Failure Evaluation of Disk Array Organizations

John Chandy
University of Illinois
1101 W. Springfield Ave.
Urbana, IL 61801
jchady@crhc.uiuc.edu

A. L. Narasimha Reddy
IBM Almaden Research Center
650 Harry Road, K56/803
San Jose, CA 95120.
reddy@almaden.ibm.com

Abstract

In this paper, we present an evaluation of some of the disk array organizations proposed in the literature. We evaluate three alternatives for sparing, hot sparing, distributed sparing and parity sparing, and two options for data layout, regular RAID5 and block designs, and systems based on combinations of these data layout and sparing alternatives. We evaluate the performance of these organizations in various modes of operation, with different reconstruction strategies. It is shown that parity sparing and distributed sparing have improved performance and shorter reconstruction times over hot sparing. It is shown that both block designs as a data layout policy and distributed sparing as a sparing policy reduce the reconstruction time after a failure. We also study the impact of reconstruction strategies and show that at higher workloads, choice of reconstruction strategy has a significant impact on the performance of the systems.

1 Introduction

With the increasing processor speeds and multiprocessor organizations, building a balanced computer requires that suitable improvements in the I/O system's performance be made. Several studies for organizing multiple disks in an I/O system have been reported [1, 2, 3, 4]. In disk arrays [3], multiple smaller disks replace traditional large disks. To achieve the same reliability of a single large disk, the data on the smaller disks is protected by parity on another disk. Disk arrays have been receiving increased attention for providing a highly-available disk system. By appropriately striping data across the disks in a disk array, high performance can be achieved.

In a disk array, a group of disks, *parity group*, are protected by parity. When a disk fails, the data on the functional disks is XOR'd to obtain the data on the failed disk. The number of disks in a parity group is called parity group length. When multiple parity groups are present in the disk array, the number of disks in the array, *array width*, will be different from the parity group length. In disk arrays, spare space is provided for reconstructing the failed data during the reconstruction process. This spare space can be organized in several ways. We consider three alternatives proposed in the literature: hot sparing [3], distributed sparing [5] and parity sparing [6].

In hot sparing, a hot spare disk is used to recover from failures. Hot sparing has some drawbacks: the failure of a spare disk may go unnoticed since it is not used during normal operation and hence can cause an unrecoverable loss if another disk fails after an undetected failure of the spare disk. In a system with n disks, only $n-1$ disks in the system are utilized during normal operation. This organization is shown in Fig. 1. In Fig. 1, each column corresponds to a disk and each row corresponds to the data layout for a track on the disks, D corresponding to data, P for parity and S for spare. Let the matrix shown in Fig. 1 have t rows. Then for a track address i, row $i \ mod \ t$ gives the data layout for that track. The representation in Fig. 1 may be augmented by subscripts when there are multiple parity groups within a single array.

D_0	D_0	D_0	D_0	D_0	P_0	S		D_0	D_0	D_0	D_0	D_0	P_0
D_0	D_0	D_0	D_0	P_0	D_0	S		D_0	D_0	D_0	D_0	P_0	D_0
D_0	D_0	D_0	P_0	D_0	D_0	S		D_0	D_0	D_0	P_0	D_0	D_0
D_0	D_0	P_0	D_0	D_0	D_0	S		D_0	D_0	P_0	D_0	D_0	D_0
D_0	P_0	D_0	D_0	D_0	D_0	S		D_0	P_0	D_0	D_0	D_0	D_0
P_0	D_0	D_0	D_0	D_0	D_0	S		P_0	D_0	D_0	D_0	D_0	D_0

Fig. 1. Array organization before and after a failure in hot sparing.

Parity sparing [6] proposes an approach where the disk array has no spare disks. The spare disk is used to provide a second parity disk and thus to reduce the parity group length. On a failure, the parities of the two groups are merged to obtain a single larger array with a single parity group with a larger parity group length. This scheme can reduce the parity group lengths by making effective use of the spare disks during normal operation. Parity sparing can be implemented in two ways. The first scheme physically partitions the disks into two smaller arrays as shown in Fig. 2 where the disks are partitioned into two smaller arrays of width 3 and 4. Disks are physically partitioned into two arrays and these two arrays function independently during normal operation. On a failure, the two arrays are merged into a single larger array.

D_0	D_0	P_0	D_1	D_1	D_1	P_1		D_0	D_0	D_1	D_1	D_1	P_{01}
D_0	P_0	D_0	D_1	D_1	P_1	D_1		D_0	D_0	D_1	D_1	P_{01}	D_1
P_0	D_0	D_0	D_1	P_1	D_1	D_1		D_0	D_0	D_1	P_{01}	D_1	D_1
D_0	D_0	P_0	P_1	D_1	D_1	D_1		D_0	D_0	P_{01}	D_1	D_1	D_1
D_0	P_0	D_0	D_1	D_1	D_1	P_1		D_0	D_0	D_1	D_1	D_1	P_{01}
P_0	D_0	D_0	D_1	D_1	P_1	D_1		D_0	D_0	D_1	D_1	P_{01}	D_1
D_0	D_0	P_0	D_1	P_1	D_1	D_1		D_0	D_0	D_1	P_{01}	D_1	D_1
D_0	P_0	D_0	P_1	D_1	D_1	D_1		D_0	D_0	P_{01}	D_1	D_1	D_1
P_0	D_0	D_0	D_1	D_1	D_1	P_1		D_0	P_{01}	D_1	D_1	D_1	D_0
D_0	D_0	P_0	D_1	D_1	P_1	D_1		D_0	P_{01}	D_1	D_1	D_0	D_1
D_0	P_0	D_0	D_1	P_1	D_1	D_1		P_{01}	D_0	D_1	D_0	D_1	D_1
P_0	D_0	D_0	P_1	D_1	D_1	D_1		P_{01}	D_0	D_0	D_1	D_1	D_1

Fig. 2. Array organization before and after a failure in parity sparing approach.

The second scheme partitions the disks into two logical arrays. In this scheme, each disk belongs to two parity groups depending on the block address. For some blocks, a disk is in parity group 0 and for others it is in parity group 1. The membership in the parity groups depends on the block number. The parity groups can be so designed that each disk interacts with any other disk in the same parity group in less than half the number of blocks. This data organization is based on block designs. These arrays limit the load increase due to a failure to less than 50% of the normal system load before the failure. In a RAID5 disk array, if one of the disks fails, all the functional disks in the array participate to reconstruct the data on the failed disk and hence these functional disks observe a load increase of 100% during a failure. The arrays designed based on block designs, by employing two parity groups and by proper data organization limit these increases to less than 50% on all the remaining disks [6]. Limiting this load increase is beneficial for quickly reconstructing the data on the failed disk. The parity sparing approach alone, when block designs are not employed, reduces the parity group length but still experiences a load increase of 100% on all the functional disks in the smaller broken array. A 7-disk system employing block designs and parity sparing is shown in Fig. 3. In Fig. 3, D_0 corresponds to data in parity group 0, P_0 corresponds to parity of group 0, and P_{01} corresponds to the combined parity of groups 0 and 1 etc. For the example

P_1	D_1	D_0	D_1	D_0	P_0	D_0		D_1	D_0	D_1	D_0	P_{01}	D_0
P_0	P_1	D_1	D_0	D_1	D_0	D_0		P_{01}	D_1	D_0	D_1	D_0	D_0
D_0	P_0	P_1	D_1	D_0	D_1	D_0		D_0	P_{01}	D_1	D_0	D_1	D_0
D_0	D_0	P_0	P_1	D_1	D_0	D_1		D_0	D_0	P_{01}	D_1	D_0	D_1
D_1	D_0	D_0	D_0	P_1	D_1	P_0		D_0	D_0	D_0	D_1	D_1	P_{01}
D_0	D_1	D_0	P_0	D_0	P_1	D_1		D_1	D_0	D_0	D_0	P_{01}	D_1
D_1	D_0	D_1	D_0	P_0	D_0	P_1		D_0	D_1	D_0	P_{01}	D_0	D_1

Fig. 3. Array organization before and after a failure in block designs approach.

shown in Fig. 3, each disk interacts with other disks in a parity group in only 3 out of every 7 tracks.

Distributed sparing proposes an approach where the spare space on a disk is distributed on all the disks in the array instead of locating it on a single disk. This approach is shown in Fig. 4. On a failure, the data on the failed disk is reconstructed on to the spare space distributed over all the disks.

D_1	D_1	D_1	D_1	D_1	P_1	S		D_1	D_1	D_1	D_1	P_1	D_1
D_1	D_1	D_1	D_1	P_1	S	D_1		D_1	D_1	D_1	P_1	D_1	D_1
D_1	D_1	D_1	P_1	S	D_1	D_1		D_1	D_1	P_1	D_1	D_1	D_1
D_1	D_1	P_1	S	D_1	D_1	D_1		D_1	P_1	D_1	D_1	D_1	D_1
D_1	P_1	S	D_1	D_1	D_1	D_1		P_1	D_1	D_1	D_1	D_1	D_1
P_1	S	D_1	D_1	D_1	D_1	D_1		P_1	D_1	D_1	D_1	D_1	D_1
S	D_1	D_1	D_1	D_1	D_1	P_1		D_1	D_1	D_1	D_1	D_1	P_1

Fig. 4. Array organization before and after a failure in distributed sparing.

Compared to the hot sparing approach, parity sparing and distributed sparing have the following features: utilize all the disks in the system during normal operation and hence can improve the response times; the problem of un-noticed hot spare failure is avoided. Distributed sparing increases the parity group length from $n-1$ to n and the parity sparing approach (whether block designs are employed or not) decreases the parity group lengths from $n-1$ to $n/2$.

Block designs are a data layout policy and hence is independent of the sparing policy. Block designs, in general can be employed with hot sparing and distributed sparing as well. However, in a system employing block designs with two parity groups, hot spring and distributed sparing require one additional disk for providing spare space and hence increase the number of non-data disks in the system to three. We restrict ourselves to systems that employ two non-data (parity/spare) disks in the system. This gives four organizations, hot sparing, parity sparing (without block designs), block designs (parity sparing with block designs) and distributed sparing. We evaluate the performance of these four organizations in various modes of operation, with different reconstruction strategies.

In a recent paper, Menon and Mattson compare hot sparing, parity sparing (without block designs) and distributed sparing [5]. Based on queuing models, they report that among these alternatives, parity sparing and distributed sparing have several advantages over hot sparing. They also conclude that since distributed sparing offers shorter reconstruction times, it is superior among the three alternatives even though the performance of parity sparing is superior in certain modes of operation. Our results in this paper, based on simulations, expand on their work. In this paper, we show

that parity sparing with block designs, which is not considered in that paper, has better characteristics among the four alternatives when only two non-data disks are employed. We also study the impact of the system size on the choice of system organizations.

Muntz and Lui discuss several reconstruction techniques for disk arrays [7]. In their paper, they propose three reconstruction strategies termed *baseline*, *rebuild with redirection of reads*, and *piggy-backing rebuild*. In the baseline strategy, the reconstruction process rebuilds the data sequentially. With redirection of reads, some of the read requests to already reconstructed data are satisfied by reading the data from the spare disk rather than reconstructing the data from the functional disks again. In [7], a policy for this redirection with a view to minimize the reconstruction time is proposed. In piggy-backing rebuild, reads to data on failed disk cause that data block to be written to the spare. In this paper, we consider two strategies: baseline and a combination of piggy-backing rebuild with redirection of reads. We call this hybrid strategy *minimal-operation* reconstruction. This hybrid strategy *always* redirects reads of reconstructed data to the spare and *always* writes data to the spare whenever data is read from the failed disk. This strategy aims at reducing the total number of operations during the reconstruction process and this may not necessarily result in minimal reconstruction time. We present a comparison of the baseline reconstruction and the minimal-operation reconstruction. Menon and Mattson [5] consider baseline strategy for their evaluation. We consider the minimal-operation strategy in this paper since much better reconstruction times are obtained with this strategy.

2 Simulated System

In this section, the simulated system model is described.

We consider two disk arrays, one with 7 disks and the second one with 16 disks. These sizes are chosen to represent small to medium size disk arrays and since block designs exist for these sizes. The parity, data, and spare space organizations for the four schemes are shown in figures 1, 2, 3, and 4 for System 2 with 7 disks. The organizations for System 1 with 16 disks are very similar. The parity group organization for the block designs in System 1 is shown in Fig. 5, where 0 corresponds to parity group 0 and 1 corresponds to parity group 1.

Two reconstruction strategies are considered for reconstructing the data on the failed disk, baseline strategy and the minimal-operation strategy.

The disk parameters considered for this study are

0 1 1 1 1 1 0 0 0 0 1 0 0 0 0 0
1 0 1 0 0 1 1 1 0 0 0 0 0 1 0 0
1 1 0 0 0 1 0 0 1 1 0 0 0 0 1 0
1 0 0 0 1 0 1 0 1 0 1 1 0 0 0 0
1 0 0 1 0 0 0 1 0 1 1 0 1 0 0 0
1 1 1 0 0 0 0 0 0 0 0 1 1 0 0 1
0 1 0 1 0 0 0 1 1 0 0 1 0 1 0 0
0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 0
0 0 1 1 0 0 1 0 0 1 0 1 0 0 1 0
0 0 1 0 1 0 0 1 1 0 0 0 1 0 1 0
1 0 0 1 1 0 0 0 0 0 0 0 0 1 1 1
0 0 0 1 0 1 1 0 1 0 0 0 1 0 0 1
0 0 0 0 1 1 0 1 0 1 0 1 0 0 0 1
0 1 0 0 0 0 1 1 0 0 1 0 0 0 1 1
0 0 1 0 0 0 0 0 1 1 1 0 0 1 0 1
0 0 0 0 0 1 0 0 0 0 1 1 1 1 1 0

Fig. 5. Array organization in System 1 using block designs approach.

shown in Table 1. We used a nonlinear model reported in [8] to model the seek time cost function. Latency is assumed to be uniformly distributed between 0, 16.6 ms. The disks are modeled to have two queues of requests. One of the queues is the normal request queue which is operational at all times. The second queue, the reconstruction queue, is operational only while the data on the failed disk is being reconstructed and when a failed disk is being replaced. The normal request queue has a higher priority over the reconstruction queue. The requests in the reconstruction queue are served only if the normal queue is empty. Once a reconstruction request is started, it is not preempted till completion. Each disk functions independently of the other disks. We used SCAN policy for serving the requests at a disk queue.

We also used a split access operation for accessing a block of data. In split access operation, the disk starts servicing the request as soon as any of the requested blocks comes under the read-write head. For example, if a request asks for reading blocks numbered 1,2,3,4 from a track of eight blocks 1,2,...8, and the read-write head happens to get to block number 3 first, then blocks 3 and 4 are read, blocks 5,6,7,8 are skipped over and then blocks 1 and 2 are read. In such operation, a disk read/write of a single track will not take more than one single revolution. Split access operation is shown to improve the performance of the disk system considerably in [9].

When a data block from the failed disk is requested, that request may be queued at a number of queues (greater than 1) to reconstruct that block on the failed disk. This request would be queued in the normal

Table 1. Disk parameters.

Avg. latency	8.3 ms
Avg. seek	14.2 ms
sectors/track	52
sector size	512 bytes
tracks/cylinder	14
cylinders/disk	1258
seek cost function	nonlinear

queues since this is a normal request. Such a request queued at a number of queues is considered complete only when all the disks finish serving that particular block. We call such a request multi-block request. The response time of such a multi-block request is the time elapsed from the time the request is issued to the time when the last block is read from the disks. The average response time of the requests is the average of response times of normal requests. Besides these requests, reconstruction requests may be queued at the disk in the reconstruction queues. These requests are not considered in calculating the response time. However, these reconstruction requests increase the load on disks and hence may affect the response time of the normal requests. Only difference between a multi-block normal request and a reconstruction request is based on whether a user request or the reconstruction process initiated this request.

In baseline strategy, the reconstruction process reconstructs the data on the failed disk sequentially, independent of the normal requests being served. In this strategy, all the operations involved in reconstuction are queued in the reconstruction queues. In minimal-operation strategy, the normal requests to the failed disk also cause that data to be written to the spare space and hence the reconstruction of that block of data. The actual operations to write the reconstructed data on to spare space are queued in the reconstruction queues in this scheme.

Response times are measured during different phases of operation: normal mode, failure mode, reconstruction mode, reconfigured mode, restoration mode. These modes of operation are identified in [5]. Normal mode is the period during which all the disks in the system are functional. Failure mode is the period during which a disk has failed and no reconstruction process is initiated. 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 reconstructing the data on the failed disk, but before a new spare is brought into the system to replace the failed disk. Restoration mode is the period during which a new spare is brought into the system to replace a failed disk. In our simulations, during normal mode and reconfigured mode operations, only normal queues are active with single block requests. In failure mode operation, only normal queues are active, but both single block requests and multi-block requests (due to requests to the failed disk) are issued. In reconstruction mode and restoration mode, both normal queues and reconstruction queues are active with single and multi-block requests. During normal mode and failure mode operation, 10,000 normal requests are served for each data point. During the reconstruction and restoration modes of operation, the length of the simulation depended on the system being simulated. The response times during these modes of operation are the averages of response times of all normal requests served during that time. Reconstruction time, the time spent in the reconstruction mode is also measured. It is important to keep this reconstruction time to be as small as possible to avoid possible loss of data due to a second failure during this time.

Normal requests are the size of a track of data. When the I/O system has a cache, performance of the I/O system may be optimized by transferring a block of data that may be different from the actual request size [9, 8]. We chose this block to be a track in the simulations reported in this paper and hence consider all requests to the disk system to be track requests. Reconstruction is also done on a track basis. Normal requests are assumed to be uniformly distributed over all the disks and over all the blocks in a disk. Normal requests are assumed to be read requests with 70% probability and write requests with 30% probability. Requests are assumed to arrive with an exponential distribution. Request rate is varied from 25 I/Os/sec to 200 I/Os/sec for System 1 and from 16.7 I/Os/sec to 66.7 I/Os/sec for System 2.

The simulations were carried out using CSIM [10], a C based simulation package. The results obtained through simulations are presented in the next section.

3 Simulation Results

3.1 Reconstruction strategy

We evaluated the two reconstruction strategies, baseline and minimal-operation reconstruction. The results of simulation are shown in Fig. 6. The minimal-operation reconstruction strategy yields much shorter reconstruction times than the baseline strategy at higher loads. For example, the reconstruction time drops from 3800 seconds to 1400 seconds at 200 I/Os sec in the system employing hot sparing.

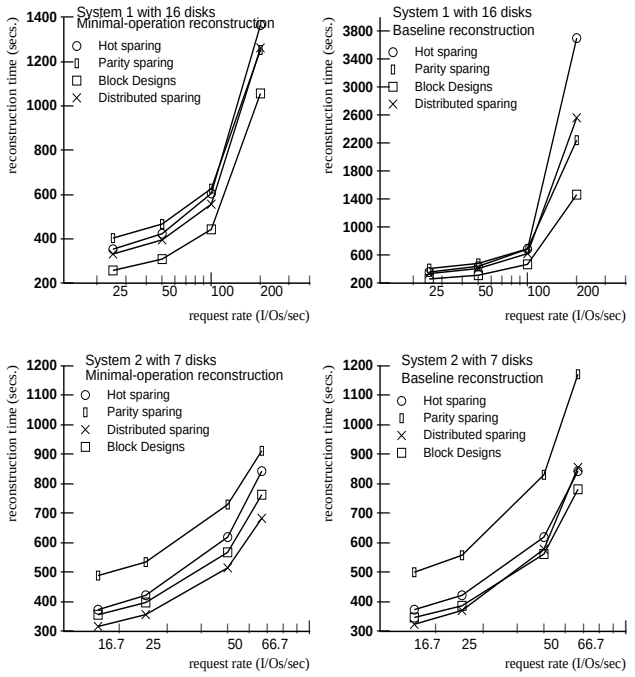

Fig. 6. Comparison of reconstruction strategies.

In minimal-operation reconstruction, the reconstructed data from the failed disk is written to the spare and necessary updates of parity for that parity group are done. At higher loads, the disks spend a larger fraction of the time serving the normal requests than the reconstruction requests. Hence at higher loads, it is more important to make use of the normal requests in assisting the reconstruction process. Minimal-operation reconstruction strategy makes use of normal requests to achieve a faster reconstruction. Also, due to request redirection, the normal requests to already reconstructed data get serviced quicker and hence allowing reconstruction requests to get served quicker as well. At low workloads, when the number of normal requests served per unit time is not very large, this benefit has less impact on the overall reconstruction time.

Minimal-operation strategy is more useful with larger parity groups. When larger parity groups are used, reconstructing data from the failed disk takes more number of operations. Hence, minimal-operation strategy is more beneficial for organizations that have larger parity group lengths. This is clearly observed in Fig. 6. We find that the minimal operation strategy has more impact on System 1 with 16 disks than on System 2 with 7 disks. For example, the 16-disk system with hot-sparing achieves a 65% reduction in reconstruction time (from that of the baseline strategy) compared to a 25% reduction in a 7-disk system. Among the different organizations, hot sparing and distributed sparing (which have larger parity group lengths) benefit more from minimal-operation reconstruction strategy than the parity sparing based approaches.

When baseline strategy is considered, block designs approach has better reconstruction times than the other organizations. This contrasts with the results of [5] where distributed sparing was shown to have best reconstruction times since block designs were not considered in that study. When minimal-operation reconstruction strategy is considered, block designs approach has better reconstruction times in System 1 and the distributed sparing approach has better reconstruction times in System 2. In parity sparing approaches, $\lceil n/2 \rceil$ disks have to be read and one of the disks (second parity disk) has to be read and written yielding $\lceil n/2 \rceil + 2$ operations per reconstructing a single block of data. Due to load imbalances in reconstruction process, the number of operations may be slightly higher. In the distributed sparing case, $n-2$ disks are read and the reconstructed data is written to the spare disk. These $n-1$ operations are incurred only for $n-1$ tracks out of every n tracks since there is no work to be done when spare space is present on the failed disk. When n is small, these numbers are almost equal and the distributed sparing approach, because of better load balance during the reconstruction process, achieves shorter reconstruction times. But when n is large, as in the 16-disk system, the parity sparing approach has far fewer operations to perform to reconstruct the data and hence achieves better reconstruction times.

At lower loads, the system organizations have a higher impact on reconstruction time than the reconstruction strategies. For example, in System 1, a reduction of 36.5% is obtained by employing block designs data layout in a parity sparing approach compared to a possible gain of only 1.5% with different reconstruction strategies. The reconstruction strategies have a larger impact on reconstruction time at higher loads. For example, in System 1 employing hot sparing, the reconstruction time is reduced by approximately 65% at 200 I/Os/sec when baseline strategy is replaced by a minimal-operation strategy. This is compared to a reduction of 1.5% at 25 I/Os/sec.

As observed from the figure, minimal-operation strategy has considerably better reconstruction times than the baseline strategy. Since the reliability of the disk array is quite dependent on the reconstruction time, we think minimal-operation strategy, even though more complex, is preferable to baseline strategy.

Since the reconstruction times are better with minimal-operation strategy, in the rest of the paper, we present results for this strategy alone. This favors the comparisons in favor of the organizations that employ

larger parity group lengths: hot sparing and distributed sparing.

3.2 Normal-mode performance

Response times, during normal operation, of the two systems considered in this paper are shown in Fig. 7. It is observed that the system with hot sparing performs worse than the other three systems. The system with hot sparing employs only $n-1$ disks, n being the total number in the system, for serving normal requests. Other systems, by employing all the n disks, improve the response times. The throughput of the system should improve roughly by a factor of $n/n-1$, i.e., by 16.6% for the 7-disk system and 6.6% for the 16-disk system. Hence, these systems obtain same throughput at lower response times when compared to hot sparing. For example, at 200 I/Os/sec in System 1, the performance difference between hot sparing and the other systems ranged from 5% to 6.4% and in System 2 at 66.7 I/Os/sec, the performance improvements are in the range of 11% to 13%.

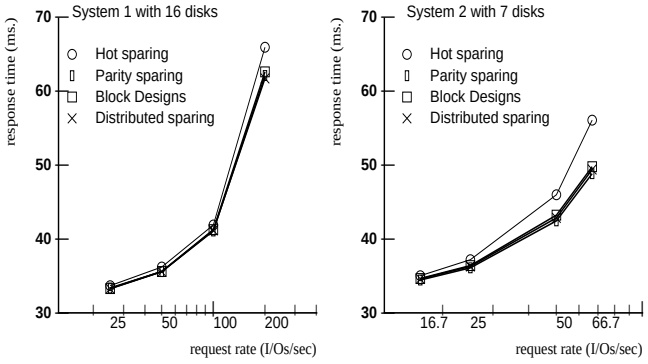

Fig. 7. Performance during normal operation.

3.3 Failure-mode performance

Fig. 8. shows the response times of the two systems when disk 0 failed. The parity sparing and the block designs approach have better average response times since a failure affects only a fraction of the disks in the array. In the other two approaches, the failure of a disk affects all the other disks in the array and hence we see a larger impact on performance. This impact on performance is more pronounced at higher loads. At lower request rates, the increased workload due to a failure does not affect the disk utilizations drastically enough to affect the performance of the various organizations.

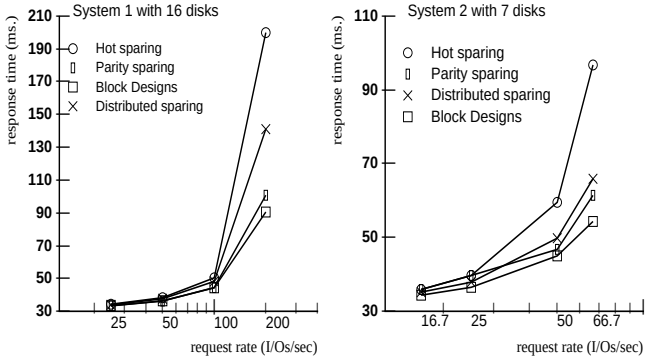

Fig. 8. Performance during failure-mode operation.

3.4 Reconstruction-mode performance

Fig. 9. shows the average response times of normal requests during the reconstruction process. The reconstruction process is given a lower priority compared to normal requests. Comparing figures 8 and 9, we observe that the average response times during the reconstruction process are better than the average response times during the failure mode even though there is more activity at the disks during the reconstruction process. As the reconstruction process progresses, any requests to already reconstructed data are served by passing the reconstructed data from the appropriate disk. As more and more data blocks get reconstructed, more and more requests to the failed disk get served in this fashion rather than through reconstructing the data on demand. Hence, as the reconstruction process progresses, the average response times improve. This improvement is visible by the improvement in average response times during the reconstruction process compared to that during the failure mode operation.

Parity sparing and block designs have better response times during the reconstruction phase because of their smaller parity group lengths. The load on the disks during reconstruction is directly proportional to the parity group lengths. Hence, smaller parity group lengths result in smaller load on the system and hence better performance. We also observed better failure-mode response times for these organizations for the same reason.

Reconstruction period varies among the different system organizations. Average response time in this mode is the average of the normal request response times over the entire period of the reconstruction mode. The response times reported in Fig. 9 are averaged over different lengths of time since the individual reconstruction periods were different among the different organizations.

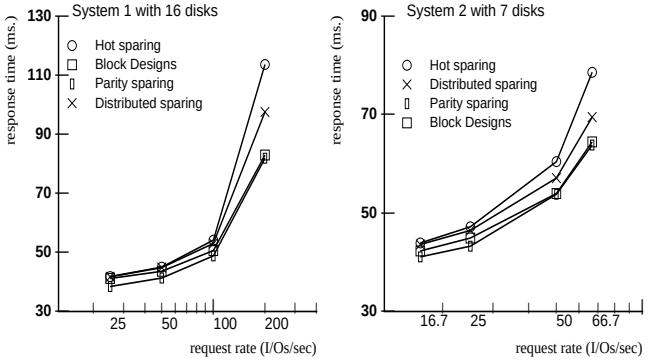

Fig. 9. Performance during reconstruction phase.

3.5 Reconfigured-mode performance

The performance of the various organizations after the data on the failed disk has been reconstructed is shown in Fig. 10. It is observed that all the systems have nearly the same response times. This is due to the fact that all the systems after reconfiguration are organized like a RAID5 array. The performance of all the organizations degrades to about the same as the hot sparing organization before a failure. The performance of the hot sparing organization before and after failure is the same since the system with that organization is the same before and after the failure.

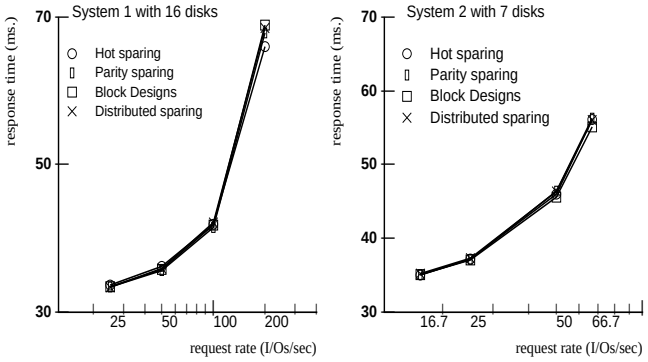

Fig. 10. Performance after reconfiguration.

3.6 Restoration-mode performance

The performance of the various organizations when a failed disk is replaced by a good disk is shown in Fig. 11. The parity sparing approach and the block designs approach have slightly worse performance than distributed sparing. In parity sparing and block design approaches, restoring the system to original organization involves reading from roughly about half the number of disks in the array and updating the two parity disks appropriately. In distributed sparing, this process involves copying the tracks from appropriate disks to the restored disk i.e., it involves a lot less activity. Hence, the observed difference in performance. In a hot sparing organization, this process of restoring the system does not involve any movement of data from one disk to another and hence that system has same response times as observed during normal performance.

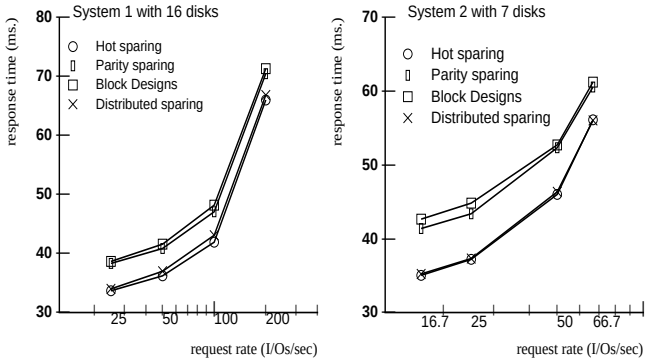

Fig. 11. Performance during restoration.

3.7 Restoration time

Fig. 12. shows the time it takes to bring back the system to original state once the failed disk is replaced. It is observed that parity sparing and the block designs approaches have considerably longer restoration periods. Restoration time is not as critical as the reconstruction time. During restoration, loss of another disk does not result in the loss of availability of data.

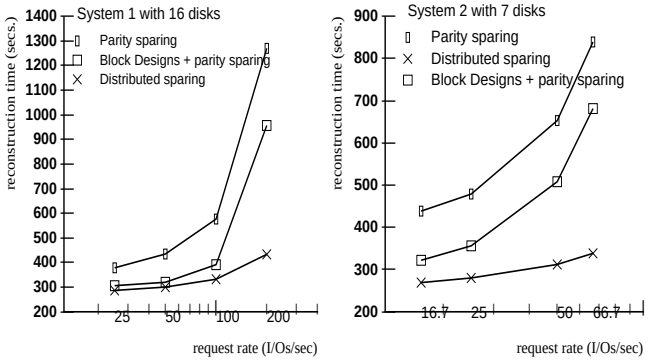

Fig. 12. Restoration times for various organizations.

4 General Discussion

We observed that parity sparing, block designs and distributed sparing approaches have significant performance advantage over hot sparing in small arrays. We also observed shorter reconstruction times with block designs and distributed sparing.

The relative advantages of distributed sparing and block designs approach combined with parity sparing depended on such factors as the number of data disks in the system, number of spare/parity disks etc. When the number of disks in the array is large, we found that the block designs approach had better characteristics than the distributed sparing approach. Distributed sparing achieved shorter reconstruction times by better distribution of load during the reconstruction mode and block designs data layout achieved shorter reconstruction times by reducing the impact of a failure on the load of the system.

Even though block designs data layout can be applied with distributed sparing, more than two spare/parity disks are needed in the system. Even with such high overhead, these systems provide only a single failure protection and hence it is unlikely that block designs and distributed sparing would be employed together. Parity sparing approach enabled employing block designs data layout with only two spare/parity disks. As shown in [6], block designs approach has diminshing performance gains with increased number of parity groups and it is unlikely that larger than two parity groups would be employed in a single array.

Block designs and distributed sparing were both observed to reduce the reconstruction time significantly over the hot sparing approach. These schemes achieve the reduction in reconstruction time in different ways. Block designs approach reduces the overall load on the system and this results in the reconstruction requests getting served quickly. Distributed sparing achieves a balanced load during reconstruction and this results in shorter reconstruction times. We observed that the parity placement has significant impact on the reconstruction times. Some block designs have good parity placements and it is important to choose these parity placements for improved performance. Of these two schemes, distributed sparing is easier to implement.

If the disk reliabilities keep improving, arrays with larger parity group lengths may be built. This reduces the overhead of parity. If this is the case, the block designs approach has more promise since it is observed that this approach offers shorter reconstruction times and better performance as the size of the array grows. On the other hand, [11] reports that smaller disk arrays have better performance in transaction processing applications. The choice of array size is dictated by these two conflicting goals of reducing the overhead of parity and sustaining good performance. Our results indicate that the choice of array size may determine the choice of array organization.

5 Conclusions

In this paper, we presented results from simulations to show how the various disk array organizations perform in various modes of operation. Our results extend or differ from previous work in the following ways: (1) we presented a more comprehensive evaluation of the various organizations through simulations; (2) we proposed a new reconstruction strategy called minimal-operation reconstruction strategy and showed that it has significantly better performance than baseline strategy; (3) we showed that the reconstruction strategy, at higher loads, has more impact on reconstruction times than the various organizations considered ; (4) we showed that among the various organizations, distributed sparing and block designs approach combined with parity sparing have performance comparable to or better than the other organizations in all modes of performance and (5) showed that the choice between these two organizations depended on the size of the disk array.

References

[1] M. Y. Kim. Synchronized disk interleaving. *IEEE Trans. Comput.*, C-35, no. 11:978–988, Nov. 1986.

[2] K. Salem and H. Garcia-Molina. Disk striping. *Int. Conf. on Data Engineering*, pages 336–342, 1986.

[3] D. A. Patterson, G. Gibson, and R. H. Katz. A case for redundant arrays of inexpensive disks (RAID). *ACM SIGMOD Conference*, June 1988.

[4] M. Livny, S. Khoshafian, and H. Boral. Multi-disk management algorithms. *Proc. ACM SIGMETRICS Conf.*, pages 69–77, May 1987.

[5] J. Menon and R. Mattson. Comparison of sparing alternatives for disk arrays. *Proc. of Int. Symp. on Comp. Architecture*, May 1992.

[6] A. L. Narasimha Reddy and P. Banerjee. Gracefully degradable disk arrays. *Proc. of FTCS-21*, June 1991.

[7] R. R. Muntz and J. Lui. Performance analysis of disk arrays under failure. *Proc. of 16th VLDB Conf.*, 1990.

[8] P. M. Chen and D. Patterson. Maximizing performance in a striped disk array. *Proc. 17th Ann. Int. Symp. on Computer Architecture*, June 1990.

[9] A. L. Narasimha Reddy. A study of I/O system organizations. *Proc. of Int. Symp. on Comp. Arch.*, May 1992.

[10] H. D. Schwetman. CSIM: A C-based, process-oriented simulation language. *Tech. Rep. PP-080-85, Microelectronics and Computer Technology Corp., Austin, Texas.*

[11] J. Menon and R. Mattson. Performance of disk arrays in transaction processing environments. *Proc. of Int. Conf. on Dist. Comp. Systems*, June 1992.