminating Redundant Network Traffic. Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM'00). 2000; 30 (4). p. 87–95.
20. Broder AZ. On the resemblance and containment of documents. Proceedings of compression and complexity of sequences (SEQUENCES'97). Salerno. 1997; 21–9.
21. Muthitacharoen A, Chen B, Mazieres D. A low-bandwidth networks file system. Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01); 2001 Oct p. 174–87.
22. You LL, Pollack KT, Long DDE. Deep Store: An Archival Storage System Architecture. Proceedings of the 21st International Conference on Data Engineering (ICDE' 05); 2005 Apr. p. 804–15.
23. Jain N, Dahlin M, Tewari R. TAPER: Tiered Approach for Eliminating Redundancy in Replica synchronization. Proceedings of the 2005 USENIX Conference on File and Storage Technologies (FAST'05); 2005. p. 21–21.
24. Bobbarjung DR, Jagannathan S, Dubnicki C. Improving duplicate elimination in storage systems. ACM Transactionson Storage (TOS). 2006; 2(4):424–48.
25. BRODER AZ. Some applications of Rabin's fingerprinting method. Sequences II: Methods in Communications, Security, and Computer Science. Springer-Verlag: New York. 1993. p. 143–52.
26. Forman G, Eshghi K, Chiocchetti S. Finding similar files in large document repositories. Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining. Chicago, Illinois, USA. 200521-24. p. 394–400.
27. Yinjin F, Hong J, Nong X, Tian L, Fang L. AA-dedupe: An applicationaware source deduplication approach for cloud backup services in the personal computing environment. IEEE International Conference on Cluster Computing; Austin, TX. 2011. p. 112–20,
28. Kruus E, Ungureanu C, Dubnicki C. Bimodal content defined chunking for backup streams. Proceedings of the 8th USENIX Conference on File and Storage Technologies; San Jose, California. 2010. p. 1–14.
29. Deutsch P, Gailly JL. ZLIB Compressed Data Format Specification version 3.3, RFC Editor, 1996. Available from: http://www.ietf.org/rfc/rfc1950.txt
30. Spiridonov A, Thaker S, Patwardhan S. Sharing and bandwidth consumption in the low bandwidth file system. Technical report. Citeseer. 2005. p. 394–400.
31. Muthitacharoen A, Chen B, Mazieres D. A low-bandwidth networks file system. ACM SIGOPS Operating Systems Review. 2001; 35(5):174–87.
32. Eshghi K, Tang HK. A framework for analyzing and improving content-based chunking algorithms; 2005. p. 1–10.
33. Bloom BH. Space/Time Trade-Offs in Hash Coding with Allowable Errors. Communications of the ACM. 1970; 13(7):422–6.
34. Hamilton J, Olsen E. Design and Implementation of a Storage Repository Using Commonality Factoring. Proceedings of the 20th IEEE/11th NASA Goddard ConferenceMass Storage Systems and Technologies (MSS' 03). 2003 Aug. p. 178–82.
35. Aronovich L, Asher R, Bachmat E, Bitner H, Hirsch M, Klein S. The Design of a Similarity Based Deduplication System. Proc. SYSTOR'09: The Israeli Experimental Systems Conference; 2009. p. 1–14.
36. Leung A, Shao M, Bisson T, Pasupathy S, Miller E. Spyglass: Fast, Scalable Metadata Search for Large-Scale Storage Systems. Proceedings of the Six USENIX Conference. File and Storage Technologies (FAST'09). 2009. p. 153–66.
37. Liu C, Gu Y, Sun L, Yan B, Wang D. R-ADMAD: High Reliability Provision for Large-Scale De-Duplication Archival Storage Systems. Proceedings of the 23rd International-Conference on Supercomputing (ICS '09); 2009. p. 370–79.
38. Bhagwat D, Pollack K, Long D, Schwarz T, Miller E, Paris J. Providing High Reliability in a Minimum Redundancy Archival Storage System. Proceedings of the 14th IEEE International Symposium. Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS '06); 2006. p. 413–21.
39. Efstathopoulos P, Guo F. Rethinking Deduplication Scalability, HotStorage '10, Second Workshop Hot Topics in Storage and File Systems. 2010 Jun. p. 1–20.
40. Kulkarni P, Douglis F, LaVoie J, Tracey JM. Redundancy Elimination within large collections of files. Proceedings of the 2004 USENIX Annual Technical Conference; Boston, MA. 2004 Jun. p. 59–72.
41. Sun Z, Shen J, Yong J. DeDu: Building a deduplication storage system over cloud computing. 2011 15th International Conference on Computer Supported Cooperative Work in Design (CSCWD). 2011. p. 348–55.
42. Douglis F, Bhardwa D, Ian H, Shilane P. Content-aware load balancing for distributed backup. Proceedings of the 25th Large Installation System Administration Conference (LISA); 2011. p.1–18.
43. Austin TC, Irfan A, Murali V, Jinyuan L, Decentralized deduplication in SAN cluster file systems. Proceedings of the 2009 Conference on USENIX Annual Technical Conference; San Diego, California. 2009. p. 101–14.
44. Yang T, Jiang H, Feng D, Niu Z, Zhou K, Wan Y. DEBAR: A scalable high-performance de-duplication storage system for backupand archiving. Proceedings of the 2010 IEEE International Parallel and Distributed Processing Symposium (IPDPS); Atlanta, GA. 2010. p. 1–12.
45. Tan Y, Jiang H, Feng D, Tian L, Yan Z, Zhou G. SAM: A Semantic-AwareMulti-Tiered Source De-duplication Framework for Cloud Backup. 39th IEEE International Conference on Parallel Processing (ICPP) ; 2010. p. 614–23.
46. Wildani A, Miller EL, Rodeh O. HANDS: A heuristically arranged non-backup in-line deduplication system. 2013 IEEE 29th International Conference on Data Engineering (ICDE). 2013. p. 446–57.
47. Zhang Y, Wu Y, Yang G. Droplet: A Distributed Solution of Data Deduplication. 2012 ACM/IEEE 13th International Conference on Grid Computing (GRID); 2012. p. 114–21.
48. Wang L, Zhang X, Zhu G, Zhu Y, Dong X. An Undirected

Graph Traversal Based Grouping Prediction Method for Data De-duplication. 2013 14th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD); 2013. p. 3–8.

49. Wang C, Qin ZG; Yang L, Wang J. A Fast Duplicate Chunk Identifying Method based on Hierarchical Indexing Structure. 2012 International Conference on Industrial Control and Electronics Engineering (ICICEE); 2012 Aug. p. 624–7.
50. Wang G, Zhao Y, Xie X, Liu L. Research on a Clustering Data De-Duplication Mechanism based on Bloom Filter. 2010 International Conference on Multimedia Technology (ICMT); 2010 Oct. p. 1–5.
51. Xing Q, Li F, Liu H. Hpod: A High-Performance Online Deduplication Cluster. 2012 Fourth International Conference on Computational and Information Sciences (ICCIS); 2012 Aug. p. 1062–5.