Capacity Optimized Deduplication for Big Unstructured Data in Scale-out Distributed Storage System

Myat Pwint Phyu[1,a], Thandar Thein[2,b]

[1]University of Computer Studies, Yangon, Myanmar

[2]University of Computer Studies, Yangon, Myanmar

[a]myatpwint.ucsy@gmail.com, [b]thandartheinn@gmail.com

Keywords: Scale-out Distributed Storage, Bloom Filter, Data Deduplication, Capacity Optimization, Unstructured Data

Abstract. Organizations in every market segment require their storage utilization to optimize and cost-effectively align with the changing storage capacity needs of their business. Scale-out distributed storage is becoming an important environment for the storage and exchange of information as the data intensive workloads are rapidly growing in the last decades. Data deduplication can provide efficient storage-space solution for the storage industries. However, during the data deduplication process, it faces the barrier of network and the burden of disk access. In this paper, an efficient deduplication scheme for scale-out distributed storage system is proposed to address the above issues. Since duplicate data detection is important in deduplication, Bloom filter array (BFA) is applied to reduce the I/O intensive workloads and accelerate the detection process without the hops between the nodes. The performance of this system is evaluated in terms of deduplication efficiency and throughput. The proposed deduplication scheme not only saves storage space but also shortens time for further processes.

Introduction

Recent years, distributed storage systems are becoming more and more popular with the rapidly increasing demand for large-scale data storage. Such systems have a major issue for the long-term storage of massive amounts of unstructured data. Optimizing the use of storage is part of a broader strategy to provide an efficient information infrastructure responsible for dynamic business requirements and the scale-out distributed storage is coming of age as a good solution. Therefore, data reduction techniques are widely used in computer networks and data storage systems to increase the efficiency of data transfers and reduce space requirements on the receiving device. Most techniques focus on the problem of compressing individual files or data streams of a certain type (text, image, audio). It differs from data deduplication. Data compression uses algorithms to reduce the file size and deduplication eliminates the copied files or the duplicate data blocks. The elimination of duplicate data, commonly known as deduplication, is now widely accepted for reducing the amount of storage space.

Therefore, one way to reduce the costs of enterprises is deduplication, in which repeated data is referenced to a unique copy; this approach is effective in cases where data is highly redundant. For example, typical backups contain copies of the same files captured at different times, resulting in deduplication ratios as high as 95% [2]. Deduplication can be useful even in primary storage, because users often share similar data such as common project files or recordings of popular songs.

Numerous systems have introduced data deduplication as a technique for data reduction [1, 5, 10] as the space efficiency is one of the primary concerns in the storage system. 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 [14, 15]. In [15], stream-informed delta compression is added to already existing deduplication systems and eliminates the need for new, persistent indexes. Only one index of fingerprints is required rather than full indexes or partial index to find similarity. But it considered only for space reduction to increase communication speed in backup system.

In this paper, an efficient deduplication framework for scale-out distributed storage environment is proposed to optimize the capacity. The proposed scheme has the properties to avoid the disk lookup overhead and remove the unnecessary communication between storage nodes. 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 performance efficiency of the deduplication is evaluated.

The rest of this paper is organized as follows. Related work describes how the deduplication mechanisms are operated in the recent storage systems. Then some background theories concerning with the proposed system are highlighted. The proposed deduplication framework is presented in the next section. Performance evaluation is performed in the following section and then we conclude the paper finally.

Related Work

In this section, the previous literatures are reviewed concerning with the various mechanisms which operate data deduplication. The requirements of today's enterprises highlight that a single-node deduplication system cannot fulfill the throughput and scalability. Besides there are willing to miss opportunities to deduplicate data chunks, which a single node deduplication systems would detect.

As single-node deduplication, B. Zhu et al. [3] presented data domain deduplication file system (DDFS). It relieves the disk bottleneck introducing the use of Bloom filters to improve the lookup speed for testing whether a write is a duplicate. While this method is much faster than a linear search over the database, it is still not fast enough to avoid affecting performance. Sparse Indexing [12] also exploits the inherent backup-stream locality to solve the index-lookup bottleneck problem. Different from the above DDFS, it is an approximate deduplication solution that samples index for fingerprint-lookup and only requires about half of the RAM usage of DDFS. But its duplicate elimination and throughput are heavily dependent on the sampling rate and chunks locality of backup streams. Both DDFS and Sparse Indexing are designed for backup workloads, and they do not address the scalability issue in distributed environment.

D. Bhagwat et al. [6] proposed a file-similarity based cluster deduplication scheme called Extreme Binning. It chooses a representative chunk ID per file. Redundancies between files are only found when the chosen chunk ID is the same for both files. The approach is based on Broder's theorem that essentially states that if two files share the same chunk ID, both files are likely very similar. W. Xia et al. [17] presented a similarity-locality based deduplication system named SiLo that uses both similarity and locality in backup streams to achieve higher throughput and near-complete duplicate elimination at a much lower RAM overhead. HYDRAstor [4] is also a clustered deduplication storage which handles data duplications at a large chunk (64KB) granularity without data sharing among the nodes. Nevertheless, this amount is still too limited to capture and preserve sufficient amount of locality for cluster deduplication purposes.

Similarly, Y Fu el al. proposed $\sum$-Dedupe [18], a scalable inline cluster deduplication framework, as a middleware deployable in cloud data centers. It uses data similarity and locality to optimize cluster deduplication in inter-node and intra-node scenarios, respectively. Besides, DEBAR [16] is a scalable and high performance deduplication 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 deduplication throughput.

From the observations over previous research, deduplication can be viewed as a technique for either the storage capacity or the network resource saving. There are many commercial clustered storage systems with deduplication such as EMC Data Domain's global deduplication array [7] and IBM's ProtecTier [9]. Moreover, currently cloud computing is applied more in data intensive areas such as e-commerce or scientific computing. Therefore, engineering oriented deduplication cloud storage systems like DeDu [19] and Dedoop [11] also come out.

In this paper, the proposed framework works as the inline data deduplication in scale-out distributed storage system. The fixed-size chunks of a file are used to find the data similarity. To reduce the network bottleneck and the disk lookup overhead, BFA is applied in the duplicate data detection scheme. The proposed framework intends to support storage capacity optimization by reducing the consumption of space required to store the data. The performance of the framework is measured with data deduplication efficiency and throughput.

Preliminary Concepts

In popular storage environment such as a cloud environment, usually there are two approaches to design the storage solutions: scale-up storage and scale-out storage.

Storage Networking Designs. In popular storage environment such as a cloud environment, usually there are two approaches to design the storage solutions: scale-up storage and scale-out storage. The traditional scale-up solutions increase the performance, capacity or throughput by adding resources within a tightly coupled system that shares a common pool of resources that work in tandem. The illustration of scale-up storage can be seen in Figure 1.

Scale-out storage uses a number of storage nodes consisting of multiple low-cost computer servers and storage components. It support balanced data growth on an as-needed basis. As shown in Figure 2, scale-out storage solution increases the performance, capacity or throughput by adding resources as a loosely coupled system composed of nodes that work side-by-side, in parallel.

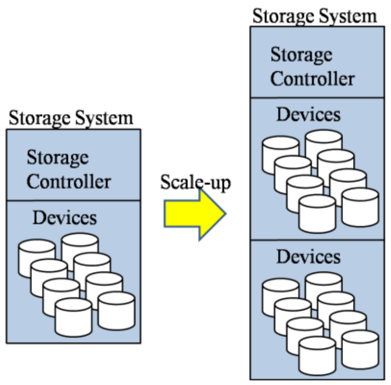


Fig. 1 Scale-up Storage Solution

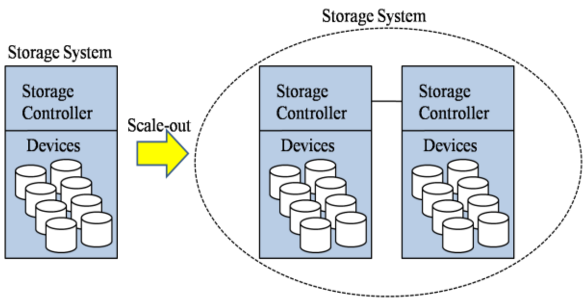


Fig. 2 Scale-out Storage Solution

Data Deduplication. In the most system, deduplication is applied as an effective technique to optimize the storage space utilization. It is a simple technique which includes dividing a file into multiple chunks, computing a hash value for each chunk, determining whether each chunk is duplicate or not, maintaining the duplicate chunks as a reference, or storing the unique chunks in the data store. Therefore, the deduplication eliminates any redundant information from given data sets.

As the limitation of deduplication, the nature of the data deduplication which has the advantage for capacity saving in storage system also has a side effect which is the potential to reduce storage reliability. It is because of the properties such as sharing and dependency of deduplication mechanism.

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 3 applying fixed block length to a data sequence is demostrated. 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 4 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.

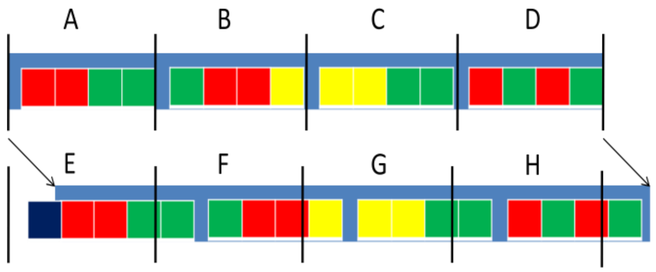


Fig. 3 Applying fixed block length to a data sequence

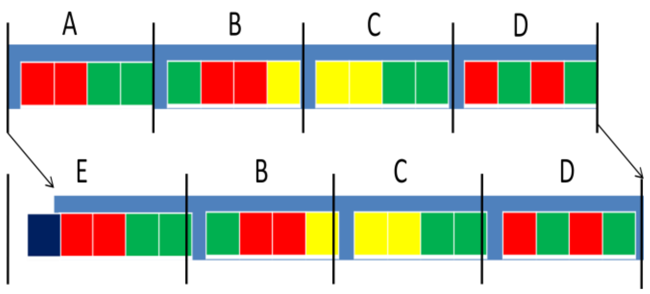


Fig. 4 Applying variable block length to a data sequence

Proposed Efficient Data Deduplication Framework

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. Deduplication plays an essential role in the storage infrastructures to utilize the space compactly. It provides a mechanism to identify duplicate data in a storage system and store only one (or few) copies of the duplicate data, thus, saving storage space. It can eliminate any redundant information from given data sets.

Scale-out Distributed Storage Architecture. Usually, scaling up an existing system often results in simpler storage management than with the scale-out approach, as the complexity of the underlying environment is reduced, or at least known. However, with a scale-up infrastructure, the performance may suffer due to increasing density of shared resources in this topology.

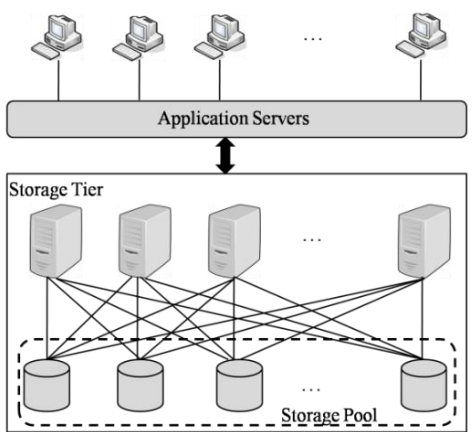


Fig. 5 Architecture of the scale-out distributed storage system

With scale-out topology, the performance may increase due to the increased number of nodes where more CPU, memory, spindle and network interfaces are added with each node. Therefore, a deduplication mechanism used in scale-out distributed storage system is described in this paper. A large number of non-expensive commodity storage servers are built in the test bed since we have focused the scale-out distributed storage architecture shown in Figure 5.

Proposed System Components. The proposed system consists of three main components such as clients, an agent server and the storage pool as depicted in Figure 6. They work as the following:

Clients act as application servers in the real environment like email servers and web servers. And it implements file chunking, fingerprint making and chunk filtering.

Agent server is a representative server from the storage tier in which BFA operates the duplicate detection.

Storage pool is a collection of storage servers on the environment of consistent hashing. The storage pool grows on as-needed basic. Blocks of data chunks can be placed into the appropriate place in the pool depending on the key-value pairs.

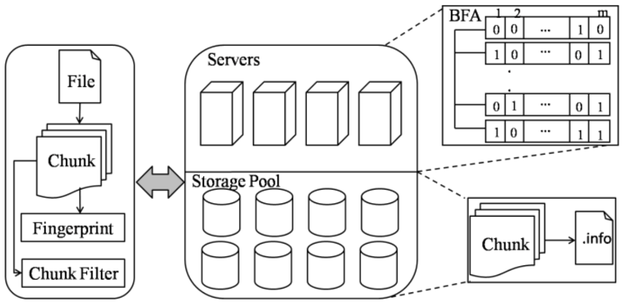


Fig. 6 Overview of the proposed data deduplication

Efficient Data Deduplication Scheme. Chunking based data deduplication is becoming a prevailing technology to reduce the space requirement for both primary file systems and data backups. There are many recent and popular approaches for data chunking. In the proposed deduplication scheme, the input file is divided into multiple chunks, each size is 4KB, and then a hash value is computed for each chunk as fingerprint. There are many good hash functions in practice which have the properties and designed goals such as: minimizing the number of collisions, distributing signatures uniformly, having a large amount of effect ensuring output varies from small input change, and detecting permutations on data order. Among them, in the proposed scheme, SHA-1 is used to produce fingerprints because it is the most widely used of the existing hash functions, and is employed in several widely used application and protocols.

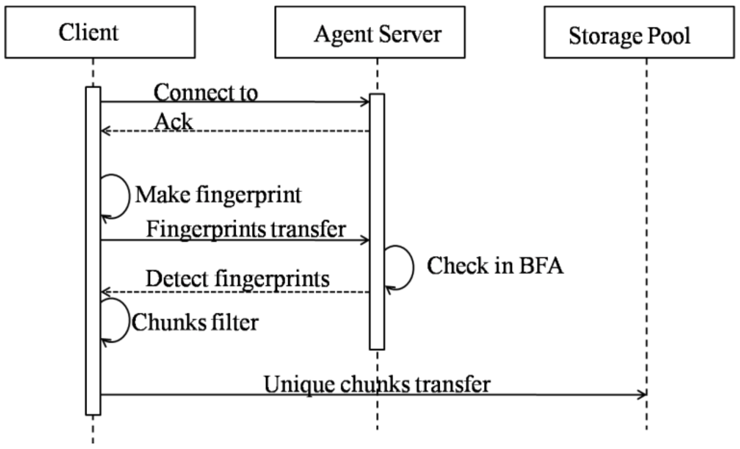


Fig. 7 Procedures for data deduplication

As can be seen in Figure 7, the data deduplication is processed as an in-line operation. The client operates 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 they have already existed. A Bloom filter array (BFA) is used to determine whether the fingerprint is likely to exist in the storage volume. If it is a hit, the corresponding list of fingerprints is returned to the client. Then the client filters the duplicate data and sends only the unique data to the storage pool.

Implementation of the Proposed Data Deduplication Scheme. Deduplicating storage systems take the advantage of redundancy to reduce the underlying space needed to contain the file systems. It can work at either the sub-file or whole-file level. In the proposed deduplication system, Bloom filter array (BFA) [13] is used to support efficient lookup in deduplication for 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.

```
Algorithm: Duplicate_Data_Detection
Lookup (x)                        // Procedure for looking up data
Input    : $x$                    // data to search in Bloom filter
Output : $true/ false$            // return hit/miss
1:  Calculate_Hash ($x$);         // do as much as the array size
2:  for $B: 1$ to $n$ do
3:      $c \leftarrow 0$;         // counter for hash functions
4:      $j \leftarrow 1$;         // flag for hash functions
5:      while $j \leq k$ do
6:          $i \leftarrow A[j]$;  // read each hash value
  //To check the specified bit position in Bloom filter is on/off
7:          if $B_i == 1$ then
8:             $c$++;             // if the bit is on, increment counter
9:          end if
10:         $j \leftarrow j + 1$; // increment the flag
11:     end while
  //To check if the 'on' bit position in Bloom filter is enough
12:     if $(c==j)$ then
13:        return $true$;
14:     end if
15: end for
Calculate_Hash (x)   //Procedure for calculating  k independent hash functions
1:  for $j: 1$ to $k$ do
2:      $A[j] \leftarrow h_j(x)$;   //calculate hash value of x with k times
3:  end
```

Fig. 8 Look-up algorithm in BFA

Figure 8 gives the algorithm for deduplication look-up in BFA. A query to BFA is encoded as a fingerprint with k independent hashes. In the BFA, there are n Bloom filters and the bit positions of the queried fingerprint are compared with those of each Bloom filter. If all matches are non-zeros, it can be said that the query is presented in the Bloom filter array.

The efficiency of a Bloom filter depends on some key parameters. The more the number of hash functions (k) and the size of filter (m) are used, the more the computation and the space and the lower the false positive rate will be got. Therefore, if the Bloom filter array is arranged by the access frequency, the operations on the BFA can be done in $O(n)$ as a worse case.

Performance Evaluation

The implementation of the system is simulated based on RMI protocol using a configuration which have Intel(R) Core(TM) i3-2600 CPU @ 3.40GHz, 4 GB RAM, 1TB hard disk and Gigabit Ethernet. The data traces are collected from the web server and mail server of the CS department in Florida International University (FIU) [8]. For both datasets, an average chunk size is chosen by 4KB with a minimal file size of about 4MB and 7MB and a maximum file size of about 50MB and 158MB for Mail and Web datasets respectively. Therefore, the chunking process generates 8,991,539 and 275,251 unique chunks for the whole data trace.

Accuracy of BFA. Since the data loss is unaffordable in deduplication, the accuracy of BFA has the great effect on the whole process. Figure 9 shows that the accuracy of BFA is proportionally high depending on the bit per element ratio $(\boldsymbol{m/n})$ and the space consumption is linear to the number of items to get a specified false positive probability.

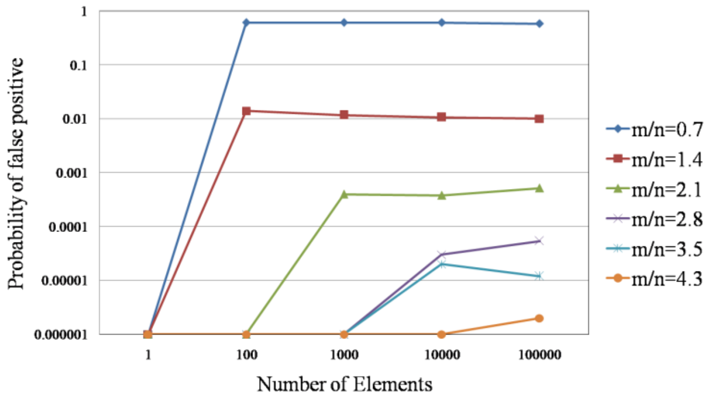


Fig. 9 False positive probability of BFA using 10 hashes

Data Deduplication Ratio. Since the proposed system is for data reduction, data deduplication is essential to calculate how many data are reduced and it can be formulated as ratio of logical dataset size to physical dataset size as shown in Eq. (1). The amount of duplicate data sets an upper limit on the space savings which can be achieved by data deduplication.

$$deduplication\ ratio = \frac{bytes\ in}{bytes\ out}. \tag{1}$$

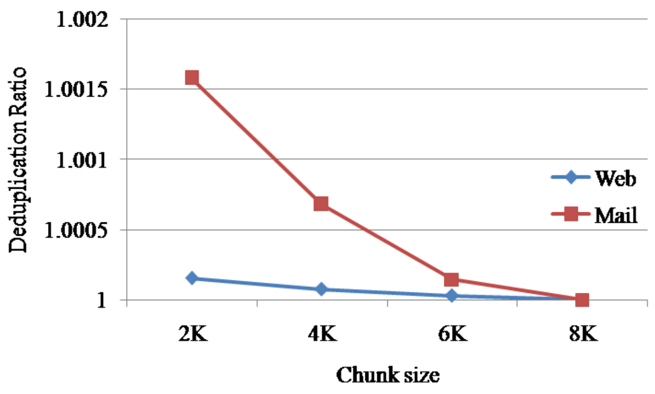
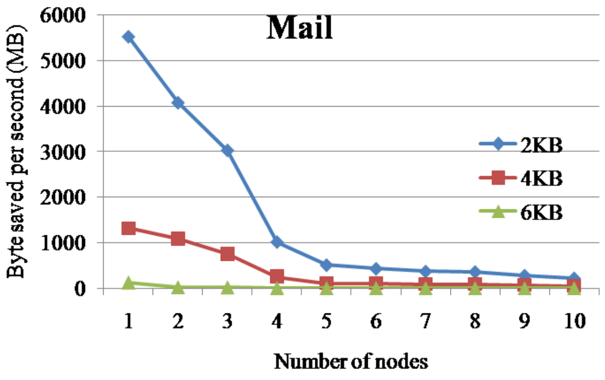


Fig. 10 Deduplication Ratio

The amount of duplicate data in a specific environment is determined by the characteristics and access patterns of the data and by the operational policies and practices in use. The space savings actually achieved depends on which data is deduplicated and on the effectiveness and efficiency of the specific technologies used to perform capacity optimization. Not surprisingly, actual deployment of specific technologies in specific environments determines the actual space savings. The deduplication ratio depends on how much the original data have duplicate sets. Figure 10 mentions the deduplication ratio according to the traces mentioned above.

Deduplication Efficiency. Deduplication efficiency is a simple metric that encompasses both capacity saving and system overhead in deduplication process. It depends on the original data duplication ratio and how many times the original data has been saved. It is calculated by the difference between the logical size L and the physical size P of the dataset divided by the deduplication process time T as expressed in Eq. (2).

$$deduplication\ efficiency = \frac{L-P}{T}\ . \tag{2}$$



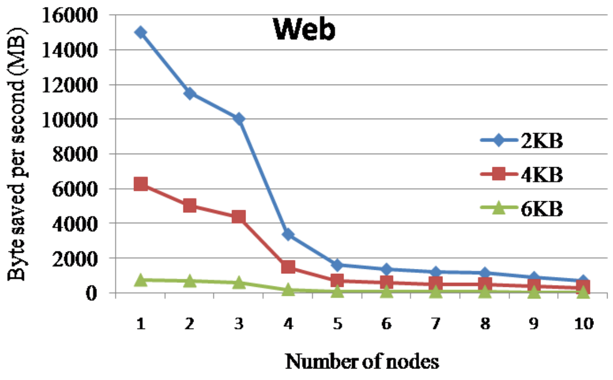
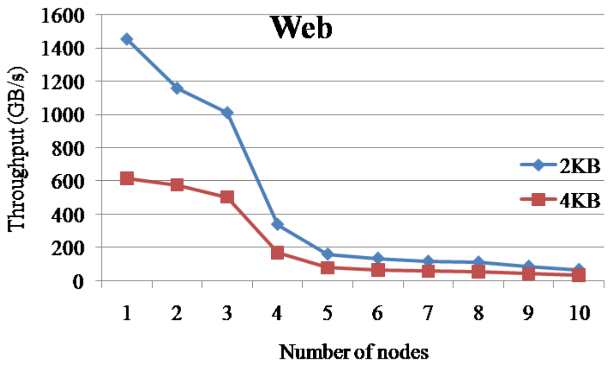


Fig. 11 Bytes saved of data deduplication

The Figure 11 shows the amount of bytes saved per second during data deduplication. The impact is proportional to the number of nodes in the cluster. However, the data deduplication efficiency result nearly the same in the cluster with more than four nodes. The deduplication efficiency for mail server and web server depending on the chunk size is also illustrated in Figure 13.

Deduplication Throughput. To determine the throughput of the deduplication system for the distributed storage system, the ratio of logical dataset size to deduplication process time is evaluated. Figure 12 shows the throughput of the system with different number of nodes in the cluster. The throughput results greatly decrease in the less than four-node cluster but the result differences is maintained smoothly in the cluster with more than four nodes.



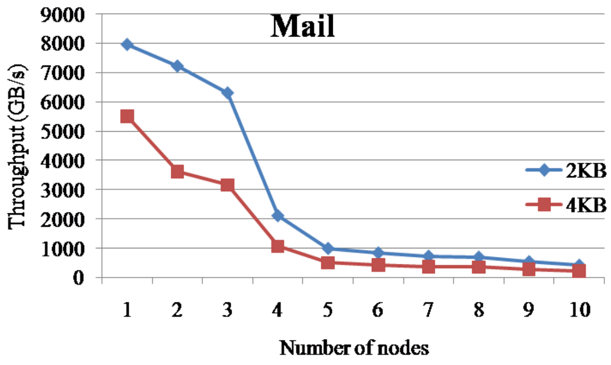


Fig. 12 Deduplication Throughput

The more overhead is occurred with smaller chunk size and the better throughput is achieved using larger chunk size as shown in Figure 14. From Figure 13 and 14, we can draw the conclusion that the deduplication efficiency and throughput is inversely proportional and the performances of the data deduplication system vary depending on the chunk size.

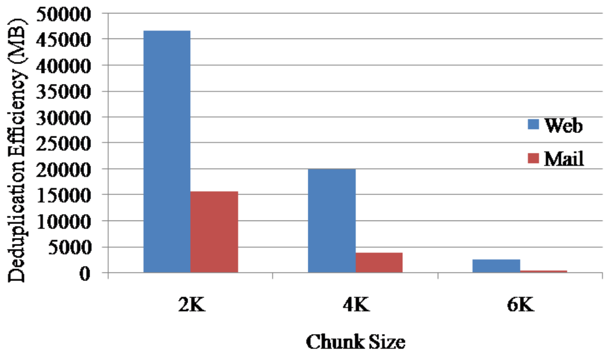


Fig. 13 Deduplication Efficiency

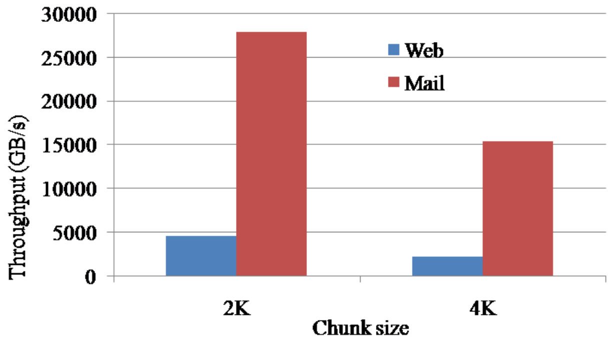


Fig. 14 Throughput measurement

Conclusion

In this paper, we present the efficient data deduplication method is presented to optimize the scale-out distributed storage capacity. To eliminate the impact of the disk bottleneck, it uses Bloom filter array (BFA) structure which also provides efficient indexing scheme in space saved deduplication method. We evaluate the system with various kinds of workloads and showed that the system can provide reasonable performance and storage saving for scale-out distributed storage system.

References

[1] A. Muthitacharoen, B. Chen, and D. Mazi`eres, “A low-bandwidth network file system”, in Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01), Banff, Canada, October 21-24, 2001, pp. 174-187.

[2] Advanced Storage Products Group, “Identifying the hidden risk of data deduplication: how the HYDRAstor solution proactively solves the problem”, Technical Report WP103-3 0709, NEC Corporation of America, 2009.

[3] 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 (FAST '08), San Jose, CA, USA, February 26-29, 2008, pp. 269-282.

[4] 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 conference on File and Storage Technologies (FAST '09), San Francisco, CA, USA, February 24-27, 2009, pp. 197–210.

[5] 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 (ICS '09), Yorktown Heights, NY, USA, June 8-12, 2009, pp. 370–379.

[6] D. Bhagwat, K. Eshghi, D. D. Long and M. Lillibridge, “Extreme Binning: Scalable, Parallel Deduplication for Chunk-based File Backup”, In Proceedings of IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '09), London, UK, September 21-23, 2009, pp.1-9.

[7] EMC Data Domain Global Deduplication Array. http://www.datadomain.com/products/global-deduplication-array.html

[8] FIU IO Dedup Traces. http://iotta.sniz.org/traces/391.

[9] IBM ProtecTIER Deduplication Gateway. http://www.03.ibm.com/systems/storage/tape/ts7650g/index.html

[10] 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 (FAST '07), San Jose, CA, February 13-16 2007, pp. 22-22.

[11] L. Kolb, A. Thor, and E. Rahm, “Dedoop: Efficient Deduplication with Hadoop”, Journal Proceedings of the VLDB, vol.5, no. 12, pp. 1878–1881, 2012.

[12] M. Lillibridge, K. Eshghi, D. Bhagwat, V. Deolalikar, G. Trezise and P. Camble, “Sparse indexing: large scale, inline deduplication using sampling and locality”, In Proccedings of the 7th Conference on File and storage technologies (FAST '09), San Francisco, USA, February 24-27, 2009, pp. 111–123.

[13] 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), vol. 60, no. 11, pp. 26-28, December 2012.

[14] P. Shilane, G. Wallace, M. Huang and W. Hsu, “Delta Compressed and Deduplicated Storage Using Stream-Informed Locality”, in Proceedings of the 4th USENIX conference on Hot Topics in Storage and File Systems, 2012, pp. 10-10.

[15] 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), San Jose, CA, February 14-17 2012, pp.49-63.

[16] 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), Atlanta, GA, April 19-23, 2010, pp.1-12.

[17] W. Xia, H. Jiang, D. Feng and Y. Hua, “SiLo: A Similarity-locality based Near-Exact Deduplication Scheme with Low RAM Overhead and High Throughput”, in Proceedings of the 2011 USENIX Conference on USENIX annual technical conference (USENIXATC'11), Portland, OR, USA, June 15-17, 2011, pp. 26-28.

[18] Y. Fu, H. Jiang, and N. Xiao, “A Scalable Inline Cluster Deduplication Framework for Big Data Protection”, in Proceedings of the 13th International Conference on Middleware (Middleware '12), Montreal, Quebec, Canada, December 3-7, 2012, pp. 354-373.

[19] Z. Sun, J. Shen and J. Yong, “DeDu: Building a Deduplication Storage System over Cloud Computing”, 15th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2011), Lausanne, Switzerland, June 8-10, 2011, pp 348 – 355.