Joint Conference of 1996 International Computer Symposium
December 19~21, Kaohsiung, Taiwan, R.O.C.

A New Reconstruction Algorithm for RAID 5 using Parity Sparing

Hyeon-Cheol Kim and Myong-Soon Park

Department of Computer Science
Korea University
Sungbuk-ku, Seoul 136-701, Korea
khc,myongsp@cslab1.korea.ac.kr

Abstract

Sparing schemes were proposed to improve the reliability of RAID 5 disk arrays. The usage of a spare disk makes the reliability of disk arrays dependent on the reconstruction time during which data of the failed disk are reconstructed on the spare disk space using the surviving disks. Several reconstruction algorithms have been suggested for reducing the reconstruction time and thus improving the reliability of disk arrays. Parity sparing shows better performance than other sparing schemes, but the merging process during the reconstruction cause the scheme to have rather long reconstruction time. We propose a new reconstruction algorithm, called variant-block piggybacking, for RAID level 5 using parity sparing. We have found that though variant-block piggybacking degrades performance a little, it dramatically reduces reconstruction time and thus improves the reliability of disk arrays using parity sparing.

1 Introduction

RAID(Redundant Arrays of Inexpensive Disks) level 5 is one of the most cost-effective, reliable storage system with high performance. It does not lose any data though a failure happens[1]. Using the parity-based data protection and block-interleaving, RAID level 5 shows high reliability and good performance[2].

RAID 5 disk arrays have been proposed as a hardware solution for providing high data availability. When a disk in a RAID 5 array fails, the data on that disk can still be accessed, albeit less efficiently, by reading surviving disks and regenerating the data via the exclusive-OR operation. The array cannot, however, survive a second disk failure. Therefore it is necessary that the data on the failed disk be restored to a spare disk in a timely manner. In case of a disk failure, sparing scheme rebuilds the data of the failed disk on a spare disk and thus improves the reliability of disk arrays[3]. Several sparing schemes have been proposed to improve the performance and reliability of disk arrays by utilizing the spare disk[2].

The reconstruction process of disk arrays means to reconstruct the contents of the failed disk on spare disk area using surviving disks. In a disk array using sparing scheme, the reliability is tightly related to the reconstruction time in reconstruction mode. Reconstruction algorithm improves the reliability of disk

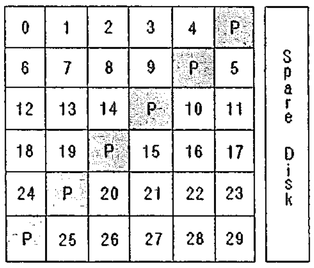

Figure 1: Array organization in hot sparing

array by reducing reconstruction time. Until now, several reconstruction algorithms have been proposed for reconstruction. In this paper, we propose a new reconstruction algorithm, called variant-block piggybacking, evaluate the performance and reliability, and compare variant-block piggybacking with the previous algorithms.

For performance and reliability comparison, we consider baseline strategy, redirection of reads, piggybacking of writes, minimal operation, and variant-block piggybacking proposed in this paper.

The rest of the paper is organized as follows. Section 2 acknowledges previous work in the field. Section 3 describes the proposed algorithm. Section 4 describes the simulator and workload used. Section 5 compares reconstruction and response times for the proposed algorithm and the other reconstruction algorithms. Section 6 presents our conclusion and future work.

2 Related Work

RAID level 5 does not lose any data though a failure happens. Without immediate and fast reconstruction, however, it may lose data by an additional disk failure. With a spare disk, the reconstruction process starts immediately after the failure, and the data in the failed disk are reconstructed from the surviving disks to the spare disk. Therefore, with a spare disk, disk arrays have high reliability because of having the less possibility of an additional disk failure.

Several sparing schemes were introduced according to the usage of a spare disk. The traditional approach

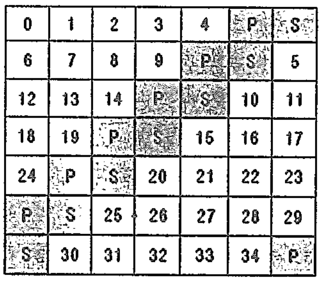

Figure 2: Array organization in distributed sparing

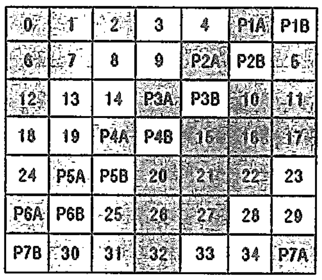

Figure 3: Array organization in parity sparing

is hot sparing[3] in Figure 1. In hot sparing, the spare disk is idle most of time and does not contribute to the normal operation of the system. But with a disk failure, it is used instead of the failed disk. In distributed sparing[4] in Figure 2, the spare disk is used to store data and parity. Instead, spare blocks are distributed evenly across the disks in the disk array. Spare blocks in each disk in disk arrays play the role of a spare disk when a failure occurs. Additionally, because each disk is partially empty, each disk failure requires less work to reconstruct the contents of the failed disk. In parity sparing[6] in Figure 3, the spare blocks are used to store another parity data. In parity sparing, a parity group is splitted into two small parity groups, and the spare block and the parity block are used as the parity blocks for two small parity groups.

With a spare disk, disk arrays have several different phases of operation: normal mode, failure mode, reconstruction mode, reconfigured mode, and restoration mode. Menon and Mattson identified these modes of operation in [5]. Normal mode is the period during which all the disks in the disk array are functional. Failure mode is the period during which a disk has failed and no reconstruction process begins. Reconstruction mode is the period during which reconstruction of a failed disk is in progress. Reconfigured mode is the period after the reconstruction process finished the reconstruction process of the data on the failed disk, but before a new spare replaces the failed disk. Restoration mode is the period during which a new spare is brought into the system to replace a failed disk. After the restoration mode, the disk array operates in normal mode.

A disk array in failure mode and reconstruction mode cannot survive a second disk failure. Thus, it is important for the disk array to enter reconstruction mode and reconstruct the failed disk as quickly as possible. Several reconstruction algorithms were proposed to reduce reconstruction time and thus improve the reliability of disk array. Muntz and Lui [8] proposed two reconstruction algorithms. In the first algorithm, called redirection of reads, when user's read request to the failed disk causes a data block to be reconstructed, the reconstructed data block is written to the spare disk as well as delivered to the requesting process. In the second algorithm, piggybacking of writes, when a user write request to the failed disk causes a data block to be reconstructed, the reconstructed data block is written to the spare drive as well as delivered to the requesting process. Without reconstruction algorithm, it is called baseline strategy. Minimal operation[7] uses redirection of reads and piggybacking of writes together. J. Chandy in [7] evaluates minimal operation through simulation and shows more improved reliability.

The reconstruction algorithms described above are the reducing methods of disk access number by using user request in reconstruction mode. M. Holland in [9] proposed multiple points of reconstruction besides the above algorithms. The motivation of this method is the fact that the load of reconstruction isn't evenly distributed across all disks. Unlike reconstructing one parity group sequentially in previous methods, multiple points of reconstruction reconstruct several parity groups in parallel and thus reduce the reconstruction time. This method enhances the utilization of all disks, reduces the possibility of sudden disk head movement, and uses disk arrays more effectively. However, since there is more possibility of reconstruction at that time of user request, response time of user request may increase. The number of parity groups reconstructed in parallel is called reconstruction thread in section 5.

In reconstruction mode, reconstruction algorithms can reduce the reconstruction time but increases the response time of user request. The target of reconstruction algorithm is to reduce reconstruction time and additionally reduce performance degradation as much as possible.

3 Variant-Block Piggybacking

Parity sparing scheme shows better performance than other sparing schemes since it uses the spare disk without disk failure. Parity sparing shows less performance degradation in failure mode and better performance in reconstruction mode than shown in other schemes. However, parity sparing shows rather bad reliability, since the merging process during the reconstruction cause the scheme to have rather long reconstruction time. In parity sparing, in order to improve the reliability, it is important to reduce the reconstruction time. The proposed algorithm, called variant-block piggybacking, focuses on the reduction of reconstruction time in parity sparing scheme.

Although track piggybacking algorithm was proposed in [10], it was only limited to hot sparing scheme. In parity sparing, however, track piggyback-

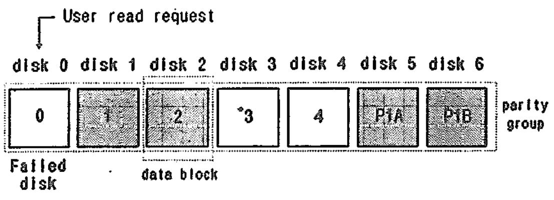

Figure 4: Read accesses for reconstruction with read request to the failed disk

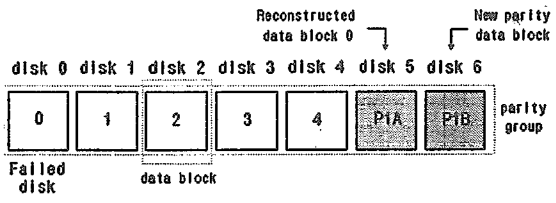
Reconstructed data block 0
New parity data block
disk 0 disk 1 disk 2 disk 3 disk 4 disk 5 disk 6
0
1
2
3
4
P1A
P1B
parity group
Failed disk
data block

Figure 5: Write accesses for reconstruction with read request to the failed disk

ing algorithm can't be used since data blocks and parity blocks are evenly distributed across all disks unlike hot sparing. So, block-based reconstruction has been used in parity sparing. Variant-block piggybacking shows single block access or continuous two block access according to the location of accessed data and parity blocks in disk arrays.

Figure 4 and 5 shows the reconstruction process of minimal operation when there is a read request to the failed disk. Figure 4 shows read accesses necessary to reconstruct the data block of failed disk. We assume that disk 0 is failed. If the read request to the data block 0 occurs, the shaded blocks should be read to service the requested data block 0, reconstruct the data block on a spare block, and generate new parity block. For example, to recover the data block 0, data block 1, 2, and parity block P1A are read and then exclusive-ORed. To generate new parity, P1B is read and then exclusive-ORed with P1A.

Figure 5 shows the writes of the reconstructed data block 0 and the new parity on disk arrays. The reconstructed data block 0 is written to the block P1A, and the new parity is written to the block P1B.

Figure 6 and 7 shows the proposed algorithm, called variant-block piggybacking. If there is a read request to a data block on the failed disk, the parity group including the requested data block and the continuous parity group in the same track are reconstructed together. Figure 6 shows the two continuous parity groups. The shaded blocks have to be read to do reconstruction process. In this case, there exist continuous blocks which can be read with one disk access. Of course, these blocks exist continuously at the same track on a disk. Figure 7 shows the writes of the reconstructed data block and the new parity block. Like the read request, continuous write blocks are written with one disk access. With the proposed algorithm, we can reconstruct the continuous two parity groups together through single block and continuous block accesses, reconstruct more quickly by reducing seek time and rotational delay and thus improve the reliability of disk arrays.

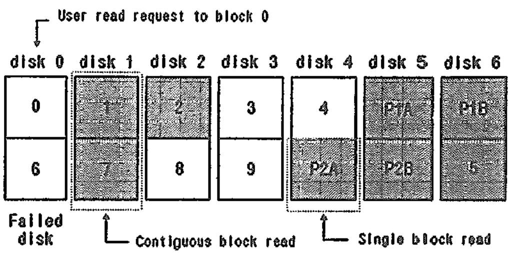
User read request to block 0
disk 0 disk 1 disk 2 disk 3 disk 4 disk 5 disk 6
0
1
2
3
4
P1A
P1B
6
7
8
9
P2A
P2B
5
Failed disk
Contiguous block read
Single block read

Figure 6: Read accesses for reconstruction of continuous parity groups with read request to the failed disk

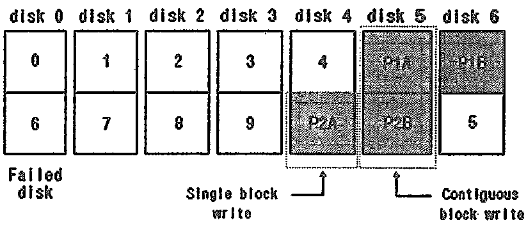

Figure 7: Write accesses for reconstruction of continuous parity groups with read request to the failed disk

Figure 8 and 9 shows read and write accesses per one track when there is a read request to the failed disk block. On reconstructing one parity group, the reconstruction of the continuous parity groups occurs simultaneously.

Figure 10 and 11 shows the read and write accesses per one track when there is a write request to the failed disk block. As the operation of read request, single block and continuous block accesses occur when there is a write request to the failed disk.

In summary, a detailed description of the algorithm is given as follows. We assume that one disk of RAID 5 using parity sparing is failed and a user read/write request to the failed disk block have to be delivered.

Step 1. The single or contiguous data/parity blocks of the failed parity group and contiguous parity group

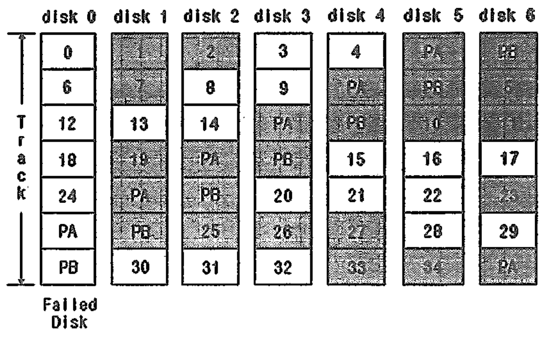

Figure 8: Read accesses for reconstruction with read request to the failed disk (track)

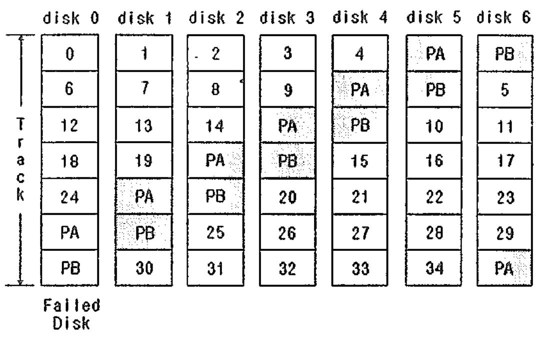

Figure 9: Write accesses for reconstruction with read request to the failed disk (track)

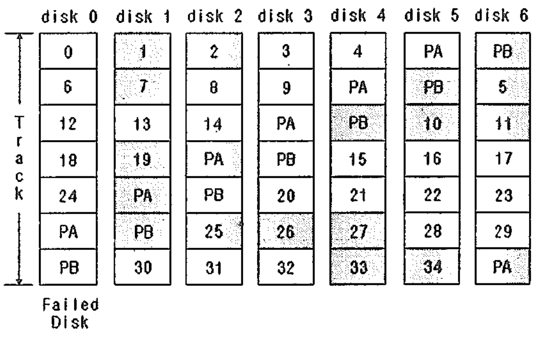

Figure 10: Read accesses for reconstruction with write request to the failed disk (track)

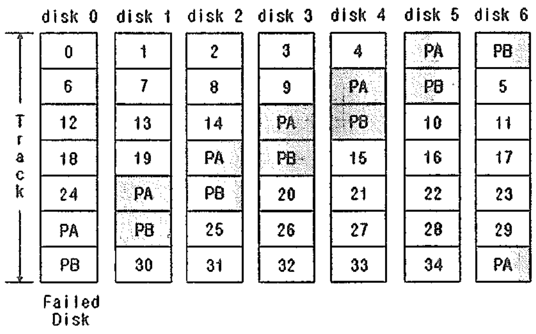

Figure 11: Write accesses for reconstruction with write request to the failed disk (track)

Table 1 Disk parameters

bytes per sector	512
sectors per track	112
tracks per cylinder	20
number of cylinders	2500
revolution time	8.3 ms
cylinder switch time	1.8 ms
average seek time	10.0 ms
maximum seek time	20.0 ms

are read to reconstruct the failed data block and the deliver the block.

Step 2. The data and parity blocks have to be exclusive-ORed according to their relation. The failed data block is reconstructed by using exclusive-OR operation is delivered to user request.

Step 3. The reconstructed data and parity blocks are written to two parity spaces of RAID 5 using parity sparing. If the contiguous blocks exist, the blocks can be written to disk at a time. After these operations, RAID 5 using parity sparing is changed to typical RAID 5.

Variant-block piggybacking reconstruct the continuous two parity groups included in the same track, when there exists read or write request to the failed disk block. In continuous parity groups, continuous blocks can be accessed at a time. Through the continuous block read and write accesses, we can reduce seek time and rotational delay and thus improve the reliability of disk array. Using variant-block piggybacking, we can dramatically improve the reliability of disk arrays using parity sparing.

4 Simulation Model

In this section, the simulated system model is described. We assume that a disk array is composed of 7 disks including a spare disk. Data and parity blocks are placed according to the left-symmetric placement[11]. We consider the 5 algorithms, baseline strategy, redirection of reads, piggybacking of writes, minimal operation, and the proposed algorithm. Disk parameters considered for the simulation are given in Table 1.

To model a seek time cost function, we used below non-linear model reported in [11].

$$ST = \begin{cases} 0, & SD = 0 \\ a\sqrt{SD-1} + b(SD-1) + c, & SD > 0 \end{cases}$$

ST : *Seek Time*
SD : *Seek Distance*

The values of constants a and b in above model are 0.2 and 0.0032 respectively, according to disk parameters. Constant c means minimum seek time, namely, cylinder switch time in table 1. Seek distance means the number of cylinders between the cylinder number where the disk head is placed to serve the previous request and the cylinder number where the head must be moved to serve the current request. The disks are modeled to have two queues of requests. One of the queues is the user request queue which is operational at all times. The other queue, the reconstruction queue, is operational only during the reconstruction or restoration process. The user request queue has a higher priority over the reconstruction queue. The requests in the reconstruction queue are served only if the user request queue is empty. However, once a reconstruction request is started, it's not preempted until completion.

Also, we assume that disk drives support a split access operation. We used C-LOOK policy for serving the requests at a disk queue.

There are many cases where a user request needs several disk accesses. The response time of those requests is the time elapsed from the time the request is issued to the time when the last access completes. The response times during reconstruction mode of operation are the averages of response times of all user requests served during that time. Reconstruction time, the time spent in the reconstruction mode is measured. It is important to keep the reconstruction time to be as small as possible to avoid data loss by an additional failure during this time.

To measure the performance according to the size of workload, requests rates are varied from 20 I/Os/sec to 120 I/Os/sec. In these cases, the requests are assumed to be read requests with 70 percent probability and writes requests with 30 percent probability. Requests are assumed to be uniformly distributed over all the disks and over all the blocks in a disk. We assume that requests arrive with a discrete uniform distribution.

The simulator was implemented using smpl[12], a C based simulation library. The results obtained through simulations are presented in the next section.

5 Simulation Results

Figure 12 shows average response times of reconstruction algorithms when the reconstruction thread is 1. Under low workload from 20 to 100 I/Os/sec, the average response time of variant-block piggybacking shows shorter response time than that of minimal operation by the reduction of the read/write response time to the blocks already reconstructed. However, under high workload over 100 I/Os/sec, the average response time is longer than that of minimal operation because of relatively long read/write time to the continuous block. With continuous block access, the proposed algorithm shows a little performance degradation under high workload.

Figure 13 shows average response times when the reconstruction thread is 4. All the response times of reconstruction algorithms are a little longer than reconstruction thread 1 in Figure 12. The increased reconstruction thread can help to reduce the reconstruction time, but degrades the performance by increasing the average response time.

Figure 14 shows average response times when the reconstruction thread is 8. The average response times are longer than those of reconstruction thread

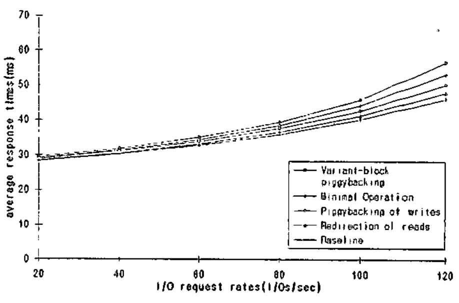

Figure 12: Average response time (reconstruction thread 1)

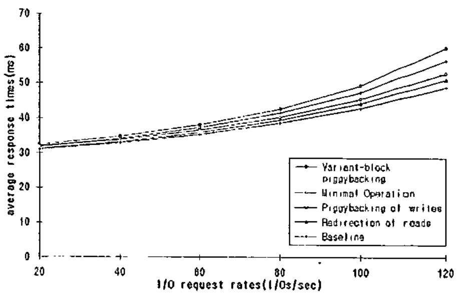

Figure 13: Average response time (reconstruction thread 4)

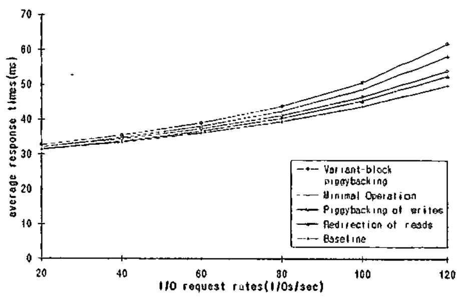

Figure 14: Average response time (reconstruction thread 8)

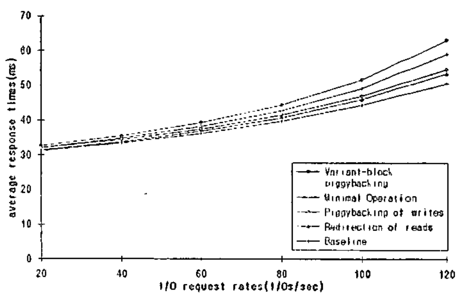

Figure 15: Average response time (reconstruction thread 16)

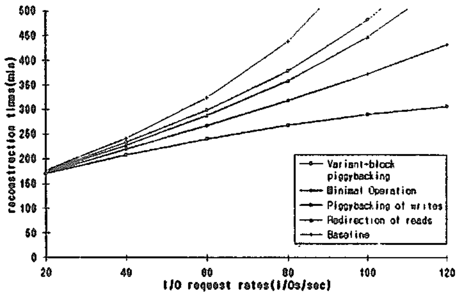

Figure 16: Reconstruction time (reconstruction thread 1)

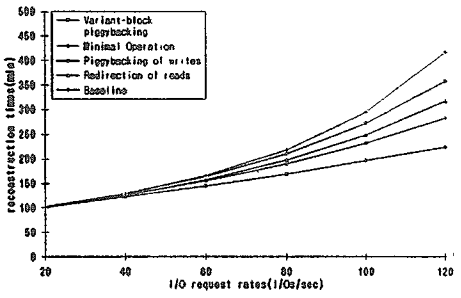

Figure 17: Reconstruction time (reconstruction thread 4)

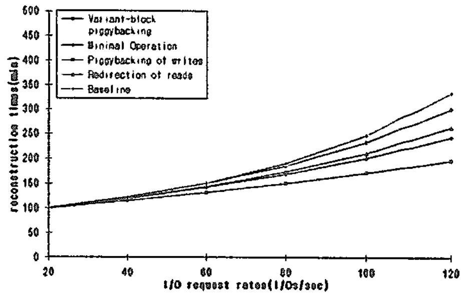

Figure 18: Reconstruction time (reconstruction thread 8)

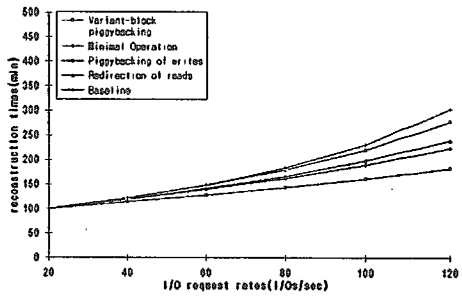

Figure 19: Reconstruction time (reconstruction thread 16)

1 and 4. However, there is no change in the order of reconstruction algorithms.

Figure 15 shows average response time when the reconstruction thread is 16. The average response times are longer than those of reconstruction thread 1, 4, and 8.

Figure 16 shows reconstruction times of reconstruction algorithms when reconstruction thread number is 1. The baseline strategy shows the longest reconstruction time. Variant-block piggybacking shows the shortest reconstruction time by reducing the seek time and rotational delay in reconstruction process. Redirection of reads shows shorter reconstruction time than piggybacking of writes since redirection of reads is more effective and read request of all service holds 70 percent.

Figure 17 shows reconstruction times when reconstruction thread is 4. Reconstruction times of reconstruction algorithms are dramatically reduced compared with the case of reconstruction thread 1. This figure shows that by increasing the reconstruction thread number, reconstruction time can be reduced and the reliability of disk array can be improved.

Figure 18 shows reconstruction times when reconstruction thread is 8. It shows more reduced reconstruction time than those of reconstruction thread 1 and 4.

Figure 19 shows reconstruction times when reconstruction thread is 16. Under reconstruction thread 16, since disk utilization arrives at peak point and thus the more reduction of reconstruction time may not be expected, reconstruction thread 16 does not show large reliability improvement compared with reconstruction thread 8.

Through the increase of reconstruction thread number, overall performance of reconstruction algorithms degrades a little. However, variant-block piggybacking is more reliable than other algorithms and through the increase of reconstruction thread number, reconstruction times of all reconstruction algorithms are reduced. These results shows that though there is a little performance degradation, we get dramatically improved reliability through variant-block piggybacking and multiple reconstruction thread.

6 Conclusion and Future Works

In this paper, we propose a new reconstruction algorithm, called variant-block piggybacking. With continuous block accesses, variant-block piggybacking reduces the reconstruction time and thus improves the reliability of RAID level 5 using parity sparing. The simulation results show that in reconstruction mode, variant-block piggybacking has the shortest reconstruction time compared with the previous algorithms. In multiple points of reconstruction, the proposed algorithm shows the same results. As a future work, we will improve the proposed algorithm, research into the reconstruction algorithm in detail, and propose more generalized reconstruction algorithm.

References

[1] D. A. Patterson, G. A. Gibson, and R. H. Katz,

" A Case for Redundant Arrays of Inexpensive Disks(RAID)," *International Conference on Management of Data(SIGMOD)*, pp. 109–116,1988.

[2] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, "RAID : High-Performance, Reliable Secondary Storage," *ACM Computing Survey*, Vol. 26, No. 2, pp. 145–185,1994.

[3] G. A. Gibson, *Redundant Disk Arrays : reliable, parallel secondary storage*, MIT Press, 1992.

[4] J. Menon, D. Mattson, and S. Ng, *Distributed of Sparing for Improved Performance of Disk Arrays*, Technical Report RJ 7943, IBM, 1991.

[5] J. Menon and D. Mattson, "Comparing of Sparing Alternatives for Disk Arrays," *Proceedings of the International Conference of Computer Architectures*, pp. 318–329, 1992.

[6] A. L. Narasimha Reddy, J. Chandy and P. Banerjee, "Design and Evaluation of Gracefully Degradable Disk Arrays," *Journal of Parallel and Distributed Computing*, Vol. 17, pp. 28–40, 1993.

[7] J. Chandy and A. L. Narasimha Reddy, "Failure Evaluation of Disk Array Organizations," *Proceedings of the International Conference on Distributed Computing Systems*, pp. 319–326, 1993.

[8] R. R. Muntz and J. Lui, "Performance analysis of disk arrays under failure," *Proceedings of 16th VLDB Conference*, pp. 162–173, 1990.

[9] M. Holland, *On-Line Data Reconstruction In Redundant Disk Arrays*, Ph.D Thesis, Carnegie Mellon University, 1994.

[10] R. Y. Hou, and Y. N. Patt, "Track Piggybacking: An Improved Rebuild Algorithm for RAID5 Disk Arrays," *Proceedings of 1995 International Conference on Parallel Processing*, pp. I-136–I-145, 1995.

[11] E. K. Lee, *Performance Modeling and Analysis of Disk Arrays*, Ph.D Thesis, U.C at Berkeley, 1993.

[12] M. H. MacDougall, *Simulating Computer Systems*, MIT Press, 1987.