

Characterizing Declustered Software RAID for Enhancing Storage Reliability and Performance

Extended Abstract

Zhi Qiao

Department of Computer Science and Engineering
University of North Texas
zhiqiao@my.unt.edu

Hsing-bung Chen

HPC-DES
Los Alamos National Laboratory
hbchen@lanl.gov

Song Fu

Department of Computer Science and Engineering
University of North Texas
song.fu@unt.edu

Bradley Settlemyer

HPC-DES
Los Alamos National Laboratory
bws@lanl.gov

ABSTRACT

Redundant array of independent disks (RAID) has been widely used to address the reliability issue in storage systems. As the scale of modern storage systems continues growing, disk failure becomes the norm. With the ever-increasing of disk capacity, RAID recovery based on disk rebuild becomes more costly, which causes significant performance degradation and even unavailability of storage systems. Declustered parity and data placement in RAID aims to enhance the recovery performance by shuffling data among all disks in a RAID group. All disks in the RAID group participate in data reconstruction, which leads to reduction of the RAID rebuild time. In this work, we extensively evaluate declustered RAID in terms of the performance of application I/O and recovery time. Our experimental results in ZFS show that the speedup of declustered RAID over traditional RAID is sub-linear to the number of disks in the storage pool.

1 INTRODUCTION

As the scale and complexity of storage systems continue to increase, reliability assurance of storage systems becomes more challenging. RAID offers a cost-effective technology to improve the throughput and redundancy. However, as the capacity of disk drives grows by 40% per year (i.e., Kryder's Law), disk rebuild time in a RAID group takes longer time, for example, the rebuild time for a 4-TB drive is almost 10 hours and that for a 8-TB drive can reach several days. Such prolonged rebuild time leads to possible data loss as additional disk failures are likely to occur during a rebuild.

To reduce the recovery time and protect data from data loss, declustered RAID distributes data and parities among drives into multiple arrays. All drives including the spares participate in disk rebuilds. To understand the performance of declustered RAID in high performance computing (HPC) environments, we test declustered RAID in ZFS on a storage platform at Los Alamos National Laboratory, and evaluate its performance with regard to application I/O and recovery time. We compare the performance of declustered RAID with that of software RAID in ZFS. The experimental results show that the recovery throughput of declustered RAID is many

times higher than RAIDZ. As part of our project, we are also investigating other approaches to improve RAID access and recovery performance, such as vectorizing Galois Field multiplications for parity reconstruction. In this poster, we present our preliminary results on performance evaluation of declustered RAID.

At Los Alamos National Laboratory, we build a storage test platform and vary a number of parameters to test declustered RAID, such as stripe width, parity ratio, storage utilization, access pattern, and redundancy level. We run I/O benchmarks and measure the storage performance including throughput, access time, file/object creation/modification/access time, recovery time and throughput, and average wear during recovery. The experimental results show that 1) Declustered RAID evenly distributes data, parity, and spare blocks among drives in a RAID group. Data organization based on *permutation development* distributes wears among storage devices and enhances data parallelism. 2) Declustered RAID significantly reduces RAID recovery time. The speedup is sub-linear with regard to the number of drives involved in the recovery process. 3) Instead of using dedicated spare drives, declustered RAID uses spare blocks distributed among all drives for recovery. As a result, I/O writes can be performed in parallel, avoid performance bottleneck. We believe exascale storage systems can take advantage of declustered RAID to achieve faster I/O access and lower recovery overhead.

2 DECLUSTERED RAID ORGANIZATION

In a traditional disk array, a number of consecutive disk blocks are grouped into a *stripe* which contains user data and parity. Several spare drives are kept as a standby in case of disk failures. When a block in a stripe cannot be accessed, all other blocks in that stripe will be read and used to reconstruct the inaccessible block. Therefore, when a drive fails, data and parity blocks are read from the remaining, available drives are used to compute the lost blocks which are written to a single spare drive. The intensive I/O read and write during RAID recovery leads to degraded storage performance for user applications or storage unavailability.

A declustered RAID shuffles its data placement layout in a way that regardless of which disk fails, the recovery workload is distributed uniformly among the available drives. In addition, the spare drives also participate in data read and write by distributed spare blocks among all drives in a RAID group. Thus, each drive in

a declustered RAID contains data blocks, parity blocks, and spare blocks, and those blocks are distributed evenly among all drives in the storage pool.

We use dRAID [1] as an implementation of declustered RAID in our study. It is based on Permutation Development Data Layout (PDDL) [5] which uses an efficient mapping of stripe blocks to drives. The traditional Parity Decustering [3] method uses layout table to determine the locations of blocks, which takes additional processing time and extra storage space, and Random Permutation [4] distributes blocks among drives using pseudo-random algorithms which cannot guarantee uniform distribution of stripe blocks. In contrast, PDDL employs a randomly generated base permutation as the initial mapping for the first stripe, and then gradually develops other stripes on the fly based on the previous mappings. PDDL achieves uniform distribution of stripe blocks and avoids costly table lookup.

3 PERFORMANCE EVALUATION

Our qualitative analysis reveals the advantages of declustered RAID over software RAID (i.e., RAIDZ) in ZFS are as follows.

- Higher data parallelism for application I/O as spare drives participate in read and write operations. There is no idle drive.
- Faster RAID recovery as all available drives are involved in reconstructing data which is written to spare blocks distributed on all drives.
- The recovery process scales with the number of drives that contribute to block read and write for block reconstruction.

To quantify the preceding advantages of declustered RAID, we set up a storage test platform at Los Alamos National Laboratory. Figures 1 and 2 compare the I/O and recovery throughput of dRAID and RAIDZ using the equivalent redundancy configuration. The I/O performance only improved slightly, since we use one spare drive for dRAID in this test. We also observe that the throughput of dRAID becomes stable when more storage space is used. In contrast, raidz suffers from performance degradation as the storage utilization increases. When 71% of disk space is used, the throughput of dRAID is 14.6% greater than that of RAIDZ. Figure 2 compares the recovery performance of different RAID configurations. From the first two tests we can see that the recovery throughput of dRAID1 and dRAID3 using 3 RAID array in same storage pool is 2.69 and 2.22 times as high as that of RAIDZ1 and RAIDZ3, respectively. While the third test using 6 RAID 5 array in the storage pool, providing same capacity as the second test. The result shows dRAID1 in this configuration outperforms RAIDZ1 by 4.35 times.

To modeling the dRAID recovery speed improvement, we use d to denote the number of data blocks, p is the number of parity blocks, s is the number of spare drives, and n is the total number of drives. Then, we have $k = \frac{n-s}{d+p}$, where k is the number of redundant arrays in a zpool. For example, the first two group in Figure 2, $k = 3$. The speedup Φ of dRAID over RAIDZ can be calculated as

$$\Phi = H \times \frac{k \times (d + p) + s - f}{d + p - f} \quad (1)$$

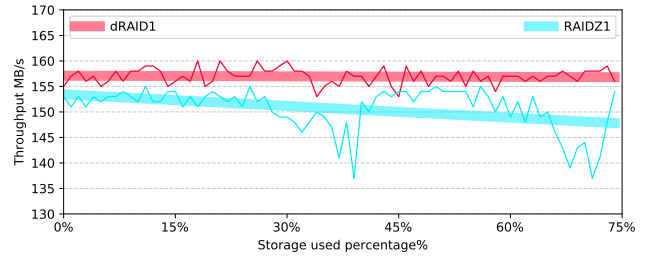


Figure 1: Comparison of I/O performance between dRAID1 and RAIDZ1. The storage configuration is 1:1 as parity ratio, 1 spare drive, and 7 drives in total. For RAIDZ1, 2 out of 7 drives are active during the ZFS recovery process, reading from available drives and writing to the spare. For dRAID1, all 7 drives perform reads and writes together, hence the aggregated I/Os are higher.

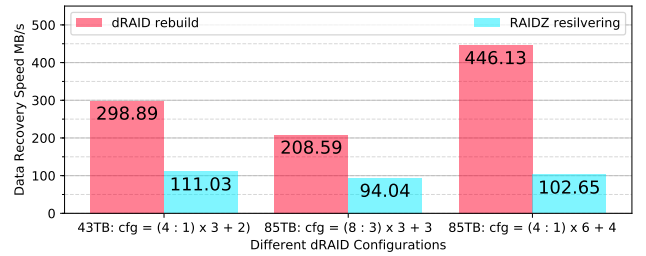


Figure 2: Comparison of data recovery speed between dRAID and RAIDZ. First test use 4:1 as parity ratio for dRAID1 and RAIDZ1 with 17 drives in total. Second test use 8:3 as ratio for dRAID3 and RAIDZ3 with 36 drives in total. Third test use 4:1 as ratio but 6 arrays in storage pool with 34 drives in total. For each test the storage utilization is 10%.

where H denotes the overhead coefficient and $0 < H < 1$. From our experimental results, we have the value of $H \rightarrow 1$, indicating the overhead is low.

4 CONCLUSIONS

Declassed RAID achieves an efficient and less intrusive data reconstruction for parallel I/O and fast RAID recovery. With Permutation Development Data Layout (PDDL) based data organization, declustered RAID shows significant performance improvement compared with traditional RAIDZ in ZFS. It uses small storage space and causes a low overhead. It can be used in the coming exascale storage systems for performance and reliability assurance.

Our ongoing work includes testing the scalability of dRAID deployed on multiple nodes over an infiniband network, conducting stress tests, and performing fault injection in a petabyte storage system. We notice that declustered RAID skips checksum verification, which can cause corrupted data undetected during RAID recovery. We plan to address this problem in our future research.

REFERENCES

- [1] 2018. The dRAID Project. <https://github.com/thegreatzoo/zfs>.

- [2] Siu-Cheung Chau and Ada Wai-Chee Fu. 2000. A gracefully degradable declustered RAID architecture with near optimal maximal read and write parallelism. In *null*. IEEE, 309.
- [3] Mark Holland and Garth A Gibson. 1992. *Parity declustering for continuous operation in redundant disk arrays*. Vol. 27. ACM.
- [4] Arif Merchant and Philip S. Yu. 1996. Analytic Modeling of Clustered RAID with Mapping Based on Nearly Random Permutation. *IEEE Trans. Comput.* 45, 3 (March 1996), 367–373. <https://doi.org/10.1109/12.485575>
- [5] Thomas JE Schwarz, Jesse Steinberg, and Walter A Burkhard. 1999. Permutation development data layout (PDDL). In *High-Performance Computer Architecture, 1999. Proceedings. Fifth International Symposium On*. IEEE, 214–217.