Comparing Rebuild Algorithms for Mirrored and RAID5 Disk Arrays

Robert Y. Hou, Yale N. Patt
Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor 48109-2122

Abstract

Several disk array architectures have been proposed to provide high throughput for transaction processing applications. When a single disk in a redundant array fails, the array continues to operate, albeit in a degraded mode with a corresponding reduction in performance. In addition, the lost data must be rebuilt to a spare disk in a timely manner to reduce the probability of permanent data loss. Several researchers have proposed and examined algorithms for rebuilding the failed disk in a disk array with parity.

We examine the use of these algorithms to rebuild a mirrored disk array and compare the rebuild time and performance of the RAID5 and mirrored arrays. Redirection of Reads provides comparable average response times and better rebuild times than Piggybacking for a mirrored array, whereas these two algorithms perform similarly for a RAID5 array. In our experiments comparing the two architectures, a mirrored array has more disks than a RAID5 array and can sustain 150% more I/Os per second during the rebuild process. Even if the size of the RAID5 array is increased to match the mirrored array, the mirrored array reduces response times by up to 60% and rebuild times by up to 45%.

1 Introduction

Processor and memory speeds have rapidly increased over the last few years. The use of multiple processors has further improved computer system performance. The net result is a huge potential increase in computer system performance. One reason computer systems have failed to reach their maximum potential is the inability of the I/O subsystem, and in particular the disk subsystem, to keep up with advances in processor and memory technologies.

Transaction processing represents one application environment whose performance is strongly dependent on the disk subsystem. Transaction processing applications require disk subsystems with both high performance and high availability. High performance is generally achieved by using many disk drives to concurrently service independent disk requests. As the number of drives increases, some form of redundancy should be incorporated to allow the disk subsystem to continue servicing I/O requests even when a disk has failed and is being replaced and rebuilt. In this manner, high availability is assured.

This paper re-examines two hardware redundancy schemes, the mirrored disk array and RAID5 disk array, and compares the effectiveness of three algorithms for rebuilding a failed disk. First, it evaluates the performance of each array when a single disk has failed. Second, it assesses the time required to rebuild the failed disk. Average response times for Redirection of Reads and Piggybacking are comparable for the mirrored array. Rebuilding a mirrored array using Piggybacking, however, often requires more time than when using Redirection of Reads. These two algorithms perform similarly for the RAID5 array as concluded by Holland and Gibson [Holl92]. Comparing the performance of the two architectures, the mirrored disk array provides lower response times during rebuild than the RAID5 disk array, and it rebuilds the failed disk faster.

The remainder of this paper is organized in seven sections. Section 2 discusses previous work in the field. Section 3 describes our approach to disk rebuild and

SIGMOD /5/93/Washington, DC,USA
© 1993 ACM 0-89791-592-5/93/0005/0317...$1.50

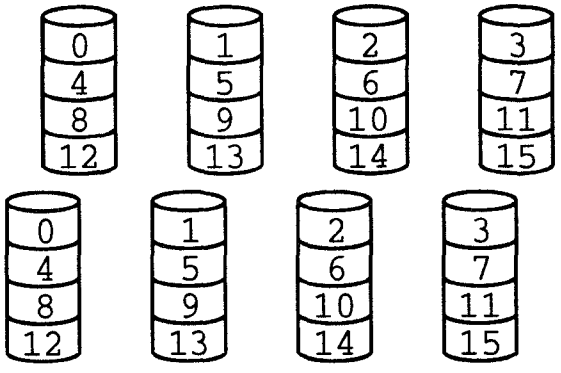

Figure 1: Mirrored disk array

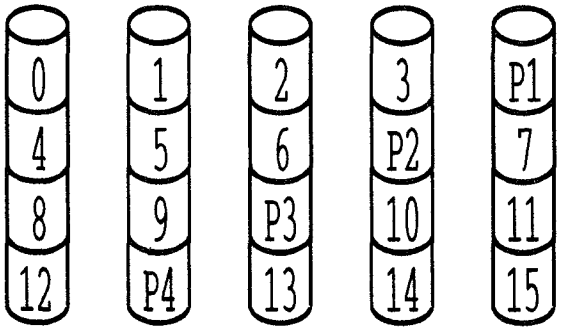

Figure 2: RAID5 disk array

lists our assumptions. Section 4 discusses our simulator and the workload we evaluated. Section 5 examines the response time during rebuild mode and section 6 analyzes the rebuild time incurred by the different rebuild algorithms. Section 7 presents our conclusions and some topics for future research.

2 Previous Work

Increasing the number of concurrently active disk drives in the subsystem is a common technique for improving disk subsystem performance. Striping (aka interleaving) data across disks can greatly increase the data transfer rate [Sale86, Kim86]. A stripe unit is the amount of data placed on each disk such that successive stripe units contain logically contiguous data. Disk striping allows the data for large requests to be accessed in parallel from multiple disks. Disk striping with a large stripe unit can also be used to form one logical disk capable of performing many concurrent accesses [Livn87]. A stripe unit is chosen to balance data transfer parallelism and access concurrency [Redd89, Chen90a]. Another important effect of disk striping is it balances the load among all the disks in the array [Livn87, Bate91, Gang93].

Unfortunately, increasing the number of disks also increases the probability of data loss due to disk failure. Several hardware redundancy schemes have been used to improve reliability, including mirroring [Bitt88] and disk arrays with parity [Kim86, Patt88, Katz89, Gray90]. Mirrored disks duplicate each data block, placing the two copies on different disks. As a result, two disks can potentially service a given read request. If both disks are available, the disk with the shorter seek time can be selected to service the request. Write requests, on the other hand, must be serviced by both disks. This paper assumes that data is striped across each set of disks and that seek time is used to select which disk will service a read request if both are available. Figure 1 illustrates how data can be arranged in a mirrored disk array.

The RAID5 disk array has become popular recently since it requires fewer disks than a mirrored disk array while ensuring data availability in the event of a single disk failure. Figure 2 illustrates how data can be organized in a RAID5 disk array. Redundancy is maintained in the form of parity. Write accesses must update both the data blocks being written and the corresponding parity blocks. In particular, single-block writes require the old data and parity to be read, exclusive-ORed with the new data to generate the new parity, and then the new data and parity can be written to their respective locations. This is called a *read-modify-write* operation. Read-modify-writes can reduce the performance of RAID5 arrays when compared to other architectures.

A RAID5 disk array can operate in three different modes [Meno92]. It operates in normal mode when all disks are available. It moves to degraded mode when one of the disks in a disk array has failed. When data on the failed disk needs to be accessed, corresponding blocks from all surviving disks must be read and exclusive-ORed to regenerate the lost data. Response times in degraded mode are therefore much higher than in normal mode.

A RAID5 array in degraded mode also cannot survive a second disk failure. Thus it is important to rebuild the failed disk as quickly as possible. Nevertheless, it is usually important to continue servicing the application's disk requests as well. Much of the research in the rebuild mode performance of disk arrays assumes the application can afford some window

of vulnerability and that the application prefers its disk requests be serviced, albeit with a longer response time. The alternative is to have the disk array postpone all application requests until the data on the failed disk has been rebuilt to a spare disk. Menon and Mattson [Meno92] analyzed response times and rebuild times for a RAID5 disk array assuming user requests are given priority over rebuild requests.

Several researchers have proposed and investigated solutions for more efficiently rebuilding the failed disk in *clustered* RAID arrays[1] [Munt90], of which the RAID5 array is a special case. Muntz and Lui [Munt90] compared three algorithms for rebuilding clustered RAID arrays using an analytical model. Their *Baseline Copy* algorithm simply reads blocks sequentially from each of the surviving disks, regenerates the lost data and writes it to the spare disk. Read requests to the failed disk are regenerated by the surviving disks. Write requests update the parity block on the appropriate surviving disk. In addition, these write requests also update the spare disk.

Their first enhancement on Baseline Copy, *Redirection of Reads*, uses the spare disk to service read requests to the failed disk if the requested data has already been rebuilt to the spare disk. They also proposed *Piggybacking* rebuild requests on a normal workload as an additional improvement. In Piggybacking, any disk block regenerated by an application request to the failed disk is written to the spare disk. As a result, this data block does not have to be regenerated at some later point in the rebuild process. Muntz assumed that the rebuild time was reduced by a fraction proportional to the number of blocks rebuilt via Piggybacking.

Holland and Gibson [Holl92] extended Muntz and Lui's work by performing simulations of the rebuild algorithms and comparing response times and rebuild times. They also evaluated a simpler rebuild algorithm, *Minimal Update*, which simply reads blocks sequentially from each of the surviving disks, regenerates the missing data and writes it to the spare disk. Unlike baseline copy, however, the spare disk only services writes which access data that has been rebuilt to the spare disk. Holland and Gibson were able to more precisely determine the benefits of Redirection of Reads and Piggybacking . They found these "improved" rebuild algorithms were not always beneficial, especially when the surviving disks are not saturated with application requests. In these cases, the sequential rebuild requests serviced by the spare disk are interrupted with random reads and writes from the application. As a result, the rebuild time can actually increase.

Another decision impacting the rebuild time is the amount of data atomically read each time from surviving disks during rebuild, which we called the *rebuild unit* [Hou93]. We compared rebuild performance for Minimal Update using three different rebuild units — block, track and cylinder. Rebuilding data one block at a time, where a block is the amount of data accessed by a disk request and is smaller than a track, ensures a small delay for incoming application requests and minimally raises their response times. The drawback is an increase in the disk array rebuild time. A cylinder rebuild unit, on the other hand, decreases the rebuild time while increasing response times. A track rebuild unit provides a favorable balance between application response time and rebuild time. Thus the rebuild unit balances the servicing of application and rebuild requests and can be different from the stripe unit, used only to optimize application requests.

Much of the work on rebuilding RAID5 arrays can be extended to mirrored arrays. In particular, the algorithms proposed by Muntz and Lui can be used to rebuild a mirrored array. We evaluate the effectiveness of Minimal Update, Redirection of Reads and Piggybacking in rebuilding a mirrored array.

3 Basic Approach

When a disk in an array fails, it is imperative to reconstruct the data on that disk to a spare or replacement disk as quickly as possible to avoid data loss. It is also necessary, however, to continue servicing application requests. We assume it is more important to service application requests than rebuild requests [Meno92]. Therefore we assign a lower priority to rebuild requests, servicing them only if there are no pending application requests for that disk. Once a rebuild request has been sent to a disk, the disk completes that request and then checks if there are any pending application requests. If not, another rebuild request may be serviced.

Unlike Holland and Gibson [Holl92], who assumed the operating system would rebuild the array, we assume there is a hardware mechanism in the disk array controller that rebuilds the array. This hardware mechanism has buffers which hold data during the rebuild process.

We evaluate response times and rebuild times for a mirrored disk array and compare the mirrored array with the RAID5 array. We start with a nine-disk

[1] also called *declustered parity* [Holl92]

bytes per block	4096
blocks per track	6
surfaces	14
cylinders	949
single track seek	2.5 ms
average seek time	12.7 ms
maximum seek time	25.5 ms
rotational speed	4318 RPM
seek time model	2.0 + 0.01 distance + 0.46 $\sqrt{distance}$

Table 1: Disk Drive Parameters

RAID5 array. This array size was chosen to ensure that parity consumed a small amount of disk space relative to the data; parity accounts for only 11% of the total disk space in a nine-disk array. We compare the RAID5 performance with a sixteen-disk mirrored array since both arrays contain the same amount of data. The mirrored array, however, has superior performance to the small RAID5 array. Thus we also evaluate a sixteen-disk RAID5 array.

4 Simulator

Trace-driven simulation is used to evaluate the response time and rebuild time of the mirrored and RAID5 disk arrays. The disk drive parameters modeled by the simulator are shown in Table 1. We model the same IBM 0661 Model 370 disk drive as Holland and Gibson [Holl92].

The workload is a synthetically generated stream of disk requests. The requests are uniformly distributed across the disk space since typical transaction processing applications exhibit little or no locality in their I/O streams. Each request accesses a single 4KB block of data. The interarrival time, or time between successive requests, is generated using an exponential distribution. 75% of the disk requests are assumed to be reads, again modeling a typical transaction processing environment [Meno92].

The I/O rate is varied by changing the interarrival time to create different workloads for the disk subsystems. When the I/O rate to a sixteen-disk array is 175 I/Os per second, the disk array is lightly utilized. When the I/O rate is increased to 300 I/Os per second, the sixteen-disk arrays approach saturation in degraded mode. For the nine-disk RAID5 array, the low and high I/O rates are 75 and 125 I/Os, respectively. Average response time is used to compare the performance of the disk arrays under these workloads.

Each disk drive stores rebuild data in its own buffer in the storage controller. The experiments in section 6 will evaluate the amount of buffer space needed in the storage controller to provide the optimal performance.

When enough rebuild data has been read from each surviving disk, the storage controller regenerates the lost data and writes it to the spare disk. Data may be written to the spare disk at a different granularity than it is read from the surviving disks. In the experiments presented in this paper, the amount of data written each time to the spare disk is fixed at one track [Meno92, Hou93]. Writing data one track at a time can take advantage of zero-latency disk accesses. It also amortizes any seek time over several reconstructed data blocks. Writing data one cylinder at a time also has these benefits. The drawback is the spare disk becomes unavailable for long periods of time and cannot service application requests that access rebuilt data. Also, the storage controller must store a full cylinder of rebuild data before beginning to write reconstructed data to the spare disk.

5 Response Time

It is important to maintain low response times for application requests during the rebuild process. To minimize the conflict between application and rebuild requests, application requests are assigned a higher priority [Meno92]. Response time does increase during rebuild mode, however, and the increase depends mainly on the rebuild unit [Hou93]. We have analyzed the impact of the rebuild unit on RAID5 rebuild [Hou93]. In this section we will analyze its impact on mirrored rebuild and compare it to RAID5 rebuild. The response times reported in this section assume there is enough buffer space in the array controller to store one cylinder of data from each disk. Increasing buffer space to ten cylinders of data gives similar results.

Response times for a mirrored disk array are examined first. Times for the Minimal Update, Redirection of Reads and Piggybacking rebuild algorithms using a track rebuild unit are shown in figure 3 as a function of I/Os per second (workload). Degraded mode performance is provided for comparison purposes. When a disk in a mirrored array fails, its duplicate disk must service twice as many reads as in normal mode. Thus the response time for the du-

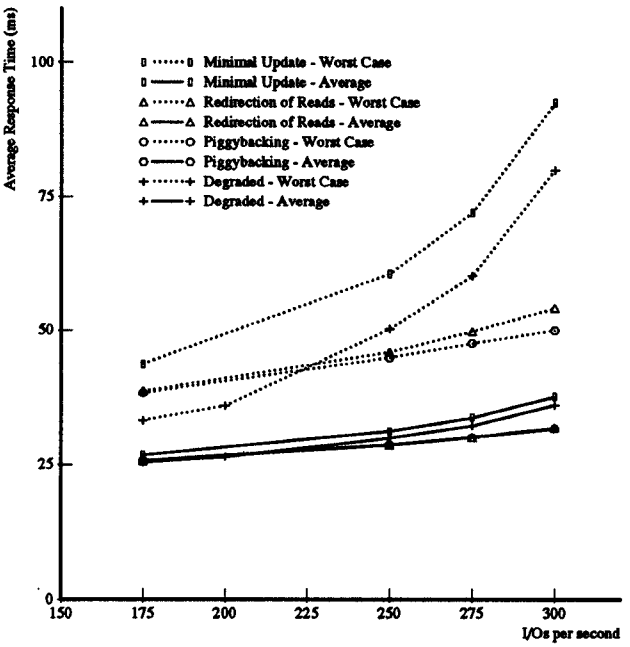

Figure 3: Response times for mirrored array using track rebuild unit

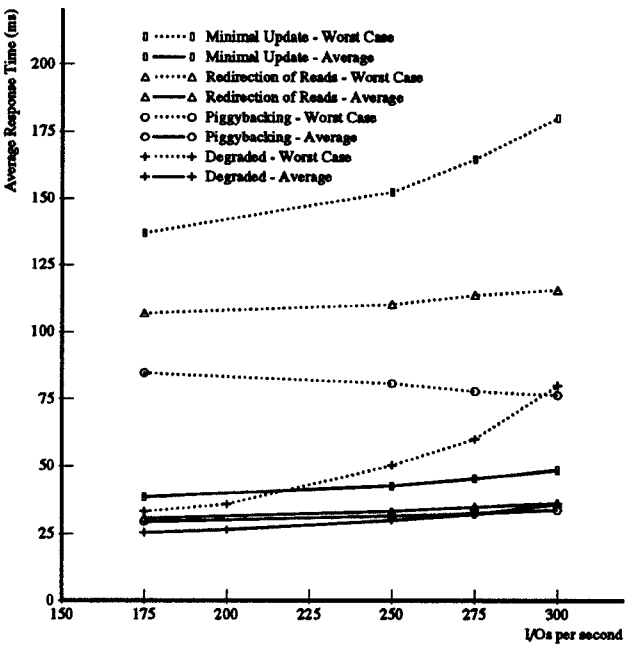

Figure 4: Response times for mirrored array using cylinder rebuild unit

plicate disk is higher than for the other disks. We will refer to this duplicate disk as the *partner disk* in the remainder of this paper and call its response time the *Worst Case* response time for the array. The *Average* response time is the average over all requests. For Minimal update, Worst Case response times are 63% - 146% higher than Average response times, while for Redirection of Reads and Piggybacking, they are 49% - 69% higher. Comparing Redirection of Reads and Minimal Update, Redirection of Reads reduces Worst Case response times by 12% - 42% and Average response times by 4% - 15%. Piggybacking provides little improvement over Redirection of Reads, which confirms Holland and Gibson's results [Holl92].

Another observation is that Redirection of Reads and Piggybacking perform better than Degraded mode at higher I/O rates [Munt90]. The reason is that many requests can be serviced by the spare disk after the desired data has been rebuilt. For these requests, response time is closer to normal mode. It is not exactly equal to normal mode since the spare disk is still writing reconstructed data.

Figure 4 shows response times for Minimal Update and Redirection of Reads using a cylinder rebuild unit. Again, Degraded mode performance is provided for comparison purposes. As can be seen, a cylinder rebuild unit increases the response times for Worst Case dramatically. In fact, it increases response times for Redirection of Reads enough that even with application requests being redirected to the spare disk, the average response times are still greater than in Degraded mode.

Comparing figure 4 to figure 3, Redirection of Reads response times degrade more gracefully with heavier workloads and Piggybacking response times actually improve. Any requests that are redirected and serviced by the spare disk do not suffer the large response time increases that result when a surviving disk is reading a cylinder rebuild unit of data. Piggybacking improves response times since a heavier workload means more data is rebuilt to the spare disk via Piggybacking, which increases the chances that future requests will be redirected to the spare disk. In addition, an increasing number of cylinders are partially rebuilt to the spare disk, in which case only part of a cylinder needs to be rebuilt.

Results for a block rebuild unit are similar to those for a track rebuild unit and are not shown. A block rebuild unit gives response times that are roughly half a rotational latency lower compared to a track rebuild unit. The time required to read a random block rebuild unit from a surviving disk is about a half rota-

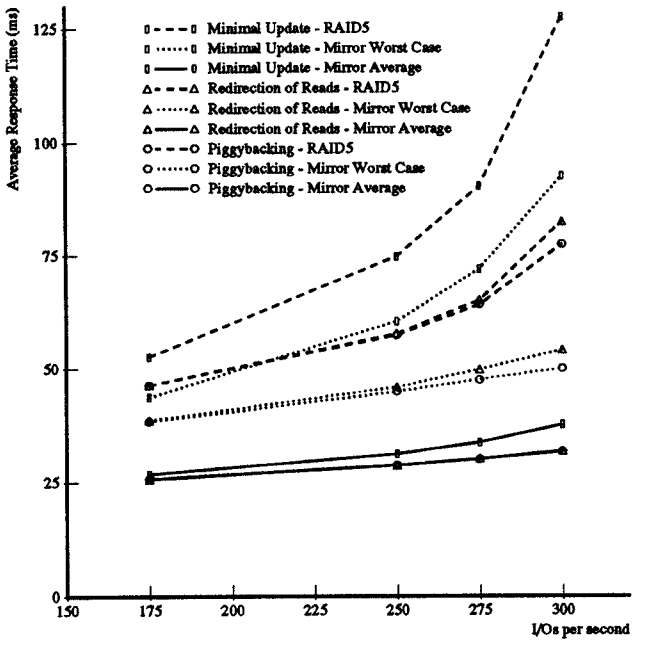

Figure 5: Response times for RAID5 and mirror using track rebuild unit

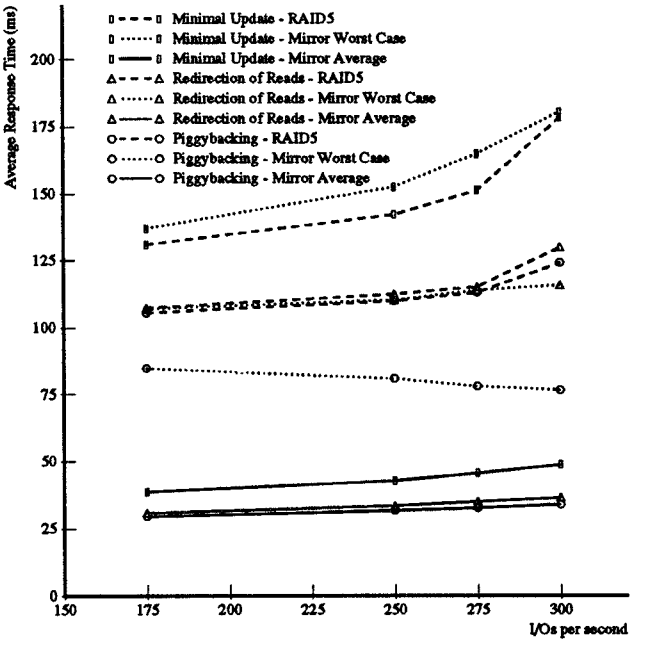

Figure 6: Response times for RAID5 and mirror using cylinder rebuild unit

tional latency shorter than reading a full track rebuild unit.

Response times comparing mirrored and RAID5 arrays for a track rebuild unit are shown in figure 5 comparing Minimal Update, Redirection of Reads and Piggybacking. We compare response times using a sixteen-disk RAID5 array since a nine-disk RAID5 array cannot operate under a workload over 175 I/Os per second. Response times for RAID5 are higher than those for mirrored (both Worst Case and Average), even though seek times for RAID5 have been reduced[2]. In normal mode, RAID5 and mirrored perform similarly for reads but RAID5 suffers when handling writes due to the read-modify-write operation [Chen90]. In degraded and rebuild modes, all surviving disks in the RAID5 array receive twice the number of application read requests due to the reconstruction of the missing data [Meno92]. In the mirrored array, the partner disk similarly sees twice the number of application read requests. Write requests, on the other hand, affect only the degraded RAID5 array, having virtually no effect on the mirrored array. Write requests to the failed disk in a RAID5 array must construct the new parity block, which requires reading the corresponding data blocks on all the surviving disks except the one containing the parity block. RAID5 response times are 20% - 38% higher than Worst Case mirrored times for Minimal Update and 18% - 55% for Redirection of Reads and Piggybacking. Compared to Average mirrored response times, RAID5 response times are 97% - 238% higher for Minimal Update and 80% - 159% higher for Redirection of Reads and Piggybacking.

Figure 6 compares response times for a cylinder rebuild unit. Worst Case mirrored response times for Minimal Update are higher than those for RAID5. This is again due to the limited buffer space of one cylinder in the controller. For a RAID5 array, the disks work in lockstep to rebuild the spare disk and are often idle since the fastest disk must wait for the slowest disk to read its cylinder of rebuild data before the spare disk can write the regenerated data. Since the disks are often idle, their response times for application requests is reduced. In a mirrored array, on the other hand, only the partner disk is rebuilding the spare disk. The partner disk only has to wait for the spare disk to write its data, so it is generally kept busy. The two curves converge at the highest workload since the surviving disks are nearly saturated with application requests and the rebuild process slows down. If

[2]The amount of data in the two disk arrays are kept constant and thus the RAID5 disks are not completely filled with data

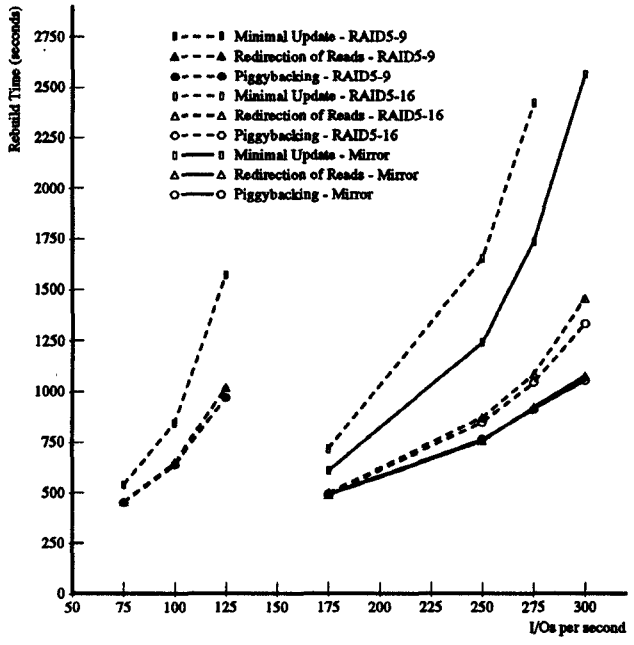

Figure 7: Rebuild time as a function of I/Os using block rebuild unit

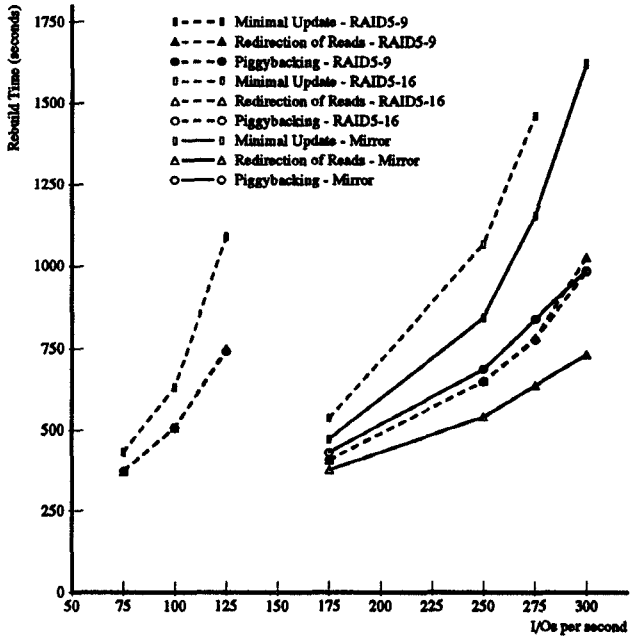

Figure 8: Rebuild time as a function of I/Os using track rebuild unit

the buffer space in the storage controller is increased to ten cylinders, then RAID5 response times are higher than Worst Case mirrored response times.

Both the block and track rebuild units give response times that are close to degraded mode response times. A cylinder rebuild unit, on the other hand, significantly increases response times. Since the response times for a cylinder rebuild unit are much higher, cylinder rebuild will be ignored in the remainder of this paper.

6 Rebuild Time

It is important to rebuild the data on a failed disk as quickly as possible to minimize the possibility of data loss. This section will evaluate rebuild time for mirrored and RAID5 disk arrays considering the size of the buffer space in the disk array controller as well as the rebuild unit. Results for both a nine-disk and sixteen-disk RAID5 array are provided. The nine-disk RAID5 array stores the same amount of data as the sixteen-disk mirrored array and thus provides a comparison of equal capacity disk arrays. The sixteen-disk arrays provide a comparison of equal cost disk arrays.

Rebuild time is shown as a function of I/Os per second (workload) serviced by the disk array in figures 7 and 8. The rebuild unit is a block in figure 7 and a track in figure 8. There is enough space in the disk controller to store ten cylinders of rebuild data from each disk. As can be seen, the nine-disk RAID5 array cannot operate under heavy workloads. This is expected since it has fewer disks to service application requests. In the remainder of this section, we will only discuss the mirrored array and the sixteen-disk RAID5 array although we will include the results for the nine-disk RAID5 array for completeness.

In general, the mirrored array out performs the RAID5 array because it depends only on the partner disk. The disks in the RAID5 array can slip with respect to each other as each becomes idle at different times. The amount of slip is limited by the available buffer space in the array controller. Thus the disks have increased idle time in the RAID5 array. The mirrored array reduces rebuild time relative to the sixteen-disk RAID5 array by 15% - 28% for Minimal Update and 0% - 26% for Redirection of Reads and Piggybacking.

Piggybacking performs marginally better than Redirection of Reads. It is beneficial only if it saves the need to write reconstructed data to the spare disk, thereby reducing rebuild time. As pointed out by Holland and Gibson [Holl92], it is unlikely that an entire

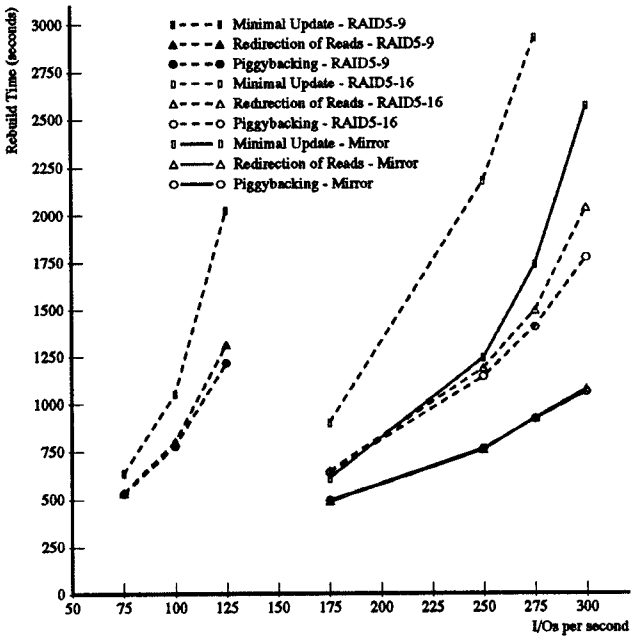

Figure 9: Rebuild time as a function of I/Os using block rebuild unit

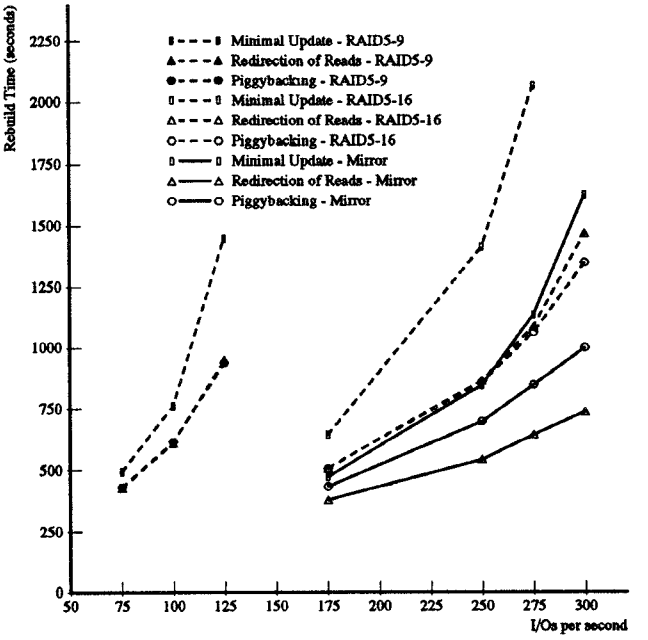

Figure 10: Rebuild time as a function of I/Os using track rebuild unit

track will be rebuilt by Piggybacking. If part of a track still needs to be rebuilt, then at least one seek and latency is still incurred by each disk participating in the rebuild process. Thus there is little benefit to using Piggybacking, especially at low I/O rates. At high I/O rates, the surviving disks are the bottleneck since they have many application requests that they also need to service. In this case, Piggybacking does rebuild some blocks that do not have to be rebuilt later. As expected, the sixteen-disk and nine-disk RAID5 arrays benefit similarly from the various rebuild algorithms.

Figure 8 shows similar results when the rebuild unit is a track. As the rebuild unit gets larger, the rebuild time decreases since seek time is amortized over more rebuild data and rotational latency is essentially eliminated. The mirrored array reduces rebuild times relative to the sixteen-disk RAID5 array by 12% - 21% for Minimal Update and 7% - 29% for Redirection of Reads.

It is interesting to note that Piggybacking performs much worse than Redirection of Reads for mirrored disks. Piggybacking is triggered on every read request to the failed disk that cannot be serviced by the spare disk. For mirrored disks, one-eighth of all read requests address blocks on the failed disk while the fraction is one-sixteenth for the RAID5 array. Yet even though Piggybacking rebuilds more data to the spare disk, it is still unlikely these requests will rebuild entire tracks of data to the spare disk. Since the rebuild unit is a track, the partner disk must still rebuild any block on a given track not rebuilt via Piggybacking. This is not the case for a block rebuild unit since any block rebuilt via Piggybacking does not have to be rebuilt later by the partner disk. As a result, mirrored rebuild time is greater than RAID5 rebuild by up to 5%.

Figures 9 and 10 show rebuild times when the array controller can store one buffer of rebuild data from each disk. The limited buffer space causes the rebuild times to increase. RAID5 rebuild time is impacted more than mirrored rebuild time since RAID5 rebuild depends on the ability of the different disks to slip with respect each other during the rebuild process. The impact on mirrored rebuild times is less than 1% for a block rebuild unit and about 1% for a track rebuild unit. RAID5 rebuild times, on the other hand, are increased by 25% - 40% for a block rebuild unit and 20% - 43% for a track rebuild unit. Compared to RAID5 rebuild times, mirrored rebuild times are 25% - 45% lower for a track rebuild unit.

It may be argued that a sixteen-disk RAID5 does

not have to rebuild the entire failed disk to the spare disk since almost half the failed disk is not being used. This requires the entity responsible for the rebuild process, whether it is an array controller or the operating system, be aware of where the valid data on the failed disk is stored. The array controller is unlikely to keep this information. The operating system may or may not keep this information. Our current assumption is that the entire failed disk should be rebuilt to the spare disk. Naturally, this assumption bears further investigation.

7 Conclusions

We have examined response times and rebuild times for mirrored and RAID5 disk arrays. By assigning a lower priority to rebuild requests, we are able to minimize the impact of the rebuild process on response times, provided the rebuild unit is either a block or track. We have determined that a mirrored array provides substantially better average and worst case response times when compared to a RAID5 array. We found that a sixteen-disk mirrored array reduces average response times by 45% - 60% for a track rebuild unit. Mirrored worst case response times were 17% - 35% lower than average RAID5 response times. We did not report response time comparisons between the mirrored array and the nine-disk RAID5 array simply because the smaller RAID5 array saturates long before the mirrored array reaches reasonable levels of utilization.

Rebuild times are also lower for mirrored arrays. Similar to RAID5 arrays [Holl92], Redirection of Reads was determined to be the best rebuild algorithm for mirrored arrays. Unlike the RAID5 arrays, Piggybacking performs considerably worse than Redirection of Reads. Mirrored arrays provide the same rebuild times as smaller RAID5 arrays while sustaining much heavier workloads. We found that a sixteen-disk mirrored array can support roughly 150% more I/Os per second during rebuild mode than a nine-disk RAID5 array with equal storage capacity. Comparing the best rebuild algorithms for the sixteen-disk mirrored and RAID5 arrays, the mirrored array reduced rebuild times by 25% - 45% for a track rebuild unit with one cylinder of buffer space per disk in the array controller.

Buffer space impacts RAID5 rebuild significantly. Mirrored rebuild, on the other hand, is almost completely insensitive to buffer size variation. Efficient RAID5 rebuild depends on the ability of the individual disks to "slip" with respect to each other during the rebuild process. This is not important for mirrored rebuild since only one disk is involved.

The mirrored array provides better response times and rebuild times than RAID5 arrays during rebuild mode. The main advantage of RAID5 is the lower cost. As disk arrays become commodity items, mirrored arrays may become a cost-effective solution to a wider range of applications.

It should be noted that the advantages offered by the mirrored array during the rebuild process should not depend on the entity responsible for the rebuild process. We have assumed the storage controller is responsible for rebuilding the failed disk. We do not expect the results to change if the operating system is made responsible [Holl92].

More work remains to be done to evaluate response times and rebuild times for mirrored disk arrays. The worst case mirrored response times indicate the need for better data layout policies. Indeed, several researchers [Tera85, Hsia90] have proposed different layout policies such as interleaved declustering and chained interleaving to reduce response times in degraded mode. It remains to be seen how these strategies affect the rebuild time. In particular, interleaved declustering can greatly reduce rebuild time, since all surviving disks are involved in the rebuild process instead of just one as in the traditional mirrored array.

Acknowledgements

This work is part of a larger research project in I/O being carried out at the University of Michigan, funded by NCR Corporation – E&M Columbia. We gratefully acknowledge NCR's support. We also acknowledge the support and use of resources at IBM Almaden Research Center. The work reported here is partly based on research performed at IBM during July and August, 1992. We also wish to thank David Jaffe of MTI, Jai Menon of IBM, and the members of our research group at Michigan, particularly Greg Ganger and Bruce Worthington, for all the technical discussions we have had on the various I/O issues.

Finally, our research group is very fortunate to have the financial and technical support of several industrial partners. We are pleased to acknowledge them. They include NCR, Intel, Motorola, Scientific and Engineering Software, HaL, Hewlett-Packard, Micro Technology Incorporated and DEC.

References

[Bate91] K. Bates, *VAX I/O Subsystems: Optimizing Performance*, Professional Press Books, Horsham, Pennsylvania, 1991.

[Bitt88] D. Bitton, J. Gray, "Disk Shadowing", *Proceedings of the Very Large Databases Conference*, September 1988, pp. 331-338.

[Chen90] P.M. Chen, G.A. Gibson, R.H. Katz, D.A. Patterson, "An Evaluation of Redundant Arrays of Disks using an Amdahl 5890", *ACM Sigmetrics*, 1990, pp. 74-85.

[Chen90a] P. Chen, D. Patterson, "Maximizing Performance in a Striped Disk Array", *Proceedings of the 17th International Symposium on Computer Architecture*, 1990, pp. 322-331.

[Gang93] G. Ganger, B. Worthington, R. Hou, Y. Patt, "Disk Subsystem Load Balancing: Disk Striping vs. Conventional Data Placement", *Proceedings of the Hawaii International Conference on System Sciences*, 1993, pp. 40-49.

[Gray90] J. Gray, B. Horst, M. Walker, "Parity Striping of Disk Arrays: Low-Cost Reliable Storage with Acceptable Throughput", *Proceedings of the 16th VLDB Conference*, August 1990, pp. 148-161.

[Hou93] R. Hou, J. Menon, Y. Patt, "Balancing I/O Response Time and Disk Rebuild Time in a RAID5 Disk Array", *Proceedings of the Hawaii International Conference on System Sciences*, 1993, pp. 70-79.

[Holl92] M. Holland, G. Gibson, "Parity Declustering for Continuous Operation in Redundant Disk Arrays", *Architectural Support for Programming Languages and Operating Systems*, 1992, pp. 23-35.

[Hsia90] H. Hsiao, D. J. DeWitt, "Chained Declustering: A New Availability Strategy for Multiprocessor Database Machines", *International Conference on Data Engineering*, 1990, pp. 456-465.

[Katz89] R.H. Katz, G.A. Gibson, D.A. Patterson, "Disk System Architectures for High Performance Computing", *Proceedings of the IEEE*, December 1989, pp. 1842-1858.

[Kim86] M. Kim, "Synchronized Disk Interleaving", *IEEE Transactions on Computers*, November 1986, pp. 978-988.

[Livn87] M. Livny, S. Khoshafian, H. Boral, "Multi-Disk Management Algorithms", *SIGMETRICS*, 1987, pp. 69-77.

[Meno92] J. Menon, D. Mattson, "Performance of Disk Arrays in Transaction Processing Environments", *12th International Conference on Distributed Computing Systems*, 1992, pp. 302-309.

[Munt90] R. Muntz, J. Lui, "Performance Analysis of Disk Arrays Under Failure", *Proceedings of the Very Large Databases Conference*, 1990, pp. 162-173.

[Patt88] D. Patterson, G. Gibson, R. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)", *ACM SIGMOD*, May 1988, pp. 109-116.

[Redd89] A.L.N. Reddy, P. Banerjee, "An Evaluation of Multiple-Disk I/O Systems", *IEEE Transactions on Computers*, December 1989, pp. 1680-1690.

[Sale86] K. Salem, G. Garcia-Molina, "Disk Striping", *International Conference on Data Engineering*, 1986, pp. 336-342.

[Tera85] Teradata Corporation. "DBC/1012 Database Computer System Manual Release 2.0", Document No. C10-0001-02, November, 1985.

[TPC90] Transaction Processing Performance Council. "TPC Benchmark B - Standard Specification". Waterside Associates, Fremont, California. August 23, 1990.