Similarity Detection improvisation in a Clustered Inline Deduplication for Secondary Data

Neha Amale, Prof. Jyoti Malhotra

Abstract— Data Deduplication is a storage saving technique which is a key component of enterprise storage environment. Clustered schemes are introduced in Data Deduplication domain to overcome the performance and capacity limitations of single node solutions. Often these clustered schemes face main problems of data routing and Disk chunk index lookup bottleneck. Solutions to these problems are similarity detection and locality. Locality based approaches make use of locality in backup stream to improve cluster deduplication. Similarity detection techniques make use of similarity features in data to distribute it among deduplication nodes. This approach reduces RAM usage in individual deduplication nodes. There are many techniques available for similarity detection. In this paper, a clustered inline deduplication scheme based on Simhash for similarity detection is presented. A routing algorithm based on Simhash to distribute data among deduplication node is also described. Proposed system can achieve improved similarity detection and higher deduplication throughput with low system overheads.

Index Terms— Clusters, Data Deduplication, Locality, Simhash.

1 INTRODUCTION

DATA deduplication plays a vital role in enterprise storage environment. Deduplication can save a lot of valuable storage space, which is a major cause of concern in today's rapid growth of data storage requirements.
There is a lot of research done in this domain and main focus is on

- Improving deduplication ratio
- Performing deduplication at higher speed

In this paper, we are concentrating more on clustered schemes in deduplication. Generally an inline clustered deduplication scheme consists of following components:

- Client: Client component performs chunking and chunk fingerprint generation.
- Server Cluster: Server Cluster consists of many deduplication nodes, each of which performs fingerprint lookup operation and unique data storage operation in parallel.

There are many proposed deduplication cluster schemes. Each of which is facing following main problems:

- **Data Routing:**

Due to communication overhead among deduplication nodes, in cluster deduplication system at large scale, deduplication is only performed within individual servers and cross node redundancy remains untouched. Hence Data Routing becomes very important issue in scalable cluster deduplication schemes.

- **Disk chunk index lookup problem:**

The chunk index of a large dataset, which maps chunk fingerprint to where that chunk is stored on disk in order to identify the replicated data, is generally too big to fit into The limited memory of a deduplication server and causes the parallel deduplication performance of multiple data streams from backup clients to degrade significantly due to the frequent and random disk I/Os to look up the chunk index.
There are mainly two solutions for above mentioned problems.

- **Data Similarity:**

Similarity based methods uses data similarity to distribute data among deduplication nodes and reduce RAM usage in individual nodes. These methods can easily find the node with highest similarity by extracting similarity features in the backup data streams, while they often fail to obtain high deduplication effectiveness in individual deduplication servers.

- **Locality:**

Locality based approaches, such as the stateless routing and stateful routing schemes [5], exploit locality in back- up data streams to optimize cluster deduplication. These schemes distribute data across deduplication servers at coarse granularity to achieve scalable deduplication throughput across the nodes, while suppress redundant data at fine granularity in individual servers for high duplicate elimination ratio in each node. However, to achieve high cluster deduplication effectiveness, it requires very high communication overhead to route similar data to the same node.

In this paper, proposed system will use concept of Superchunk for locality and Simhash [16] in a routing algorithm for similarity detection phase.

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 presents the design and implementation of proposed system. Section 4 conducts a series of experiments to compare the efficiency of proposed system.

The last section concludes our work.

2 RELATED WORK

All research work done in data deduplication area focuses on how to achieve better deduplication ratio, that is, remove the duplicate data as much as possible and how to make effective use of resources available. Various deduplication clustering schemes were proposed [3],[5],[6],[7],[8],[9],[10], [11], [12] to achieve scalability and high performance.

Zhu et. al. [15] have proposed use of Summary vector and Locality to reduce disk access but still this solution causes disk bottleneck issue as data size grows. To avoid chunk lookup disk bottleneck problem that most inline, chunk-based deduplication schemes face, a technique called as Sparse Indexing [13] is used.

In Ho Min Sung approach [14], authors have proposed a clustering backup system that exploits file fingerprint mechanism for multi-level Deduplication of redundant data. Their approach differs from the traditional file server system. And it works better than Sparse Indexing [13] as Sparse Indexing depends on sampling rate. The proposed approach [14] enables the efficient use of the storage capacity and network bandwidth without the transmission of the duplicate data block. This approach makes use of simple mod operation to route each chunk to respective node. Hence it suffers data fragmentation issue as well as false positive and false negative issue.

Time and space requirement of deduplication can be further reduced by using Guohua Wang's approach [10].Multiple nodes are used for performing chunk level Deduplication in parallel. This improves performance considerably. In-memory "fingerprint summary" is used that avoids duplicates among nodes in cluster. This approach also makes use of simple mod operation to route each chunk to respective node. Hence it suffers false positive, false negative and fingerprint summary refreshing issue.

Dirk Meister's proposes a new dedupv1 system [9] that further improves throughput by making use of Solid State Drives (SSDs).This system achieves a backup speed of 160MB/s with a single node without depending on locality. System proposed here has storage and capacity limitation associated with single node deduplication system.

Kaiser et. al. have proposed an Exact Deduplication cluster [6] which will be able to detect as much as redundancy as single node solution."Exact Deduplication" means a system which can detect all duplicate problems. Main aim is to achieve Exact Deduplication with small chunk size and scalability in one environment. In this system, Available storage is split into many partitions and each partition contains its own file system and it is accessed by a single node at a single point of time. But main disadvantage of this scheme is communication overhead among server deduplication nodes.

This Communication overhead issue can be resolved using ∑-Dedupe System proposed by Yinjin Fu et. al. [5].∑-Dedupe System is used to overcome intra node and internodes deduplication overheads by making use of a new routing algorithm based on similarity indexing. This routing algorithm will always route similar data to same nodes in clusters which will have maximum possibility of duplication removal. Similarity Duplicate detection is done using Jaccard Index.

Chunk-based deduplication is a commonly used deduplication technology, which divides data objects into fixed or variable length chunks. A hash fingerprint of each chunk is used to determine whether that chunk has been stored before. Chunks with the same fingerprint are assumed identical. New chunks are stored and references are updated for duplicate chunks. Chunk-based deduplication is very effective for backup workloads, which tend to be files that evolve slowly, mainly through small changes, additions, and deletions. However, unless some form of locality or similarity is exploited, chunk-based deduplication has to face the disk bottleneck problem. The total size of fingerprints required to detect duplicate chunks increases with the size of dataset, which may quickly overflow the RAM capacity and must be paged to disk. Moreover, hash fingerprint values are natively random, so the index of fingerprint cannot be cached effectively without locality, and it is not uncommon for nearly every index access to require a random disk access. This disk bottleneck severely limits deduplication throughput and increases the system overheads. To provide high deduplication throughput and low system overheads, a new deduplication scheme named Simdedup[2] is proposed for massive data storage environment, which leverages simhash algorithm to achieve the goal .In this scheme, a fat client is used to perform majority of deduplication activities, whereas server performs only storage activities.

From all above studied paper, we have learned about different scalable clustered schemes for deduplication. We are focusing on building a new deduplication clustered scheme based on Yinjin Fu et. al. [5] and further improving its similarity detection phase by making use of Simhash.

3 SYSTEM ARCHITECTURE

3.1 Objective

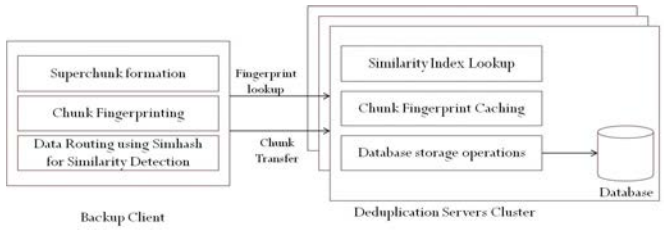
To implement a Clustered-Dedupe System using Simhash to improve similarity detection phase.

3.2 Goals and Objective

Main goals of proposed system are as follows:

- To improve deduplication ratio with low system overheads.
- To reduce inter node communication overhead.
- To increase similarity detection ratio.

3.3 System Architecture


Superchunk formation
Chunk Fingerprinting
Data Routing using Simhash for Similarity Detection
Backup Client
Fingerprint lookup
Chunk Transfer
Similarity Index Lookup
Chunk Fingerprint Caching
Database storage operations
Database
Deduplication Servers Cluster

System [1] consists of following main components:

1. Backup Clients:

Backup clients perform following tasks:

a. Data Partitioning:

This module performs data chunking with fixed or variable chunk size and super-chunk grouping. It is observed that variable chunking would be preferred as described by authors Cai Bo et. al.[4]

b. Chunk fingerprinting:

This module calculates chunk fingerprints by a collision-resistant MD5 hash function.

c. Data Routing:

This module selects a deduplication node for the routing of each super-chunk by the data routing scheme.

2. Deduplication Server Cluster:

It performs following main tasks:

a. Similarity index lookup:

This module returns the results of similarity index lookup for data routing.

b. Chunk index cache management:

This module buffers the recent hot chunk fingerprints in chunk index cache to speed up the process of identifying duplicate chunks.

c. Parallel container management:

This module stores the unique chunks in larger units, called containers, in parallel. It consists of mainly database operations.

3.4 Proposed Similarity detection algorithm

- Input: a chunk fingerprint list of a super chunk S,{fp1,fp2,…,fpn}
- Output: a target node ID,i

1. Calculate Simhash Index SIndex for Super Chunk S and sent SIndex to candidate node with ID {SIndex mod N} in deduplication server cluster of N nodes.
2. In deduplication server cluster, obtain the most similar superchunk by comparing the SIndex of the previously stored super-chunks in the similarity index. The returned value would be most similar previously stored superchunk on that node.
3. Choose the deduplication server node with ID i that satisfies SIndex mod N as the target node. On this node, get all fingerprints of chunks that belong to most similar Superchunk for deduplication.

4 Evaluation and Results

4.1 Objectives

We have simulated this system using Java Threads. And Following results are taken on incremental backup data. And results are compared with ∑-Dedupe system simulated in Java environment.

4.2 Result and Analysis

1. Deduplication ratio

Deduplication ratio = Bytes in / Bytes out

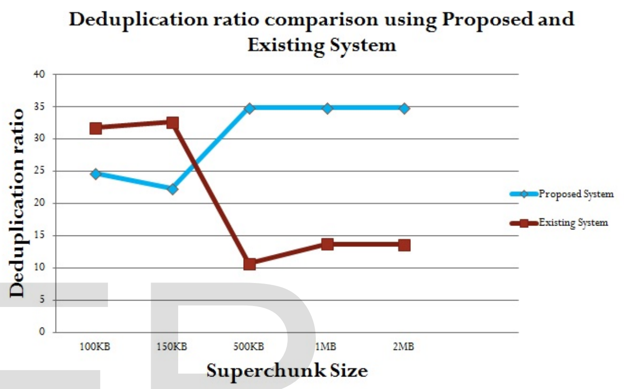


Fig: 2 Deduplication ratio

The fig 2 shows the comparisons of the Deduplication ratio of existing system using Jaccard Index for Similarity detection and proposed deduplication system using Simhash for Similarity detection. It can be seen from graph that as Superchunk size increases, System using Simhash performs better (Deduplication ratio is higher) than System using Jaccard Index.

2. Similarity Detection Accuracy

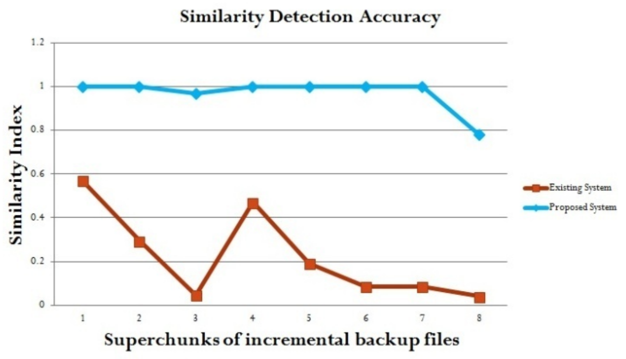


Fig: 3 Similarity Detection accuracy

The fig 3 shows the comparisons of similarity count obtained for different superchunks in existing system and proposed system. In Proposed system simhash is used for similarity detection and as simhash works on bit level and computes hash for entire Superchunk, it is much accurate in finding Similarity among Superchunks. Whereas existing system uses Jaccard Index and it tries to find ratio of number of common element in Superchunks and Total number of elements in Superchunks.

3. Number of Database Comparisons

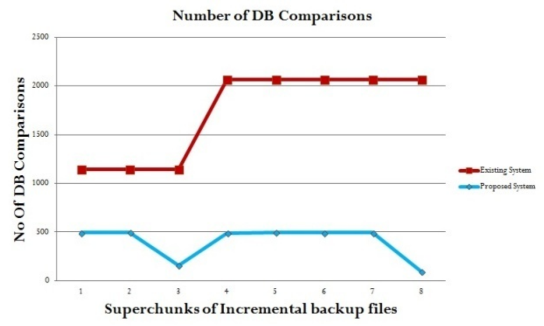


Fig 4: Number of Database Comparisons

The fig 4 shows the database comparisons obtained for different superchunk using existing and proposed system. As proposed system, performs deduplication with most similar Superchunks it required less comparisons whereas existing system uses most similar cluster node for deduplication , it requires more comparisons.

5 Conclusion

Data deduplication is a storage saving technique which is a key component of enterprise storage environment. All Deduplication solutions proposed so far are either focusing on how to reduce more duplicates or how to improve performance of the system.

Proposed clustered inline deduplication for secondary data performs well in terms of following factors -

- Deduplication ratio is improved by average 79%.
- Scalability- In terms of inter-node communication overhead.
- Similarity detection - Improved by average 80%.
- Deduplication throughput – In terms of number of comparisons required for deduplication - 70% less comparisons are required.

From Result and Analysis done during this phase, it can be said that Simhash works better than Jaccard Index for Similarity detection phase and hence is used in the proposed system.

References

[1] Neha Chetan Amale, Jyoti Malhotra, “A Clustered Inline Deduplication for Secondary Data”, Cyber Times International Journal of Technology & Management (CTIJTM), Vol. 7, Issue 1, April-2014.

[2] Wenbin Yao, Pengdi Ye, “Simdedup: A New Deduplication Scheme Based on Simhash”, WAIM 2013 Workshops, LNCS 7901, pp. 79–88, 2013

[3] Chao Chen , Jonathan Bastnagel , Yong Chen ,“Data Deduplication in a Hybrid Architecture for Improving Write Performance”, ROSS '13 Proceedings of the 3rd International Workshop on Runtime and Operating Systems for Supercomputers, 2013

[4] Cai Bo, Zhang Feng Li, Wang Can ,“Research on Chunking Algorithms of Data De-duplication”, Proceedings of the 2012 International Conference on Communication, Electronics and Automation Engineering,pp.1019-1025,2012

[5] Yinjin Fu, Hong Jiang, and Nong Xiao,” A Scalable Inline Cluster Deduplication Framework for Big Data Protection”, IFIP International Federation for Information Processing 2012, pp. 354–373, 2012

[6] Kaiser, H., Meister, D., Brinkmann, A., Effert, S. “Design of an Exact Data Deduplication Cluster.”, Proc. of IEEE MSST, pp.1-12, 2012

[7] Dong, W., Douglis, F., Li, K., Patterson, H., Reddy, S., Shilane, P.,”Tradeoffs in Scalable Data Routing for Deduplication Clusters.”, Proc. of USENIX FAST, 2011

[8] Jaehong Min, Daeyoung Yoon, and Youjip Won,” Efficient Deduplication Techniques for Modern Backup Operation”, IEEE Transactions On Computers, Vol. 60, No. 6, pp.824-840,2011

[9] D. Meister and A. Brinkmann, “dedupv1: Improving deduplication throughput using solid state drives,” in Proceedings of the 26th IEEE Conference on Mass Storage Systems and Technologies (MSST), 2010.

[10] Guohua Wang and Yuelong Zhao, Xiaoling Xie, and Lin Liu.,” Research on a clustering data de-duplication mechanism based on Bloom Filter”, Multimedia Technology (ICMT), 2010 International Conference, 2010.

[11] Dubnicki, C., Gryz, L., Heldt, L., Kaczmarczyk, M., Kilian, W., Strzelczak, P., Szczepkowski, J., Ungureanu, C., Welnicki, M.,”HYDRAstor: a Scalable Secondary Storage.”, Proc. of USENIX FAST, 2009.

[12] D. Bhagwat, K. Eshghi, D. Long, and M. Lillibridge, “Extreme binning: Scalable, parallel deduplication for chunk-based file backup,” in Proceedings of the 17th IEEE International Symposium on Modeling, Analysis, and Simulation (MASCOTS), 2009.

[13] M. Lillibridge, K. Eshghi, D. Bhagwat, V. Deolalikar, G. Trezise, andP. Camble, “Sparse indexing: Large scale, inline deduplication usingsampling and locality,” in Proceedings of the 7th USENIX Conferenceon File and Storage Technologies (FAST), 2009.

[14] Ho Min Sung, Wan yeon Lee, Jin Kim, and Young WoongKo, “Design and implementation of clustering file backup server using file fingerprint,” Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, September 2008, pp. 61-73.

[15] B. Zhu, K. Li, and H. Patterson, “Avoiding the disk bottleneck in the Data Domain deduplication file system,” in Proceedings of the 6thUSENIX Conference on File and Storage Technologies (FAST), 2008.

[16] Caitlin Sadowski, Greg Levin,” SimHash: Hash-based Similarity Detection”, 2007.