RAID6-based Distributed Storage System Implementation

Kai Liu
School of Computer Engineering
Nanyang Technological University
Singapore 639798
kliu006@e.ntu.edu.sg

Bingshui Da
School of Computer Engineering
Nanyang Technological University
Singapore 639798
da0002ui@e.ntu.edu.sg

Jianglei Han
School of Computer Engineering
Nanyang Technological University
Singapore 639798
jhan011@e.ntu.edu.sg

I. Introduction

In distributed systems, data is partitioned and stored in storage devices (nodes) that are separated from each other in location. For example, in a small scale distributed system, a file could be stored in multiple hard disks. In large distributed systems, data segments could be stored in different data centers located in separate geographical locations. When a particular data is being accessed by an user, the system resembles data from partitions and response to the request. Logically, serving read and write request to a distributed file should be no different from accessing a local file, from user's perspective.

Two of the most important considerations when designing distributed storage system is performance and redundancy. The former refers to I/O speed and latency user may experience. It is subjected to system architecture design and quality of hardware and communication channels. Data redundancy provides backup so that probability of data access error due to storage system is reduced. The fault-tolerance of a system is system's capability of handling situations when the data segment needed is unavailable. Individual disk availability in a norm instead of an exception in distributed systems. Except hardware difficulties, a storage node goes unavailable during upgrading or when the workload is exceptionally high. In either cases, the system should be able to serve the data request without accessing the unavailable nodes.

In this project, we implemented a minimal distributed storage system using open source packages to realize low-level coding algorithms. The system performance is evaluated and compared with alternative implementations. The main features of the system developed include:

1) Store and access abstract data objects across storage nodes using RAID-6 for fault-tolerance
2) Ability to determine the failure of storage nodes at execution time
3) Ability to rebuild lost redundancy at a replacement storage node

The remaining part of this report is organized as the following: Background section introduces concepts and technologies that is relevant to the system design and development. System architecture and algorithms are covered in Implementation section. Experiment setup and results analysis are found in Experiment section, followed by conclusion.

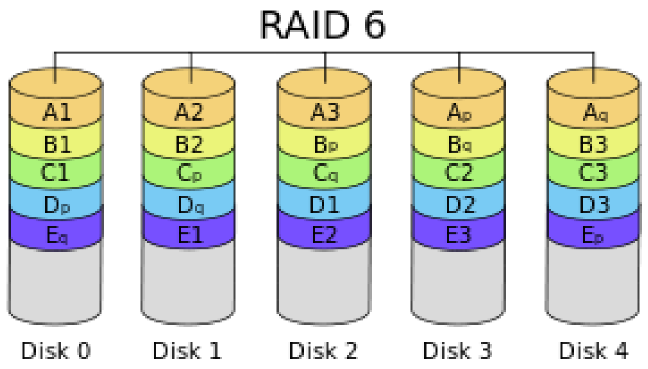

Fig. 1. Diagram of a RAID 6 storage system with 3 block level data strippings and 2 parities [2]

II. Background

A. RAID

Redundant array of independent disks (RAID) is originally designed as a visualization tool that establishes a whole storage space with multiple storage units. Different variation is developed to provide different level of data redundancy and I/O performance enhancement [1]. In a RAID system, a file is stripped and store in different physical disks depends on the actual variation. This characteristic makes RAID a natural strategy for distributed storage systems. These variations are *levels*, known by 'RAID' followed by a level number. For example, the most common types of RAID are RAID 0, RAID 1 to RAID 6. Different levels of RAID are differ in their strategies in file striping, mirroring and parity computation. We focus on RAID 6 in this project.

By definition, RAID 6 refers to setup that is resilient to two arbitrarily disk failure. The resiliency here means it is capable of carry on read and write request to any logic disk in the RAID system. Figure 1 shows an example of a RAID 6 storage system with data blocks A, B, C, D and E, each block is stripped into three parts (1-3) and two parities p and q. In the example, parities are being stored in all available disks. An alternative scheme is to store p segments and q segments in two disks separated from the data parts. The parities are computed independently be performed by constructing and solving linear equations with one or two variables.

B. Galois field

One of the parity is computed with a linear or XOR Besides linearity, parity computations is preferred to be performed in a Galois field, or a finite field that the size of membership is limited to p^k, where p is a prime number and k is a positive integer.

For example, the range of valid numbers in a Galois field $GF(2^8)$ is 0 to $2^8 - 1 = 255$, and the

C. Reed-Solomen Coding

In general, there are two main strategies to guarantee certain level of fault-tolerance. The first is full duplication, where all storage nodes have a independent mirror backup. In case of access failure, the backup copy can be used. The advantage of such design is, for a single fault, data recovery takes exactly the same read as the original request, incurring no additional read overhead. However, it is less cost-efficient, using only $1/2$ of the total hardware effectively in data storage. The second is using erasure coding

III. Implementations

A. System Architecture

B. Pseudo Code

IV. Experiments

Compared 2+2, 3+2, 4+2, 5+2, 6+2

A. Storage

B. Recoverability

Raid-6 recoverability theory

C. Speed

V. Conclusion

to conclude

References

[1] R. H. Arpaci-Dusseau and A. C. Arpaci-Dusseau, *Operating systems: Three easy pieces*, 2012.

[2] Wikipedia, "Standard raid levels," 2015, [Online; accessed 13-Nov-2015]. [Online]. Available: https://en.wikipedia.org/wiki/Standard_RAID_levels#RAID_6