Using Efficient Deduplication Method in Large-scale Distributed Storage System

Myat Pwint Phyu
University of Computer Studies, Yangon
myatpwint.ucsy@gmail.com

Ni Lar Thein
University of Computer Studies, Yangon
nilarthein@gmail.com

Abstract

Today's storage systems have a major issue for the long-term storage of massive amounts of unstructured data. Availability and reliability are the basic properties of the most storage system. Replication which is the simplest redundancy scheme can help the storage system to achieve continuous access. But too much redundancy will not improve the data availability when the amount of replication reaches a certain point. In this paper, an efficient data deduplication method in large-scale distributed storage system is presented. Since good data indexing is very helpful for duplicate detection, the deduplication scheme with Bloom filter array is used for the sake of space and look-up efficiency in distributed storage system.

Keywords

Unstructured data, large-scale distributed storage, Bloom filter, deduplication

1. Introduction

As storage system grows larger and more complex, many file systems have emerged focusing on specialized requirements such as data sharing, remote file access, distributed file access, parallel files access, high performance computing, archiving etc. Moreover, storing and managing the growth of unstructured data is one of the great challenges.

The boost of the massive amount of data in storage system results in large bandwidth and high latencies demands to provide fault tolerant continuous access in distributed storage system. Many research communities have attempted to solve these availability and reliability issues introducing replication techniques. On the other hand, space efficiency is also one of the primary concerns in the storage system. Numerous systems have introduced data deduplication as a technique for data reduction [2, 3].

Data deduplication is a simple technique which includes dividing a file into multiple chunks and computing a hash value for each chunk, determining whether each chunk is duplicated or not, maintaining the duplicated chunks as a reference and storing the unique chunks in the data store. So, the deduplication eliminates any redundant information from given data sets.

There are some schemes achieving deduplication based on delta encoding, in which duplicate regions of files are eliminated and similar regions of files are also compressed with delta compression [7]. Only one index of fingerprints is required rather than full indexes or partial index to find similarity. Efficient stream-informed caching technique is used to remove extra disk indexes.

In this paper, Bloom filter array (BFA) is used as an incoming query filter for distributed storage system. The array structure of the Bloom filter is used to decide whether a specific attribute value belongs to a given set while serving the look-up requests in $O(1)$ time complexity from memory. The efficiency of the deduplication using BFA is evaluated in this paper.

The rest of this paper is organized as follows. Section 2 described how the deduplication mechanisms are operated in the recent storage systems as related work. Section 3 highlights some observations over challenges. The proposed system is introduced in Section 4.

Performance evaluation is in Section 5 and 6 and then we conclude the paper with future work in Section 7.

2. RELATED WORK

LBFS [4] optimizes the communication using Rabin fingerprint algorithm to divide each file into variable sized chunks, and every chunk with a global unique identifier. Through comparing the identifiers, duplicate chunks can be eliminated.

De-duplication storage creates heavily chunk sharing among different files and as a side effect, it can make file chunks spread among multiple storage nodes of the chunk repository thus gradually reducing read performance. To solve this problem, DEBAR [8] employs a defragmentation mechanism that automatically aggregates file chunks to one or few storage nodes, thus significantly reducing storage fragmentation and retaining high read throughput. So, it is a scalable and high-performance de-duplication storage system for backup and archiving, to overcome the throughput and scalability limitations of the data de-duplication schemes. It uses a two-phase de-duplication scheme (TPDS) that exploits memory cache and disk index properties to judiciously turn the notoriously random and small disk I/Os of fingerprint lookups and updates into large sequential disk I/Os, hence achieving a very high de-duplication throughput.

Zhu et al. also provide a comprehensive look at data deduplication on backup workloads in addition to introducing the use of Bloom filters to improve the lookup speed for testing whether a write is a duplicate [9]. While this method is much faster than a linear search over the database, it is still not fast enough to avoid affecting performance.

Replication of deduplicated, compressed data offers the most economical approach to the automated movement of data copies to a safe site using minimum WAN bandwidth. This ensures fast recovery in case of loss of the primary data, the primary site, or the secondary store. EMC Data Domain Replicator software provides simple, fast, robust WAN based disaster recovery for the enterprise. This software offers comprehensive flexibility for a variety of topologies in the distributed enterprise, from remote offices at the edge to large core data centers [10]. Unlike other deduplication methods, Data Domain deduplication is inline, so replication completes as fast as possible to minimize risk and maximize currency of the restore point. EMC Data Domain leverages dynamically variable length deduplication coupled with local compression and therefore can eliminate up to 99 percent of the bandwidth used by normal replication methods.

HYDRAstor [1] is a multi-node, content-addressable storage system that stores blocks at configurable redundancy levels and supports high-throughput reads and writes for streams of large blocks. High storage efficiency is facilitated by variable block size global deduplication.

In this paper, the efficient data deduplication scheme with an efficient indexing mechanism is presented for large-scale distributed storage system.

3. Observations over Challenges

This section will describe some observations over challenges for the proposed system.

3.1 Block level Data Deduplication

Block deduplication requires more processing power than the file deduplication, since the number of identifiers that need to be processed increases greatly. Correspondingly, its index for tracking the individual iterations gets also much larger. Using of variable length blocks is even more source-intensive. Moreover, sometimes the same hash number may be generated for two different data fragments, which is called hash collisions. If that happens, the system will not save the new data as it sees that the hash number already exists in the index. Block-level deduplication focuses more on the sub-file level changes and typically results in lower levels of storage consumption when sub-file duplicates exist in the data.

3.2 File level Data Deduplication

The file-level deduplication can be performed most easily. It requires less processing power since the hash numbers of files are relatively easy to generate. However, there is the reverse side of a coin: if only one-byte of a file is changed, its hash number also changes. As a result both file versions will be saved to storage and the data reduction will not be obvious.

3.3 Byte level Data Deduplication

Byte-level deduplication doesn't need additional processing – in this case data chunks are compared in the most primitive way – byte by byte. It performs checks for redundant fragments even more accurately. Byte-level deduplication takes quite much time and as a rule is applied to in post-process deduplication.

3.4 Variable and Fixed Sized Segment

In the conventional file system, the data streams are created as fixed-sized blocks for simplicity. But in the deduplication process of fixed-sized segments, any change in a data block creates changes in all the subsequent blocks. This can be avoided by using variable-sized segment. In figure 1 applying fixed block length to a data sequence. After making a single change to Block **A** (an insertion), all of the blocks have changed content and no duplication is detected. Therefore, 8 unique blocks must be stored.

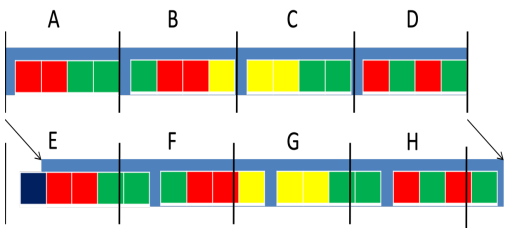

Figure. 1. Applying fixed block length to a data sequence

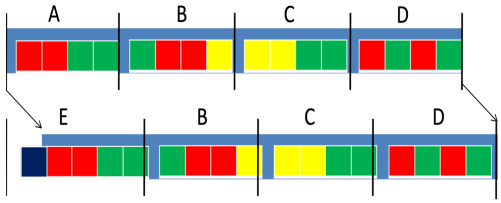

Figure. 2. Applying variable block length to a data sequence

Figure 2 shows data deduplication which utilizes variable-length blocks or data. In this case, Block **A** changes when the new data is added but none of the other blocks is affected. Blocks **B**, **C**, and **D** are all recognized as identical to the same blocks in the first line. So, only 5 unique blocks are needed to store.

3.5 Data Deduplication

In the most system, deduplication is applied as an effective technique to optimize the storage space utilization. As shown in figure 3, the most common de-duplication method has been to divide a file or stream into chunks and eliminate the duplicate copies of chunks. Duplicate chunks are identified by comparing the chunk fingerprints represented by the hash values of chunk contents.

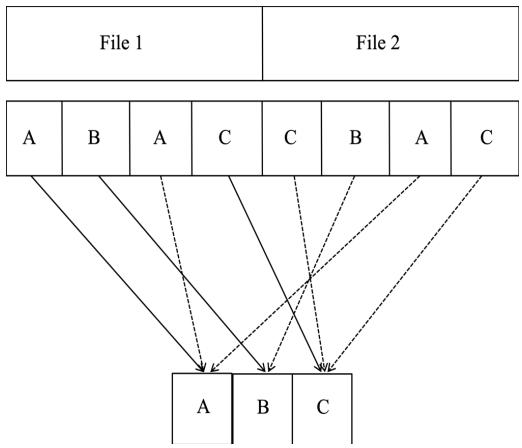

Figure. 3. Data de-duplication process

4. THE PROPOSED SYSTEM

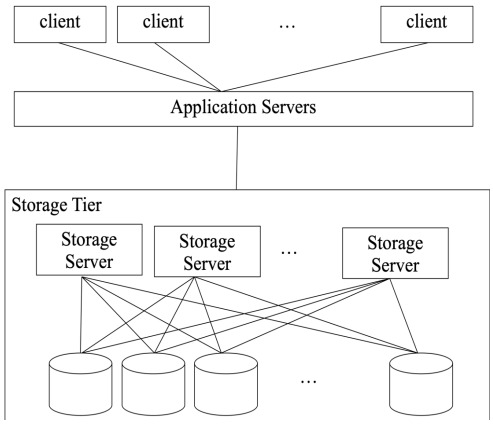

Figure. 4. Architecture of the large-scale shared storage system

In our ongoing research, we will describe the efficient replication method used in large-scale distributed storage system. A large number of non-expensive commodity storage servers are built in the test bed since we have focused the large-scale distributed storage architecture shown in figure 4.

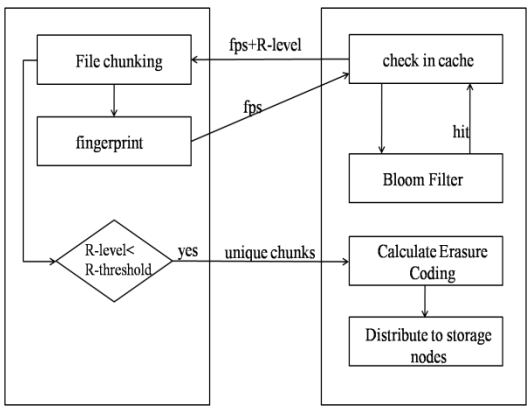

Figure. 5. Process flow of the availability and reliability aware data replication in large-scale distributed storage system

A system to replicate data in large-scale distributed storage system is being developed for space efficiency and reliability guarantee. A client will operate by sending the fingerprints of a file to a randomly selected storage node as an agent server. In the server, the fingerprints are checked to see if it is in the cache. If so, there will be existing segments. If it is not, a Bloom filter is checked to determine whether the fingerprint is likely to exist in the storage volume. If it is a hit, the corresponding list of fingerprints is loaded into the cache. The response of which segments have already existed and their associated reliability levels are returned to the client and then the system is going to do the rest of the flow as shown in figure 5. The detail system explanation is omitted in this paper and the performance of data deduplication with the Bloom filter array lookup is only emphasized.

4.1 Deduplication with Bloom Filter Array

In the proposed system, Bloom filter array (BFA) [5] is used to support efficient lookup in deduplication to support data availability in distributed storage system. When a request comes to a node (server), Bloom filter array (BFA) starts to return the hit/ miss response. Each node has a bloom filter maintaining fingerprints of locally stored chunks and the bloom filters of other nodes as an array.

Bloom filter can be used as a filter to reduce the number of unnecessary lookups. This is done in a space and time efficient manner so that the membership queries can be performed in-memory, while the retrieval may require an I/O operation.

It is a bit array of M bits for representing a set $S = \{a_1, a_2, \dots, a_n\}$ of n items. All bits in the array are initially set to 0. Then, a Bloom filter uses q independent hash functions $\{h_1, \dots, h_q\}$ to map the set to the bit address space$[1, \dots, M]$. For each item a, the bits of $h_i(a)$ are set to 1. To check whether an item a is a member of S, we need to check whether all $h_i(a)$ are set to 1. If not, a is not in the set S . If so, a is regarded as a member of S with a false positive probability. The false positive performance depends on the bit per elements ratio m/n and the optimal

number of hash functions is determined by $k = (m/n)ln2$.

5. Experimental Details

In this paper, the data deduplication experiments are performed using trace-driven simulation which can give absolute results getting the actual data and feeding it to a simulation. The various types of data such as text, image, and video with some repetitive information are used during the experiment. The variable length chunks are calculated by using Rabin fingerprints [6]. Table 1 describes the statistics for the test dataset.

Table 1. Statistics of the experimental data

Number of files	1000
Minimum file size	4KB
Maximum file size	45MB
Number of unique chunks	700000
Minimum chunk size	1KB
Maximum chunk size	32KB

6. Performance Evaluation

The proposed system focuses on the data reduction in distributed storage system. Therefore, we emphasis on compression ratios and throughput results during the experiment.

6.1 Data Compression Ratio

Since the proposed system is for data reduction, data compression is essential to calculate how many data are reduced and it can be formulated as $\frac{original\ size}{after\ compression\ size}$. Figure 6 shows the comparison of compression factors for data deduplication, delta compression and LZ compression. It can be seen that data deduplication scheme has the most compression factor than the other techniques.

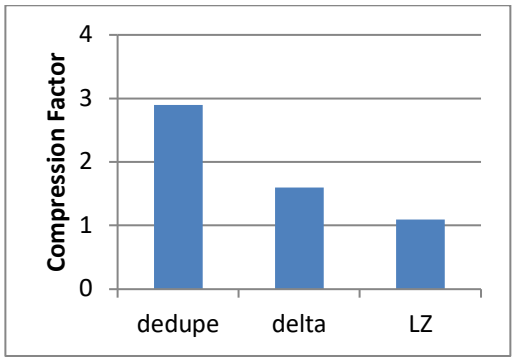

Figure. 6. Comparison of different compression techniques

6.2 Throughput Evaluation

To determine the throughput of the deduplication system for the distributed storage system, various types of data shown in table 1 are used. The communication between client and storage servers is processed by Remote Method Invocation (RMI) simulation. Figure 7 shows the throughput of the system with different requested files from the client. According to the figure, the throughput is still large even with the maximum file size of our test data.

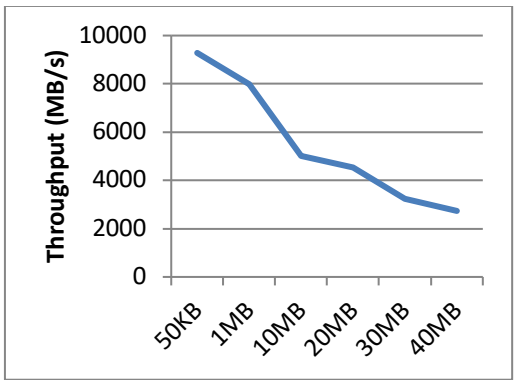

Figure. 7. Throughput measurement depending on requested file size

7. Conclusion and Future Work

In this paper, we present the efficient data deduplication method is presented to optimize the large-scale distributed storage system. It uses Bloom filter array structure to provide efficient indexing scheme in space saved deduplication method. We evaluate our system with various kinds of workloads and showed that the system can provide large storage saving for large-scale distributed storage system.

Actually data deduplication opposes data reliability with the different goal. The data reliability protects its associate information against failures by generating more redundant information, while the deduplication eliminates any redundant information from given data sets. Only an efficient data deduplication scheme is discussed in this paper and the complete availability and reliability aware data replication in large-scale distributed storage system is still in progress.

8. REFERENCES

[1] 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 Proceedings of the 7th USENIX File and Storate Technologies, 2009, pp. 197–210.

[2] K. Eshghi, M. Lillibridge, L. Wilcock, G. Belrose, and R. Hawkes, "Jumbo store: Providing efficient incremental upload and versioning for a utility rendering service", in Proceedings of the 5th USENIX Conference on File and Storage Technologies, 2007.

[3] C. Liu, Y. Gu, L. Sun, B. Yan, and D. Wang, "R-ADMAD: High reliability provision for large-scale deduplication archival storage systems", in Proceedings of International Conference on Supercomputing, 2009, pp. 370–379.

[4] A. Muthitacharoen, B. Chen, and D. Mazi`eres, "A low-bandwidth network file system", in SOSP '01: Proceedings of the 18th ACM Symposium on Operating Systems Principles, pages 174–187, 2001.

[5] M. P. Phyu and N. L. Thein, "Using Bloom Filter Array (BFA) to Speed up the Lookup in Distributed Storage System", International Journal of Computer Applications (IJCA) 60(11):26-28, December 2012.

[6] M. O. Rabin, "Fingerprinting by random polynomials", Technical Report TR-15-81, Center for Research in Computing Technology, Harvard University, 1981.

[7] P. Shilane, M. Huang, G. Wallace, W. Hsu, "WAN Optimized Replication of Backup Datasets Using Stream-Informed Delta Compression", in Proceedings of the USENIX Conference on File and Storage Technologies (FAST'12), 2012.

[8] T. Yang, J. Hong, F. Dan, N. Zhongying, Z. Ke, and W. Yaping, "Debar: A scalable high-performance de-duplication storage system for backup and archiving", in Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS '10).

[9] B. Zhu, K. Li, and H. Patterson, "Avoiding the disk bottleneck in the data domain deduplication file system", in Proceedings of the 6th USENIX Conference on File and Storage Technologies, page18.USENIX Association, 2008.

[10] "EMC Data Domain Replicator: A detailed review", White paper, EMC^2, May 2012.