Ensuring Reliability in Deduplicated Data by Erasure Coded Replication

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

Thandar Thein
University of Computer Studies, Yangon
thandartheinn@gmail.com

Abstract

As computer systems are taking more and more responsibilities in critical processes, the demand for storage is increasing due to widespread applications. Saving the digital information in a large disk is expensive and unreliable. As a result, if the disk fails all the data is lost. Therefore, the yearning for a better understanding of the system's reliability is ever increasing. In greatest hit storage environments, deduplication is applied as an effective technique to optimize the storage space utilization. Usually, the data deduplication impacts the bad result for the reliability of the storage system because of the information sharing.

In this paper, reliability guaranteed deduplication algorithm is proposed by considering reliability during the deduplication process. The deduplicated data are distributed to the storage pool by applying the consistent hash ring as a replicas placement strategy. The proposed mechanism is evaluated and the result is compared with pure replication and erasure coded replication. The proposed mechanism can provide the better storage utilization and the one hundred percent of assurance for demanded reliability level in compared with the existing systems.

Keywords

Deduplication, Distributed storage, Erasure coding, Reliability, Replication.

1. Introduction

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. Understanding the reliability of data is important because it's a big problem if the actual safety of data is lower than expectation, but too high a safety is also a problem, which means money is wasted on unnecessary devices. Space efficiency is also one of the primary concerns in the storage system.

Theoretically, replication in large-scale distributed system can solve most of the challenges of distributed system. It can improve the system performance such as availability, reliability and scalability. However, if the replication factor reaches a certain point, the system performance will level off and it stops rising and stays at the same level. As a consequence, it just spends the storage of the system and wastes cost. Many researches do data deduplication to reduce the redundant data and save the storage space in various environments. At other side, 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. So, in this paper, efficient reliability-aware data deduplication is proposed to provide the present huge amount of unstructured data storage.

The rest of this paper is organized as follows. Section 2 highlights the reliability analysis over pure replication and coding schemes and the consistent hashing for replica distribution. The mechanism for the assurance of reliability in deduplicated data is introduced in Section 3.

Performance evaluation is in Section 4 and Section 5 describes how to get the data reliability in the recent storage systems as related work. Finally we conclude the paper in Section 6.

2. Preliminary Concepts

Some concepts which make a prelude in designing the assurance of reliability in deduplicated data are presented in this section.

2.1 Redundancy Techniques

As the amount of data increases exponentially in large data storage systems, it is crucial to protect data from loss when storage devices fail to work. Reliability is usually obtained through redundant nodes in distributed storage systems. The simplest way of providing redundancy might be by repetition. Recently, both academic and industrial storage systems have addressed this issue by relying on erasure codes to tolerate component failures. Erasure codes provide space-optimal data redundancy to protect against data loss. A common use is to reliably store data in a distributed system, where erasure-coded data are kept in different nodes to tolerate node failures without losing data.

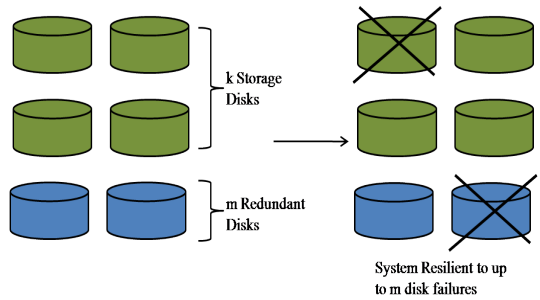

Figure 2.1 Data Resiliency

In an erasure coded system, a total of $n = k + m$ disks are employed, of which k hold data and m hold coding information as depicted in Figure 2.1. The act of encoding calculates the coding information from the data, and decoding reconstructs the data from surviving disks following one or more failures. Storage systems typically employ Maximum Distance Separable (MDS) codes, which ensure that the data can always be reconstructed as long as there are at least k disks that survive the failures.

2.2 Consistent Hashing

Consistent hashing is based on mapping each object to a point on the edge of a circle. The system maps each available machine to many pseudo-randomly distributed points on the edge of the same circle. Consistent hashing specifies how keys are to be assigned to nodes and how a node can discover the value for a given key by first locating the node responsible for that key. It maps the items to corresponding nodes where node's identifier is the hashing the node's IP address and the key's identifier is hashing the key.

According to the example illustrated in Figure 2.2, there are two nodes A and B and three objects 1–3 initially. The objects 3 and 1 are mapped to node A, object 2 to node B. When a node leaves the system, data will get mapped to their adjacent node (in clockwise direction) and when a node enters the system it will get hashed onto the ring and will overtake objects.

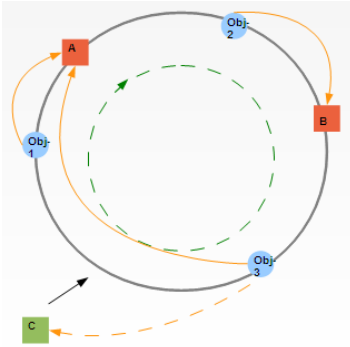

Figure 2.2 Example of Consistent Hashing

3. Assurance of Reliability in Deduplicated Data

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.

To build a capacity optimized mechanism for scale-out distributed storage system, it is important to solve the challenges of optimizing the storage capacity achieving data reliability with space efficiency. As a first part, an efficient data deduplication method in scale-out distributed storage system is presented [7, 8]. Since good data detection is very helpful for data deduplication, the deduplication scheme using BFA [6] is applied for the sake of space and look-up efficiency in distributed storage system. Moreover, the second part explores how to ensure the reliability of distributed storage system. Erasure coded redundancy is performed in the scale-out distributed storage environment using consistent hashing. It can provide data reliability with space savings.

In this paper, how to overcome the conflict between data deduplication and erasure coding techniques by adjusting the necessary reliability and capacity utilization is only focused. Since the storage systems usually use some data redundancy techniques to provide data reliability and the pure replication introduces the storage overhead, erasure coded replication is utilized in the proposed mechanism by distributing data replicas to the consistent hash ring.

3.1 Distribution of Data Replicas

In the proposed mechanism, after the deduplication process, replication manager gathers the unique chunks and the duplicated chunks which need more reliability in a container. The container (S_c bytes) is then divided into k blocks. Each of them is around $\frac{S_c}{k}$ bytes and the erasure coding process encodes them as n total blocks generating $n-k$ coding blocks. The illustrated procedure for encoding data replicas can be seen in Figure 3.1.

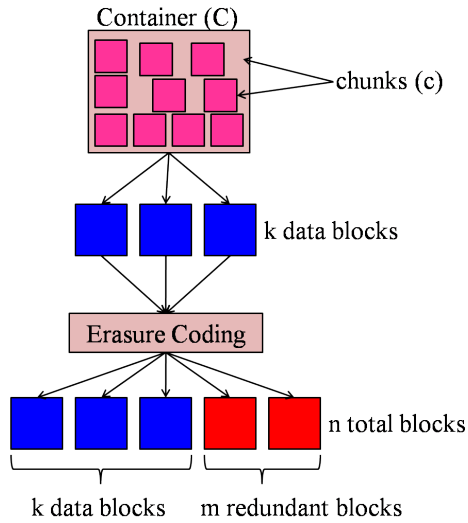

Figure 3.1 Encoding for Data Replicas

To put the encoded blocks (n) in the storage pool, the core concept of the consistent hashing is applied. A consistent hash function is the mapping of items to corresponding nodes. A node's identifier is defined by hashing the node's IP address, $Hash(IP)$, and the identifier of a key, which is an item, is produced by hashing the key, $Hash(key)$. During the process, <*key, value*> is submitted to a node of the ring which has the associated key. Each encoded block is in the form of <$Hash(n_{id})$, n_i>. n_{id} is the identifier of the blocks.

3.2 Reliability Model

One of the widely used methods to improve data reliability is distribution of data with redundant information on several storage devices. As a result, if a storage node fails, enough information can still regenerate the failed node or reconstruct the original file. In the distributed storage environment, reliability can be defined as tolerance to storage node failure while availability means promptly access to the original file and the storage efficiency is the amount of redundant information stored in the system. Reliability is defined as the probability that a device will perform its required function

under stated conditions for a specific period of time.

Table 3.1 Parameters Used In Reliability Model

Parameter	Description
p	probability that a node remains functional
q	probability that a node fails
n	total number of nodes
k	number of data nodes
m	number of parity nodes
$L(k,m)$	probability that a (k,m) value cannot be recovered

In the process of reliability aware replication, the threshold of the reliability is calculated using a reliability model. As summarized in Table 3.1, in the model, $p(t)$ is defined as the probability that a node remains functional after time t and the probability that a node fails before time t is described as $q(t) = 1 - p(t)$. As the total number of nodes is $n = k + m$, let $L(k,m)$ be the probability that a (k,m) value cannot be recovered after time t. During the calculation process, it is assumed that all nodes failures are independent. Based on these parameters, the following can be derived:

$$L(k,m) = \sum_{i=m+1}^{n} \binom{n}{i} p^{n-i}\, q^{i} \quad (1)$$

Then, the approximation can be taken by taking the first term of the definition of $L(k,m)$ because subsequent terms decrease by roughly a factor of q, which is close to 0, shown in the follow.

$$L(k,m) \approx \binom{n}{m+1} p^{k-1} q^{m+1} \quad (2)$$

3.3 Reliability-aware Deduplication

This paper intends to settle the dispute over conditions such as deduplication then reliability ("dedup then reliability") and the proposed deduplication plus reliability (dedup+reliability). "dedup then reliability" can face the data loss since it operates data deduplication first and then does the data reliability mechanism such as replication. "dedup+reliability" can address the above issue because it considers data deduplication and reliability at the same time and balances their different properties.

Algorithm: Reliability_aware_Deduplication

Reliability_aware_dedup(c_i^j)

// Algorithm for reliability in deduplication

input: each chunk of a file

1: if (num_c_i^j > 1) then //check if the chunk has more than one duplicate
2: if (R_c_i^j > eR_ c_i^j) then // check whether the demanded reliability level is higher than the existing one
3: store c_i^j to the container
4: update eR_ c_i^j
5: else
6: S_ c_i^j ← S_ c_i^j + 1 // increment the reference count
7: end if
8: else
9: store c_i^j to the container
10: end if

(R_c_i^j = demanded reliability, eR_ c_i^j = existing reliability, S_c_i^j = shared chunk)

Figure 3.2 Reliability-aware Dedupliaction Algorithm

The algorithm of reliability-aware deduplication works for each chunk of a file as presented in Figure 3.2. First of all, it is checked if the chunk c_i^j has more duplicates (line1). If there is more than one chunk, it is checked again whether the reliability level of c_i^j is higher than the existing ones. If so, put the chunk c_i^j to the container for further store and then update the reliability level. When the existing chunk reliability is enough for demanded reliability, the process does not need to store the chunk again and only share the existing one (line 6).

4. Performance Evaluation

The proposed reliability guaranteed deduplication system intends to get the demanded reliability maintaining the least storage space in scale-out distributed storage system. Therefore we focus on storage space utilization and how to achieve the reliability during the experiment. 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.

4.1 Theoretical Analysis of Redundant Data

Replication is a process where a whole object is replicated some number of times, thus providing protection if a copy of an object is lost or unavailable. In the case of replicating a whole object, the overhead would be 100% even for a single replica. Erasure coding is a process where data protection is provided by slicing an individual object in such a way that data protection can be achieved with greater storage efficiency that is some value less that 100%.

Erasure codes are space-efficient schemes to encode data into redundant fragments to protect against erasures of some of the fragments. Such erasure codes have been used traditionally in communication systems and more recently, in storage systems as a way to protect data against node crashes. It can also make the storage systems reliable.

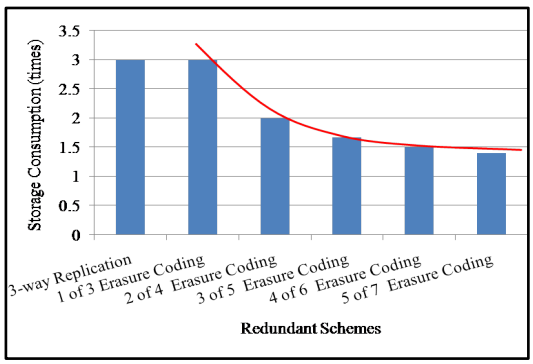

Figure 4.1 Storage Usages of Different Redundant Schemes with 2 Fault Tolerances

As can be seen in Figure 4.1, the system can tolerate 2 faults and the storage consumption is 3 times by using 3-way replication. By 3 of 5 erasure coding, while the system can also tolerate 2 faults, it needs only 1.6 times storage. So, less data is written over the network onto storage while tolerating the same number of faults. The extent of occupied storage depends on the type of redundancy in which there can be variants even with erasure coding techniques according to the configuration shown in Table 4.1.

Table 4.1 Different Configurations of Redundant Schemes

	3-way PR	1-3 EC	2-4 EC	3-5 EC	4-6 EC	5-7 EC
n	3	3	4	5	6	7
k	1	1	2	3	4	5
m	0	2	2	2	2	2

PR = pure replication, EC = erasure coding

4.2 Storage Space Utilization

Storage space utilization is a measure of how well the available data storage space in an enterprise is used. It measures how full the space is compared to its capacity. Utilization rates can be assessed in terms of both actual use and predicted use. At this point, one question comes out: What can be done to increase storage utilization rates? Increasing storage utilization can come in two areas; making the data more space efficient (deduplication) and making the capacity more efficient (redundancy mechanisms).

Capacity utilization rate is a metric which is used to compute the rate at which probable output levels are being met or used. The output is displayed as a percentage and it can give a proper insight into the general negligence that the organization is at a point of time. Capacity utilization rate is also called as operating rate. It also helps in verifying the level at which piece costs will rise.

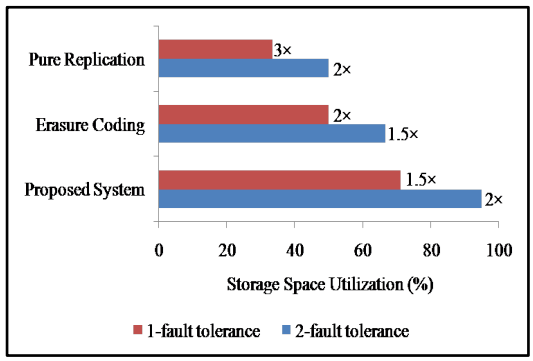

Figure 4.2 Utilization of Storage Space

Figure 4.2 shows the experimental results of storage capacity utilized by three different redundancy schemes such as pure replication, erasure coding and the proposed system. Beside it compares the storage space utilization for two kinds of fault tolerance level. Obviously, to stand for 1-fault tolerance, it needs only 1.5x storage overhead in erasure coding and the proposed system although there is 2x storage overhead for pure replication. But the storage space utilization for pure replication is only 50% and 30% in pure replication and the utilization is significantly better in erasure coding and the proposed system.

In Summary, the proposed system can provide the better storage utilization for any fault tolerance level. Moreover, it has better utilization of the storage capacity with the same storage overhead.

4.3 Reliability Guarantee

During the experiment for providing the demanded reliability level, the testing environment works with three input files which have some amount of duplicate data over its previous input.

Four different reliability level schemes are set up for the experiment:

(i) Upward: The demanded reliability level rises for each input. It means that the demanded reliability level for the second input file is higher than the first one and the third is also higher than the second.

(ii) Downward: There is a reverse concept from the above scheme.

(iii) Arch: The arch represents that the middle input has the peak demanded reliability level.

(iv) Concavity: The concavity has the opposite meaning of arch.

Figure 4.3 illustrates the comparison of assuring reliability level between different amounts of overlapped data. The percentage of reliability guarantee drops if the amount of duplicate data increases. However, the "downward" reliability level scheme can maintain full reliability guarantee because of its nature.

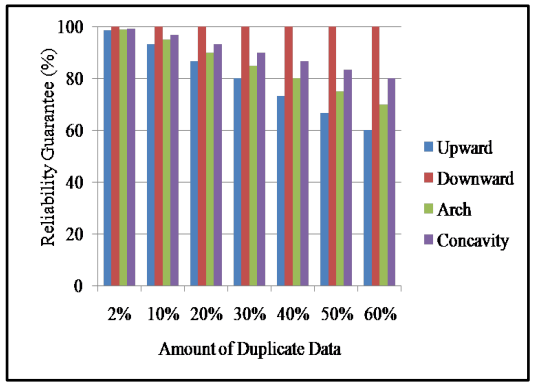

Figure 4.3 Percentage of Assuring the Demanded Reliability Level based on Deduplication

The assurance for demanded reliability level is analyzed in Figure 4.4. By analyzing with four different reliability schemes such as "upward", "downward", "arch" and "concavity", the "upward" can give the least reliability guarantee because it eliminates all duplicate data by making already existing data being shared without any consideration of demanded data reliability.

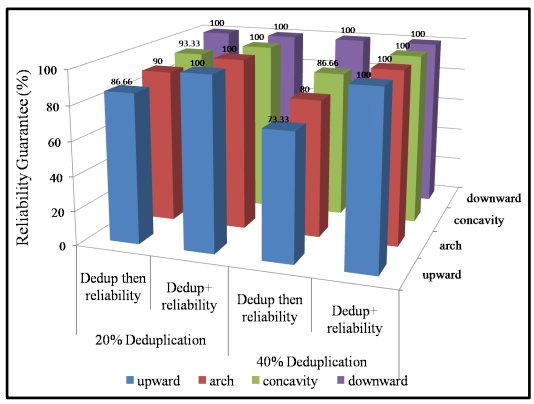

Figure 4.4 Assurance of Reliability

In conclusion, the combination of deduplication and reliability mechanisms proposed as DSCapOpt can provide the one hundred percent of assurance for demanded reliability level. The consideration of deduplication and reliability separately can only provide the reliability in the condition of decreasing the demanded reliability level

(downward). The similar results can be achieved with a double deduplication ratio.

5. Related Work

High reliability is increasingly critical for storage systems. However, achieving it at a reasonable cost can be a challenge, especially in large-scale systems built from commodity devices. Even when the devices are fairly reliable, failures are frequent in such systems due to the large number of devices they employ. Redundancy is used to improve reliability, either by replicating the data blocks, as in RAID-1 or replica-based distributed systems [1] or by storing additional information, as in RAID-5 or erasure-coded distributed systems [9]. This section surveys some previous literatures concerning the above issue.

Bhageat et al. [2] measured the importance of a chunk to determine the number of replicas for each chunk maintaining the minimum redundancy level. Their strategy balances the storage space savings and reliability. It requires half storage space than the approach of combining data mirroring and intra file compression. However, it does not consider the organization of the data chunk in large-scale storage environment. Moreover, the number of copies of a chunk is logarithmic to the popularity of that chunk.

HYDRAstor [3] is a scalable secondary storage which provides high availability, reliability and storage efficiency. It allows the chunks to be placed in different resilience class and there is no relationship to the number of replicas for each chunk. Data resiliency is provided with erasure codes on a DHT with virtual supernodes spanned over physical nodes. But it does not automate the choice of the resilience class.

Similarly, an archival storage called R-ADMAD [4] mentions that all chunks are equally important and they are distributed using erasure code with same redundancy configuration. Although it considers the organization of data chunks providing dynamic load balancing, it seems that the approach is deduplication then replication.

Li et al. [5] performed reliability analysis of deduplicated and erasure coded storage. They proposed deduplication should guarantee initially requested data reliability by using more robust erasure coding. Their main contribution is automatically deciding the proper resilience class for each chunk based on the importance of documents containing that chunk.

Nevertheless, the previous researches operated the reliability issue separately from the deduplication process. In this paper, the assurance of reliability in deduplicated data is presented and the reliability of data is achieved by the erasure coded replication scheme with consistent hashing. The requirement of the reliability level is taken into account during data deduplication to decide whether the chunks are shared or not.

6. Conclusion

In large storage systems, it is crucial to protect data from loss due to failures. Erasure codes lay the foundation of this protection, enabling systems to reconstruct lost data when components fail. Erasure codes can however impose significant performance overhead in two core operations: encoding, where coding information is calculated from newly written data, and decoding, where data is reconstructed after failures. Nevertheless, one fact that the consideration of the data deduplication and the data reliability separately cannot guarantee the demanded reliability is argued and the reliability guaranteed deduplication in scale-out distributed storage system to optimize the storage usage is proposed in this paper.

The extent of occupied capacity in storage system depends on the type of redundancy techniques. The proposed system can provide the best storage utilization with the same storage overhead when the storage capacity utilized by three different redundancy schemes is evaluated. According to the analysis of the assurance for demanded reliability level, the combination of deduplication and reliability mechanisms proposed can provide the one hundred percent of assurance for demanded reliability level.

6. References

[1] E. S. Andreas, A. Haeberlen, F. Dabek and H. Weatherspoon et al., "Proactive replication for data durability", in Proceedings of the 5th International Workshop on Peer-to-Peer Systems (IPTPS), Santa Barbara, CA, USA, February 27-28 2006.

[2] D. Bhagwat, K. Pollack, D. Long, T. Schwarz, E. Miller, and J. Paris, "Providing high reliability in a minimum redundancy archival storage system", in Proceedings of the 14th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), September 2006, pp. 413-421.

[3] 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.

[4] 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, June 8-12 2009, pp. 370–379.

[5] X. Li, M. Lillibridge, and M. Uysal, "Reliability analysis of deduplicated and erasure-coded storage," in Proceedings of HotMetrics, 2010.

[6] 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.

[7] M. P. Phyu and N. L. Thein, "Using Efficient Deduplication Method in Large scale Distributed Storage System", in Proceedings of the 11th International Conference on Computer Applications (ICCA, 2013), Yangon, Myanmar, February 2013, pp.89-93.

[8] M. P. Phyu and T. Thein, "Capacity Optimized Deduplication for Big Unstructured Data in Scale-out Distributed Storage System", International Journal of Information Engineering Research Institute (IERI), [Lecture Notes in Information Technology (ISSN: 2070-1918)], December 2013.

[9] R. Rodrigues and B. Liskov, "High availability in DHTs: Erasure coding vs. replication", in Proceedings of the 4th International Workshop on Peer-to-Peer Systems (IPTPS '05), Ithaca, NY, USA, February 24-25 2005, pp. 226-239.