3-28-1994

Performance analysis of disk mirroring techniques

Taysir Abdalla
Florida International University

DOI: 10.25148/etd.FI13101501

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

PERFORMANCE ANALYSIS OF DISK MIRRORING TECHNIQUES

A thesis submitted in partial satisfaction of the

requirements for the degree of

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

By

Taysir Abdalla

1994

To: Arthur W. Herriott

College of Arts and Sciences

This thesis, written by Taysir Abdalla, and entitled Performance Analysis of Disk Mirroring Techniques, having been approved in respect to style and intellectual content, is referred to you for judgment

We have read this thesis and recommend that it be approved.

[illegible]
Dawn Holmes

[illegible]
Raimund Ege

[illegible]
Nagarajan Prabhakaran

[illegible]
Cyril Orji, Major Professor

Date of Defense: March 28, 1994

The thesis of Taysir Abdalla is approved.

[illegible]
Dean Arthur W. Herriott
College of Arts and Sciences

[illegible]
Dr. Richard L. Campbell
Dean of Graduate Studies

Florida International University, 1994

ACKNOWLEDGMENTS

I wish to thank my advisor, Dr. Orji, for his continuos support and encouragement. I also want to thank the members of my committee — Raimund Ege, Nagarajan Prabhakaran and Dawn Holmes.

ABSTRACT OF THE THESIS

Performance Analysis of Disk Mirroring Techniques

By

Taysir Abdalla

Florida International University, 1994

Miami, Florida

Professor Cyril Orji, Major Professor

Unequaled improvements in processor and I/O speeds make many applications such as databases and operating systems to be increasingly I/O bound. Many schemes such as disk caching and disk mirroring have been proposed to address the problem. In this thesis we focus only on disk mirroring. In disk mirroring, a logical disk image is maintained on two physical disks allowing a single disk failure to be transparent to application programs. Although disk mirroring improves data availability and reliability, it has two major drawbacks. First, writes are expensive because both disks must be updated. Second, load balancing during failure mode operation is poor because all requests are serviced by the surviving disk. Distorted mirrors was proposed to address the write problem and interleaved declustering to address the load balancing problem. In this thesis we perform a comparative study of these two schemes under various operating modes. In addition we also study traditional mirroring to provide a common basis for comparison.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF ABBREVIATIONS

CPU	Central Processing Unit
DFF	Disk Full Factor
I/O	Input/Output
OLTP	On Line Transaction Processing
RAID	Redundant Array of Inexpensive Disks

CHAPTER 1
INTRODUCTION

1.1 The Pending I/O Crisis

Processor and memory speeds have been growing at a rapid pace. New technology is making it possible to build faster processors. However, these advances in technology have not equally affected all components of a computer system thereby limiting the ability of computer systems to reach their maximum potential. One of these components that has lagged behind is the I/O subsystem, and in particular the disk subsystem. Currently, there exists a huge disparity between I/O subsystem's performance and processor performance. While advances in technology have resulted in the production of high speed processors, the progress in the disk subsystem has been primarily in increased capacity and lower cost. If this trend continues, the speed of the computer system will be limited by the speed of the I/O subsystem. When subsystems of a component experience unequal speed up, the slowest component becomes a bottleneck to the system. According to Amdahl's law [1]

$$s = \frac{1}{(1-f)+f/k}$$

Where

S = the effective speed up;

f = fraction of work in faster mode; and

k = speedup while in faster mode.

When all components of a system improve at the same rate (i.e., f = 1) then the system speed will increase by k, while if only parts of a system improve (i.e., f <

1), then the system speed up is increased by less than k.

For example, suppose that some applications spends 10% of their time waiting for I/O. Then when the CPU speed is increased by a factor of 10 — then Amdahl's law predicts that the system speed will only increase by a factor of 5, wasting 50% of potential speedup. So it is essential that improvements be made to the I/O subsystem to avoid the I/O becoming a bottleneck.

1.2 Proposed Solutions

Several schemes have been proposed to improve the I/O system performance and reliability, including disk arrays [13] and mirroring [2,4]. An example of disk arrays is RAID — Redundant Array of Inexpensive Disks [19]. RAID is a taxonomy of data architectures. In [19], five levels of RAID were defined ranging from disk mirroring as RAID level 1 and rotated parity as RAID level 5. For completeness a non-redundant single disk system was later included as RAID level 0. Figure 1 is an example of the rotated parity RAID. In a level 5 RAID, a logical unit of data is stripped across *N*-1 disks in an *N* disk array. The parity of these *N*-1 units is stored in the *N*th disk. For example in figure 1, P1 in disk4 is the parity information associated with data blocks 0, 1, 2, and 3 stored in disks 0, 1, 2 and 3 respectively. Assuming the unit of stripping is a block, then during normal operation, reading a block requires access to only a single disk. Writing a block, on the hand, requires 4 disk accesses: a read and a write access for the block being written, plus a read and a write access for the parity block. When a disk fails and 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. Since any request to the failed disk involves reading data from all survived disks, the response time during a disk failure is much higher than when all disks are operative [5,15].

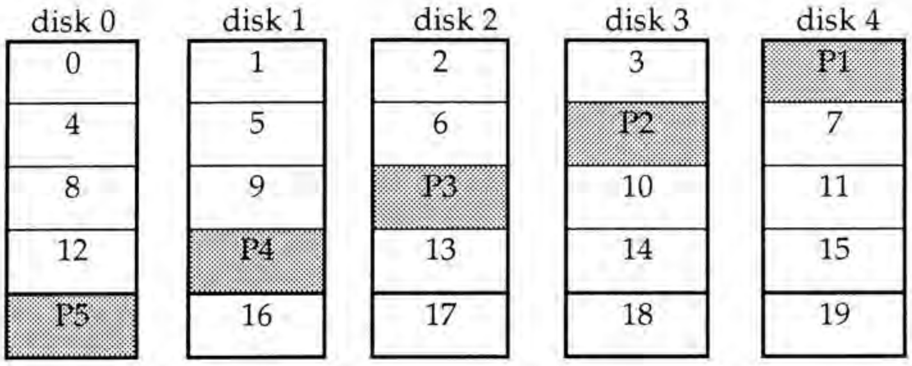

Figure 1: Disk Arrays (RAID)

In mirroring [4], a logical disk image is maintained on more than one physical disk. The physical disks that contain identical data are called the mirror set. When one disk fails, the other disk can continue to be used. Unless both disks where data is stored fail at the same time, the failure of one disk will be transparent to users and no interruption of service will occur. Since a read request can be serviced by any of the disks in the mirror set, a nearest free arm scheduling algorithm can be used to schedule a read request. This reduces the average seek distance of a read request in an n cylinder disk to about $n/5$ cylinders [3,4], in contrast to the average seek distance of about $n/3$ cylinders using a single disk. Although disk mirroring improves data availability and reliability, it has two major drawbacks:

1. **Write requests are expensive**. A write request updates both disks in the mirror set, therefore both disks arms must seek to the same location. Since the two disks arms are not synchronized, the seek cost will be the maximum of the two random seeks.

2. **Poor load balance in the event of failure**: When a disk fails, all requests for data on the failed disk are redirected to the image disk, potentially doubling the load on that disk. This could lead to a major degradation in system performance.

Many schemes have been proposed to deal with the write and load balancing problems in mirrored disks. By batching writes as proposed in [21], disk mirroring with alternating deferred updates [20] achieves improved write performance even with increasing write ratio. In [22,16] distorted and doubly distorted mirrors were used to improve small write performance by up to a factor of 2 over traditional mirroring. To address the load balancing problem interleaved declustering [25] and chained declustering [11,6] were proposed. In the next section we discuss three of these mirroring schemes, traditional mirroring, distorted mirroring, and interleaved declustering. We consider these three schemes to be representative of the various forms of mirroring.

1.3 Mirroring Techniques

In this section we will discuss traditional mirroring, interleaved declustering, and distorted mirroring.

1.3.1 Traditional Mirroring

Disk mirroring [4], also called shadowing, is a technique for replicating 1 logical disk across two physical disks. The two physical disks are exact mirrors of one another. During normal operation, each disk in the mirror set can service a given read request, potentially doubling throughput. However, since both disks contain identical data, a write request must be serviced by both disks. Figure 2 is an example of a mirror set of size 2. Each disk is an identical copy of another disk; for example disks 0 and 4 form a pair, so also disks 1 and 5, 2 and 6, and 3 and 7.

When a disk in a mirrored system fails, the surviving disk in the set assumes the workload of the failed disk until the failed disk is reconstructed in a new disk. Disk reconstruction usually involves copying the survivor disk to a new disk. Unless both disks in the mirror set fail at the same time, data will always be available.

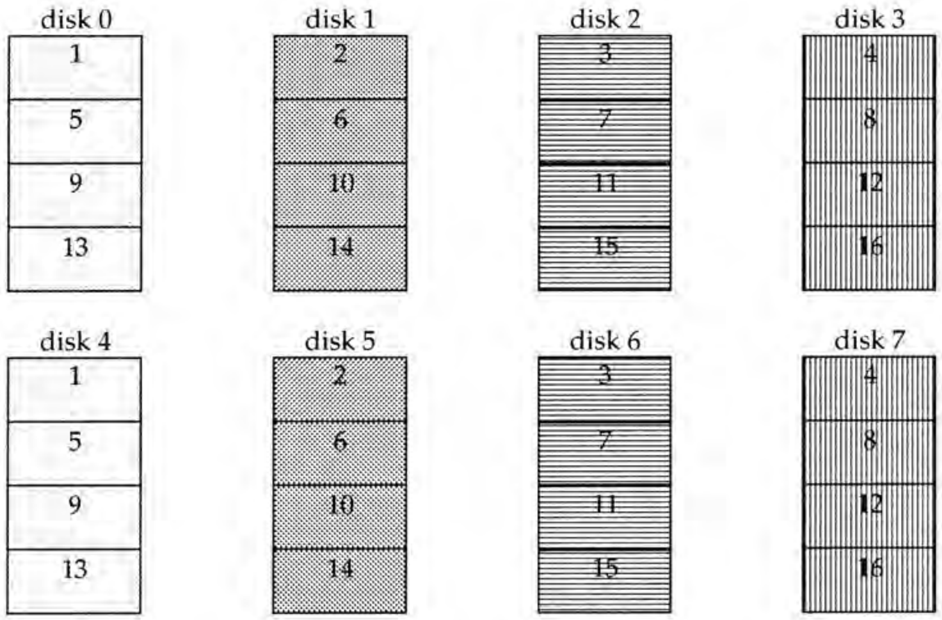

Figure 2: Traditional Mirroring (Mirror set = 2)

Assuming that a second failure does not occur during the replacement window, degraded mode performance depends heavily on the type of workload experienced. In high read environments, the failure of a disk doubles the load on its mirror. However, if most requests are write requests, degraded mode performance may even be better than normal mode performance since the write cost is now the cost of a single disk write.

1.3.2 Distorted Mirroring

Distorted mirrors [22,23] is a mirroring technique that addresses the write deficiency of traditional mirroring. Like traditional mirrors, it replicates 1 logical disk across 2 disks in the mirror set. Unlike traditional mirrors, the organization of data on the disks is not identical. Each disk in the set contains the same logical blocks but in permuted order. Hence, the name distorted mirroring.

Each physical disk is divided into two unequal parts, a master partition and a slightly larger slave partition. The two master partitions in a mirror set form a logical disk; for example in Figure 3, $M0 \cup M1$ = 1 logical disk. Each block in a logical disk is mastered on one disk and slaved on the mirror. Blocks are organized sequentially in the master partition and written arbitrarily in the slave partition. In Figure 3 the blocks in $M0$ are mastered and written sequentially on disk 0, the same blocks are written arbitrarily in the slave partition $S0$ of disk 1.

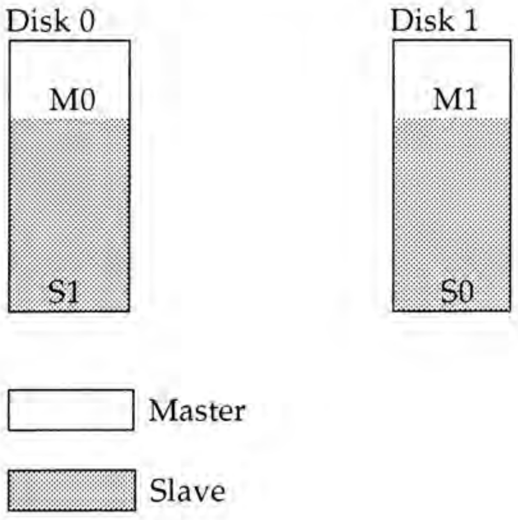

Figure 3: Distorted Mirroring

Distorted mirror uses a *diskfull factor* (DFF) to determine the amount of free space in the slave partition. DFF is the proportion of the physical disk that holds valid data. For example , a DFF of 80% means that 40% of the physical disk is used as master partition and 60% as slave partition. We sometimes use the term *backup factor* to present the same information. DFF is defined as 100% - backup factor.

During normal operation, single block requests can be serviced by the master or the slave disk while sequential read requests are serviced by the master disk. Write requests are serviced by both. Like traditional mirroring, a surviving disk in a mirror set picks up the load when its partner disk fails. However, unlike traditional mirroring, disk reconstruction is more complex and time consuming since a disk-to-disk copy can not be performed.

The advantage of this scheme is that slave writes can be performed without a disk seek and with reduced latency. So the cost of a write request is bound by the seek and latency cost of the master disk. However this scheme has two major drawbacks:

- Rebuilding a disk in case of a disk failure is a costly operation. Since the disks in the mirror set are not identical, when a disk fails, a disk to disk copy is not possible during recovery.
- The distortion map needed to support this scheme must be kept in memory for this scheme to operate efficiently. Since large disks are currently used, this map will require a large portion of main memory.

1.3.3 Interleaved Declustering

The Teradata database machine [25] uses interleaved declustering for fault tolerance and load balancing in the event of a failure. In interleaved declustering the mirror set is divided into clusters. Each cluster contains from 2 to 16 disks. A relation is declustered among the disks within one or more clusters. Each cluster contains two copies of each relation. One copy is designated the primary copy and the other as backup copy. Figure 3 shows a relation R declustered across 8 disks (*Ri* represents the *i-th* primary fragment whereas *ri* represents the *i-th* backup fragment). These 8 fragments are divided into 2 clusters, cluster 0, and cluster 1. Each relation fragment *Ri* is partitioned into fragments which are stored in the other disks in the same cluster. For example, *r*0.0, *r*0.1, *r*0.2 fragments of *R0* are stored in disks 1, 2 , and 3 respectively.

During normal mode operations, read requests are serviced by the primary fragments and write requests update both primary and the backup fragments. When a node fails, an appropriate backup copy is promoted as a primary copy and the workload of the failed disk is shared by the remaining disks in the cluster. This reduces the chance of overload of a single disk; a situation that is likely to occur in other mirrored schemes.

	Cluster 0				Cluster 1			
Node	0	1	2	3	4	5	6	7
Primary Copy	R0	R1	R2	R3	R0	R1	R2	R3
Backup Copy		r0.0	r0.1	r0.2		r4.0	r4.1	4.2
	r1.2		r1.0	r1.1	r5.2		r5.0	r5.1
	r2.1	r2.2		r2.0	r6.1	r6.2		r6.0
	r3.0	r3.1	r3.2		r7.0	r7.1	r7.2	

Figure 4: Interleaved Declustering (cluster size = 4)

1.4 Statement of the Problem

Parallelism and redundancy are two common techniques used in high performance systems to provide fault tolerant computing. However, because the system is designed with these as primary factors, it is not clear how the system performs when it loses this capability. For example, when a system is designed to transfer data in parallel and it loses one of the transfer components, it is not clear what the impact on the entire system will be. This issue has attracted a lot of attention in the research community and we intend to pursue an aspect of it a little further in this thesis.

When a disk fails in a system that uses replication, it is necessary to provide immediate repair of the failed disk to avoid data loss. Usually, standby disks are provided and the contents of the failed disk are rebuilt on the standby disk from the redundant information on the surviving disks. The behavior of the system during the rebuild process is dependent on the availability scheme being used and the rebuild algorithm.

Many researchers [5,8,10,15] have studied the behavior of mirroring and RAID-5 during failure mode. We briefly overview some of these studies in the next section.

1.5 Related Work

In [14], three operating modes were defined for drives in a disk array. These modes were used as a basis for evaluating the performance of drives in a RAID-5 array. We use similar modes for evaluating the three mirrored schemes. These are *normal, degraded,* and *rebuild* operating modes. The system operates in normal mode when all disks are available. When a disk has failed and no attempt is being made to reconstruct it, it is said to be in degraded mode. During degraded mode any request to the failed disk is serviced by the backup disk. In order to rebuild the failed disk the system moves to the rebuild mode. If a second drive fails while the system is in degraded mode or before the failed drive is fully reconstructed in the rebuild mode, the system will lose data.

Studies have been performed to evaluate techniques for reconstructing a failed disk in some of these high availability techniques. Using analytical model, Munz and Liu [15] developed a *baseline copy* algorithm for rebuilding a failed disk in RAID-5 array. The baseline copy algorithm is similar to disk-to-disk copy algorithm except that in this case, the survivor disks are read and their data used to regenerate data in the failed disk. All read requests are directed to the survivor disk. If a read request maps to the failed disk, the data is regenerated from the survivor disks. There are two enhancements to the baseline copy algorithm- *redirection of reads*, and *piggybacking*. Redirection of reads allows read requests to be redirected to the replacement if the data has been reconstructed. With piggybacking, if data is regenerated to satisfy a read request that maps to a failed disk, the data is also written to the replacement disk so it does not have to be regenerated again.

The studies of Muntz and Lui were theoretical. However, they were sound and formed a basis for subsequent work in this area. For example, Holland and Gibson [8] extended these algorithms and performed simulation studies to verify them. Using Simulation, they were able to detect some nonintuitive behavior of some of the algorithms. For example, their simulation showed that redirection of reads and piggybacking may not always be beneficial especially when the survivor disks are not saturated.

In [9], the affect of the rebuild unit on rebuild time and response time was investigated. Three rebuild units - *block, track,* and *cylinder* were studied and their impact on rebuild time and response time of incoming application requests investigated. Using a block as the rebuild unit ensures a small delay for incoming application requests and minimally raises their response times. The drawback is an increase in the time to rebuild the failed disk. On the other hand using a cylinder as a rebuilding unit decreases the time to rebuild a disk while increasing response time. Hou, Menon and Pat concluded that a track rebuild unit provides the best compromise between response time and rebuild time.

There have been other studies in rebuilding disk arrays. Menon and Kasson studied various sparing alternatives in disk arrays. Their study which concentrated on small disk subsystems (fewer than 32 disks) concluded that distributed sparing offers major advantages over dedicated sparing in normal, degraded, and rebuild modes operation [14]. In a similar, but different study, Chandy and Reddy [5] using simulations, found that parity sparing with block designs, which was not considered in [14], has better characteristics among the alternatives they investigated.

Most of the studies in the literature have concentrated on failure evaluations of RAID-5 arrays. However, in [10], Hou and Patt evaluated RAID-5 arrays and similar capacity mirrored arrays. No new algorithms were developed in this study, rather, RAID-5 and traditional mirroring organizations were evaluated on the basis of the algorithms developed in [15,8].

1.6 Thesis Objective

Most of the studies we looked at in the previous section focused on RAID-5 or rotated parity RAID. Those studies that have looked at mirroring assumed a traditional mirroring scheme. Since other mirroring schemes have different organizations than traditional mirroring, results from traditional mirroring studies may not necessarily hold for other mirroring schemes.

In this thesis we present a detailed study of the behavior of various mirroring schemes under various operating modes. Specifically, using simulation, we

- Evaluate the relative performance of three mirroring schemes under normal, degraded, and rebuild mode. The three schemes are traditional mirroring, distorted mirroring and interleaved declustering. Traditional mirroring is considered the base mirroring scheme and provides a basis for comparative evaluation against other schemes. Distorted mirroring is chosen as an example of a scheme that addresses the write problem of traditional mirroring. We use Interleaved declustering as an example of a scheme that addresses the load balancing problem.
- Study the tradeoffs between availability and performance as they relate to the three schemes.
- Investigate alternative algorithms to quickly reconstruct a failed disk and hence return the system to fully operational mode.

CHAPTER 2
THE SIMULATOR

2.1 Simulation Model

We wrote a simulator to evaluate the expected performance of traditional mirroring, interleaved declustering and distorted mirroring under various operating modes. We assumed an array of 16 disks; thus for the traditional and distorted mirrors, there are 8 mirror sets, with 2 disks in each set. The two disks contain identical data as described in chapter 1. For the distorted mirror, 80% diskfull factor was used. The implication of this is that in the distorted mirror, the logical disk is slightly smaller than the logical disk in the traditional mirror. However, we considered the effect of this to be minimal and is not considered any further in the experiments. For the interleaved declustering, we partitioned the 16 disks into 2 clusters (clusters 0 and 1), with a cluster size (C) of 8 disks. Data distribution in the 8 disks is as described in chapter 1. One-half of each physical disk contains primary data blocks, and the other half contains backup data blocks. The primary data blocks of each disk are logically divided into 7 (or C - 1), and the parts are uniquely placed in the backup partitions of the other 7 disks in the same cluster. The logical block distribution in cluster 0 is shown in Table 1. For example, D0 represents 50% of the data blocks in *disk 0* and represents the primary partition of this disk. D0 is further divided into *d0.i* fragments ($0 \leq i \leq 6$), and these fragments are placed in the remaining disks in the cluster. Note also that the remaining space in disk0 contains fragments of the primary partitions of the other disks as backup copies.

The parameters of the disk drives modeled in the experiments are shown in Table 2. The parameters are identical to those of IBM 0661 Model 370 disk

	Disk 0	Disk 1	Disk 2	Disk 3	Disk 4	Disk 5	Disk 6	Disk 7
Primary	**D0**	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**
	—	d0.0	d0.1	d0.2	d0.3	d0.4	d0.5	d0.6
	d1.6	—	d1.0	d1.1	d1.2	d1.3	d1.4	d1.5
	d2.5	d2.6	—	d2.0	d2.1	d2.2	d2.3	d2.4
Backup	d3.4	d3.5	d3.6	—	d3.0	d3.1	d3.2	d3.3
	d4.3	d4.4	d4.5	d4.6	—	d4.0	d4.1	d4.2
	d5.2	d5.3	d5.4	d5.5	d5.6	—	d5.0	d5.1
	d6.1	d6.2	d6.3	d6.4	d6.5	d6.6	—	d6.0
	d7.0	d7.1	d7.2	d7.3	d7.4	d7.5	d7.6	—

Table 1: **Interleaved Declustering: A Cluster of 8 Disks** *Each disk holds primary and backup blocks. The backup blocks hold copies of the fragments of primary blocks in the other disk.*

Disk Parameters	No of cylinders	949
	Track per cylinder	14
	Blocks per track	6
	Block size	4096
	Average seek	12.50 ms
	Rotational speed	4318 RPM
	Seek cost function	$2.0 + 0.01(\text{distance}) + 0.46\sqrt{\text{distance}}$
	Latency cost function	uniform
Workload Parameters	Request Size	4 KB
	Read Ratio	0%, 50%, 75%, 100%
	Request Distribution	Uniform over all disk blocks
	I/O Rate	100 to 350 per second

Table 2: **Disk and Workload Parameters**

drive. Also shown in Table 2 is a summary of the workload parameters used in the experiments. The system may experience two types of workloads depending on its operating mode. Under normal and degraded operating modes, all requests are assumed to be initiated by application programs. The workload is modeled as a synthetically generated stream of disk requests characterized by its type (whether a read or a write). Requests are generated uniformly across the disk space and each request accesses 4KB (one block) of data. Other studies have used 75% read, 25% write as a representative mix for most on-line transaction processing (OLTP) applications. We also studied this mix, but in addition, we studied an all write workload (0% read), an all read workload (100% read) and a workload with an equal mix of read and write requests (50% read). We varied the I/O rate to the 16 disks from 100 I/Os to 350 I/Os per second.

When the system is in rebuild mode, in addition to the normal application requests, the system initiates rebuild requests. A rebuild request triggers a disk read on the surviving disk(s) and a corresponding disk write on the replacement disk. In all the schemes, requests generated by the application are given higher priority over rebuild requests. A rebuild request is serviced if and only if there is no outstanding application request; however, once started, a rebuild request cannot be preempted by an application request.

2.2 Rebuild Algorithms

In this section we describe the algorithms used during the rebuild phase in each mirroring scheme.

2.2.1 Traditional Mirror

The rebuild algorithm for traditional mirror is similar to the baseline copy algorithm in [15]. However, the algorithm does not perform a sequential scan of the disk; instead, whenever there is no outstanding application request, a rebuild request is initiated for the ``nearest'' *unrebuilt* unit in the survivor disk. This policy minimizes arm motion in the survivor disk but could cause a substantial disk seek in the replacement disk. The reason is that the disk arms in the two disks are randomly positioned since all incoming application read requests are serviced by only the survivor disk. A variation of this algorithm would initiate the rebuild on the ``nearest'' unrebuilt unit in the replacement disk, thereby potentially increasing arm motion in the survivor disk. We preferred not to explore this alternative since the survivor disk is already highly loaded by normal application and rebuild requests.

In our experiments, three rebuild units were considered — block, track and cylinder rebuild units. Each unit defines the amount of data atomically read from the survivor disk and atomically written to the replacement disk. For example, if a track rebuild unit is in effect, a read request is initiated for the ``nearest'' unrebuilt track on the survivor disk. In an enhancement to this algorithm, an application read request could be redirected to the replacement disk if the requested data has been reconstructed. If redirection is in effect, the choice of disk to service the request is made as in the normal mode of operation. The preferred choice is to schedule the request on an idle disk. If both disks are busy, a minimum service time model is used. Ties are randomly broken. If an application requests a write, the block is written to both disks and is considered reconstructed regardless of whether it was previously reconstructed or not.

2.2.2 Distorted Mirror

In distorted mirror a logical block has a fixed address on the master disk and a floating address on the slave disk. Blocks mastered on one disk are slaved on the mirror; consequently, a disk-to-disk copy algorithm cannot be used to reconstruct a failed disk. As in the other schemes, three rebuild units — block, track and cylinder were considered. With block rebuild unit, the algorithm simulated is exactly identical to the traditional mirror algorithm.

But the situation is different with track and cylinder rebuild units. With these rebuild units, we divided the reconstruction process into two phases. In the first phase, we reconstruct the slave partition of the replacement disk, and reconstruct the master partition in the second phase. This order of reconstruction has significant impact on system performance. The slave partition can be speedily reconstructed in less than 20% of the total reconstruction time. The reason for this will be obvious shortly; but given that 50% of the replacement disk blocks are reconstructed in such a short time, they can be made available for servicing incoming application requests.

First Phase: Reconstructing Slave Partition of Replacement Disk

Slave partition reconstruction involves reading data from the master partition of the survivor disk, and writing the data anywhere in the slave partition of the replacement disk. Since the replacement disk is initially empty, writing blocks anywhere in the slave partition incurs minimum cost because free blocks are easily located. Moreover, enough contiguous free blocks can be located (most of the time) to allow many sequential transfers. In addition, since data is sequentially written in the master partition, reading the survivor disk and writing the replacement disk can proceed at sequential access rate in this phase. This is why we reconstruct the slave partition before the master partition.

Second Phase: Reconstructing Master Partition of Replacement Disk

Master partition reconstruction involves reading data from slave partition of the survivor disk, and writing the data to the master partition of the replacement disk. Fetching slave blocks that map to contiguous addresses in the master partition requires random disk accesses on the slave disk. Assume cylinder rebuild unit is in effect; the first step in the rebuild process is to select the cylinder to reconstruct. The obvious choice is to choose a cylinder that minimizes the write cost on the replacement disk[1]. The blocks are randomly fetched from the survivor disk and sequentially written to the replacement disk. The master partition reconstruction is thus dominated by the random disk accesses to read

[1]Due to the random placement of blocks in the slave partition of the survivor disk, the cost of fetching the blocks is independent of the cylinder chosen

the blocks from the survivor disk.

Some points are in order here. In each experiment, we used a buffer that holds the number of blocks in the rebuild unit. For example, a buffer that holds just a block is used when the rebuild unit is block. An alternative that could improve master partition reconstruction time is to use write buffering techniques [21]. In this technique, a large buffer is used to hold blocks and the disk transfer is delayed until enough blocks have accumulated in the buffer. Blocks that map to ``nearby'' locations on disk can be written together. We did not explore this alternative in the current study. Another alternative which we also did not explore is the use of two replacement disks [20]. In this technique, data is copied to two replacement disks and at the end of reconstruction, the survivor disk and failed disk are returned to the system as spares. The advantage of this technique is that during reconstruction, all incoming write requests are directed to the two replacement disks and the survivor disk is placed in read-only mode relieving it of all write operations.

2.2.3 Interleaved Declustering

The rebuild algorithm for interleaved declustering is similar to that of traditional mirroring. Because reconstructing data on the failed disk involves all the other disks in the same cluster, choosing the disk to fetch data from, at any point during the reconstruction process can impact performance. In our experiments, we considered two disk selection options. In the first option, the survivor disk in the cluster that minimizes the fetch time for a unit of unrebuilt data is selected. This scheme leads to a minimal increase in disk utilization for the survivor disks, and also to a minimal increase in response time for incoming application requests. In the second option, we aim to minimize the (seek) cost of writing data to the replacement disk. We select a survivor disk that helps us achieve this objective. This second option ensures that the replacement disk performs optimally. The result is that rebuild time is shorter than in the first option. Results reported in this thesis were obtained using the first option. Block, track and cylinder rebuild units are also investigated.

CHAPTER 3
SIMULATION RESULTS

3.1 Performance Metrics

In this chapter we present results of our experiments. The results are presented in three sections. In Section 3.2 we present results of the systems when operating in normal mode. In Section 3.3 we present degraded mode performance and present rebuild mode performance in Section 3.4. We present results for various read/write ratios. However, our discussion will focus on results obtained for 75% read workload, which is typical of transaction processing environments. In Section 3.5 we briefly discuss some results representative of other read/write ratios.

The performance metrics used in our studies depend on the operating mode being investigated. Under normal and degraded operating modes, the response time of application read requests was of interest. Under rebuild mode, in addition to the response time of application read requests, we also collected data on the time taken to rebuild a failed disk (rebuild time). We note that all response time data are given as 90th percentile response time for application read requests.

3.2 Normal Mode

Figures 5 and 6 show the relative performance of the three schemes with respect to response time of read requests. At low I/O rates, the three schemes have comparable performance. However, as I/O rate increases, the performance of traditional mirroring degrades more dramatically than the performance of the other schemes. This is because of the write requests in the request stream. Since writes are expensive in traditional mirror, every write request potentially adds

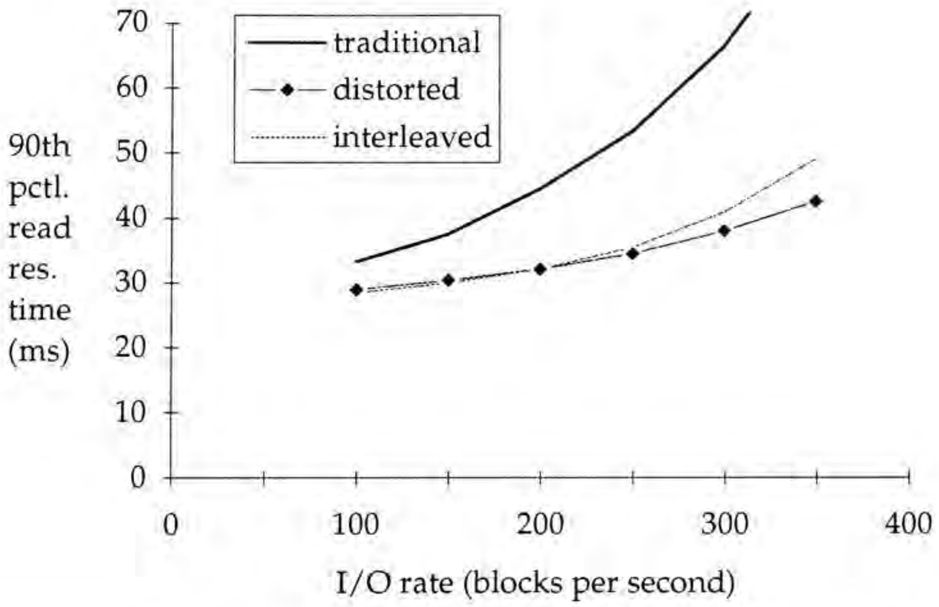

Figure 5: 90th Percentile Read Response Time vs I/O Rate (50% read: Normal mode)

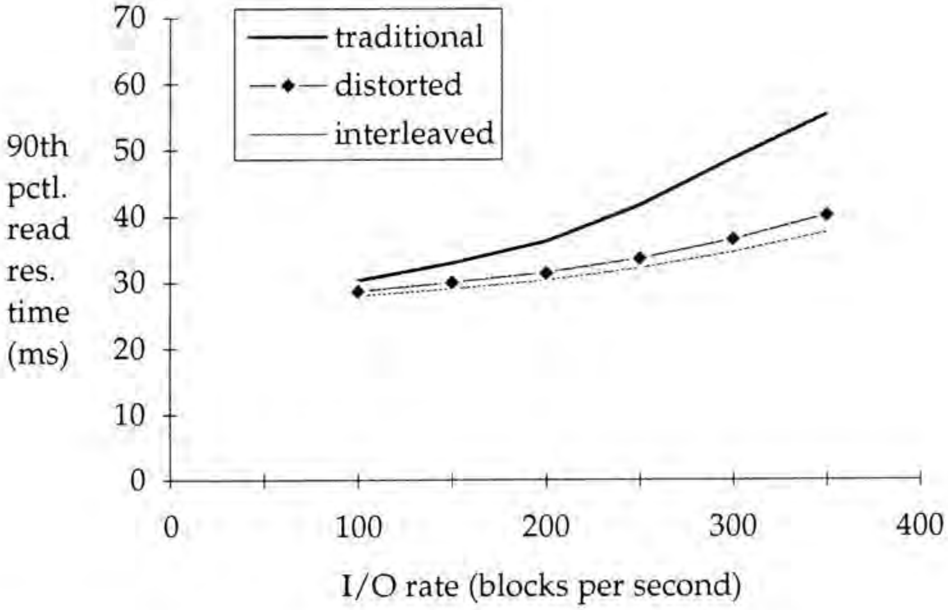

Figure 6: 90th Percentile Read Response Time vs I/O Rate (75% read: Normal mode)

to the queuing time of pending requests. In Figure 6 at 250 I/Os per second, the 90th percentile read response time is 32.12 ms for interleaved declustering. The corresponding figures for traditional and distorted mirrors are 41.65 ms and 33.61 ms respectively.

Figure 7 shows the disk's utilization of the three schemes at 75% read workload. Distorted mirror has the lowest utilization. This is due to the ability of Distorted mirror to perform write requests with minimum seek and latency cost. At 100% read (0% write) workload, the disk's utilization of the three schemes are identical (graph not shown). In Figure 7 at 250 I/Os per second, the disk's utilization is 39.12% for interleaved declustering. The corresponding figures for traditional and distorted mirrors are 40.33% and 31.20% respectively.

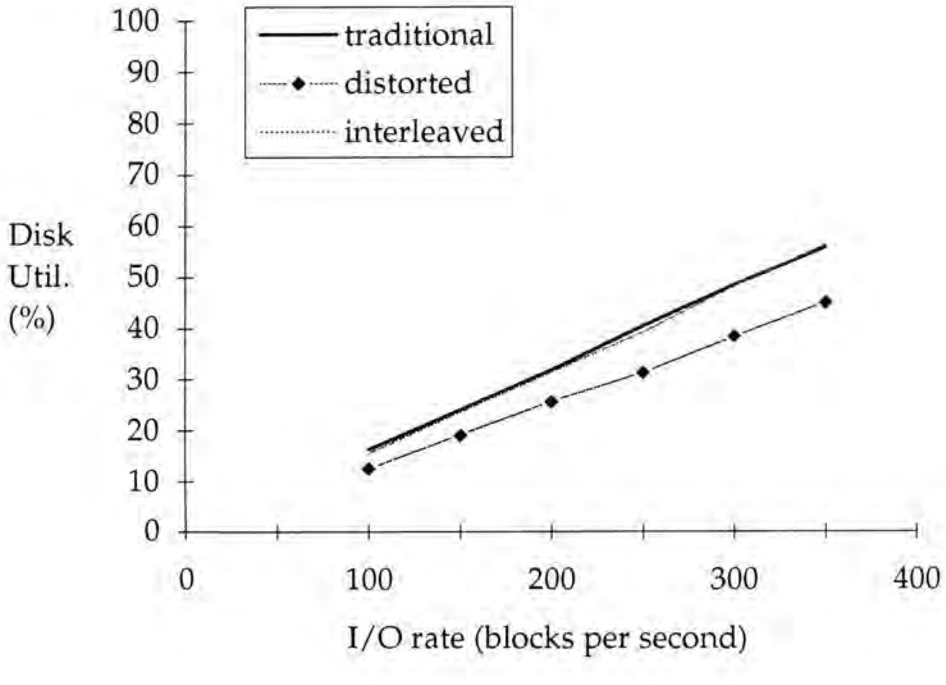

Figure 7: Disk Utilization vs I/O Rate (75% read: Normal mode)

3.3 Degraded Mode

Figures 8 and 9 show the response time of all application read requests when the system is in degraded mode (50% and 75% read workload). The performance of distorted mirror appears similar to that of interleaved declustering. In general, we observed that with some writes in the request stream, traditional mirror is unable to sustain high I/O rates; hence the sharp decline in performance for traditional mirror in Figures 8 and 9 as the I/O rate was increased. However, as the read workload increases, the performance of distorted and tradtional mirrors becomes similar. For example, at 250 I/Os per second with 100% read, the response time for application requests in interleaved declustering is 30.92 ms. The corresponding figures for traditional and distorted mirrors are 41.06 ms and 37.13 ms respectively.

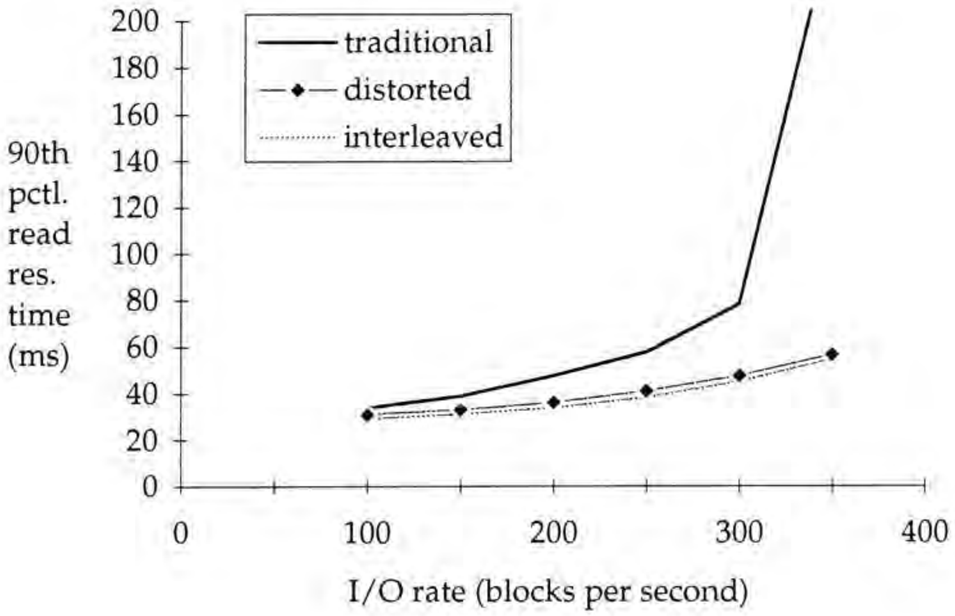

Figure 8: 90th Percentile Read Response Time vs I/O Rate (50% read: The system is in degraded mode and response time is measured *over all the application read requests. This includes requests that map to the mirror set with a failed disk*)

In Figures 10 and 11 we isolate and show the response time for requests that map to the set with a failed disk. The superior performance of interleaved declustering in degraded mode becomes more obvious. In Figure 11 at 250 I/Os per second, with the system in degraded mode, the response time in interleaved declustering increases by about 7.3% over normal mode response time. Increases were also observed in the other schemes. For distorted and traditional mirrors these were 148.5% and 113.2% respectively. These numbers demonstrate the ability of interleaved declustering to spread the workload of a failed disk over many disks.

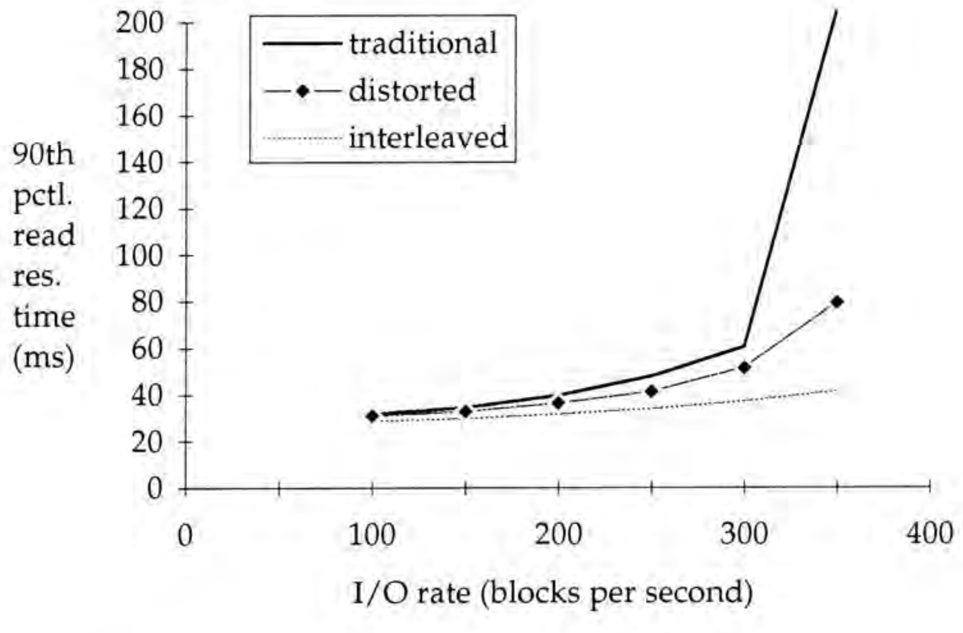

Figure 9: 90th Percentile Read Response Time vs I/O Rate (75% read: The system is in degraded mode and response time is measured *over all the application read requests. This includes requests that map to the mirror set with a failed disk*)

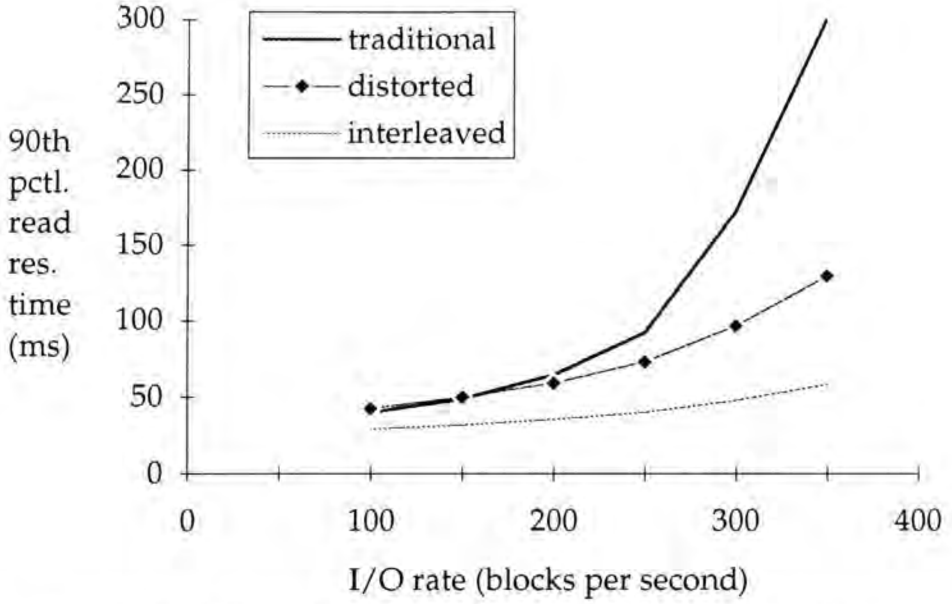

Figure 10: 90th Percentile Read Response Time vs I/O Rate (50% read: The system is in degraded mode and response time is measured over *only those application read requests that map to the mirror set with a failed disk*)

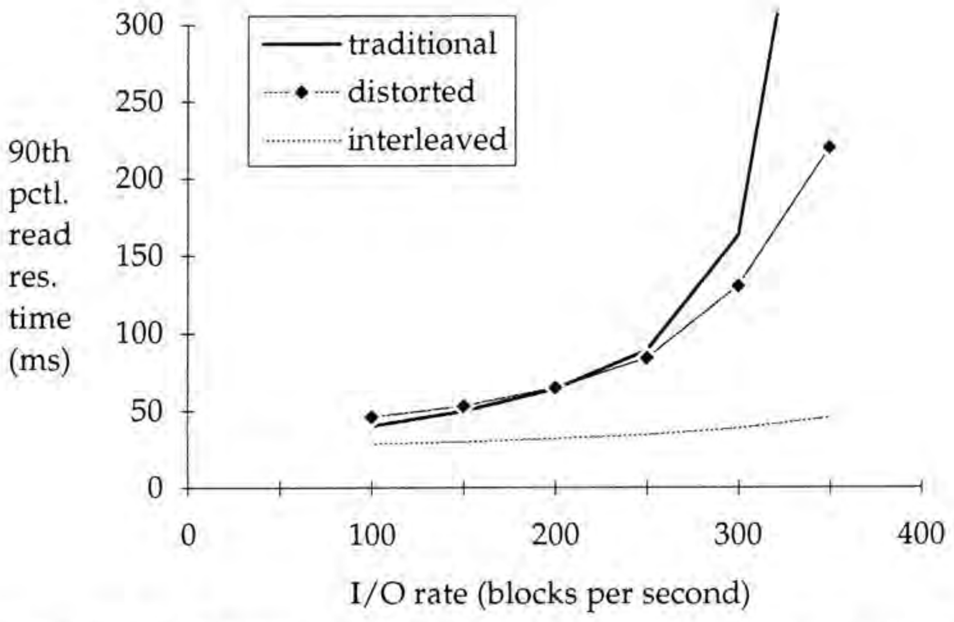

Figure 11: 90th Percentile Read Response Time vs I/O Rate (75% read: The system is in degraded mode and response time is measured over *only those application read requests that map to the mirror set with a failed disk*)

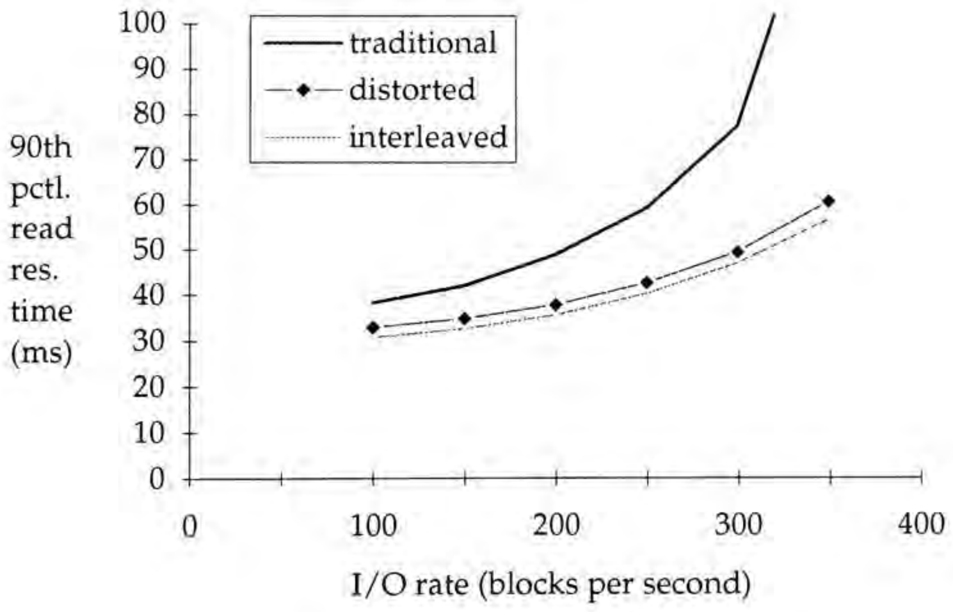

Figure 12: 90th Percentile Read Response Time vs I/O Rate (50% read: The system is in rebuild mode and response time is measured over *all the application read requests. This includes requests that map to the mirror set with a failed disk.* The rebuild unit is block)

3.4 Rebuild Mode

We use two metrics for evaluating performance in rebuild mode — read response time and rebuild time. First we consider block rebuild unit, then track and cylinder rebuild units. We focus on results for 75% read (25% write) workload.

Figures 12 and 13 show the response time of read requests while the system is in rebuild mode. The curves in Figure 13 show that over the range of I/Os simulated, traditional and distorted mirror have comparable performance. The graph shows rapid saturation of traditional and distorted mirror beyond about 300 I/Os per second. At 250 I/Os per second, there is a 12.0% increase over normal mode response time in interleaved declustering. Corresponding

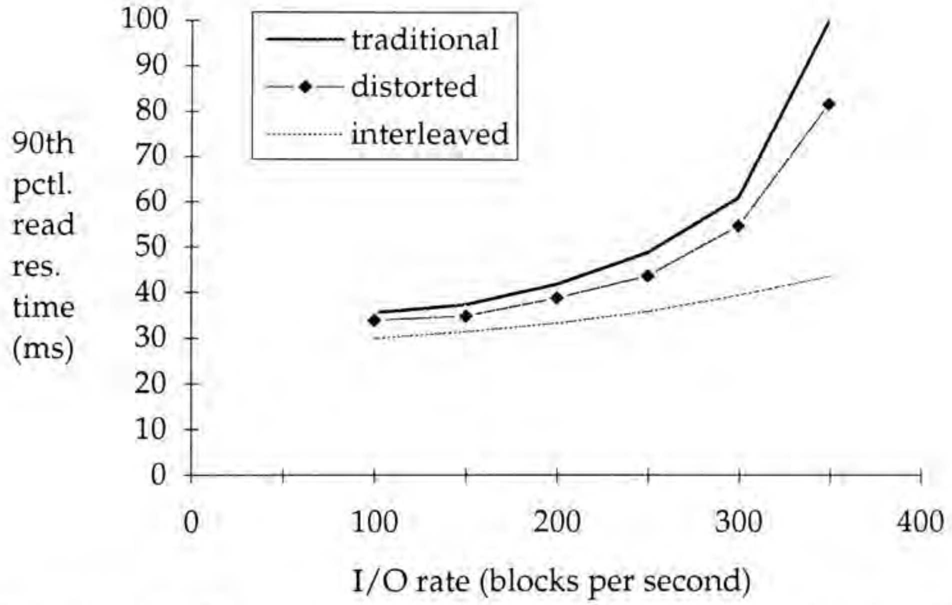

Figure 13: 90th Percentile Read Response Time vs I/O Rate (75% read: The system is in rebuild mode and response time is measured *over all the application read requests. This includes requests that map to the mirror set with a failed disk.* The rebuild unit is block)

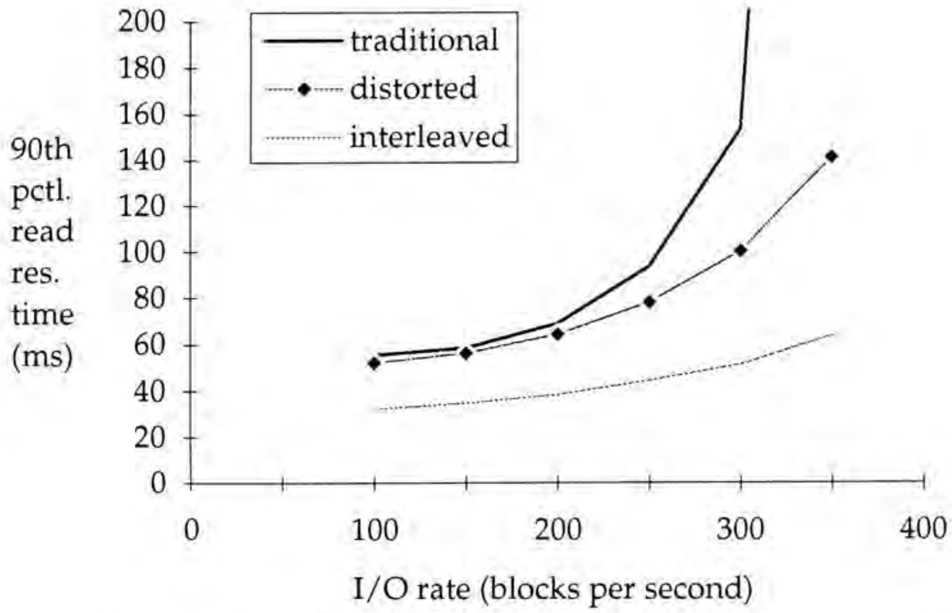

Figure 14: 90th Percentile Read Response Time vs I/O Rate (50% read: The system is in rebuild mode and response time is measured over *only those application read requests that map to the mirror set with a failed disk.* The rebuild unit is block)

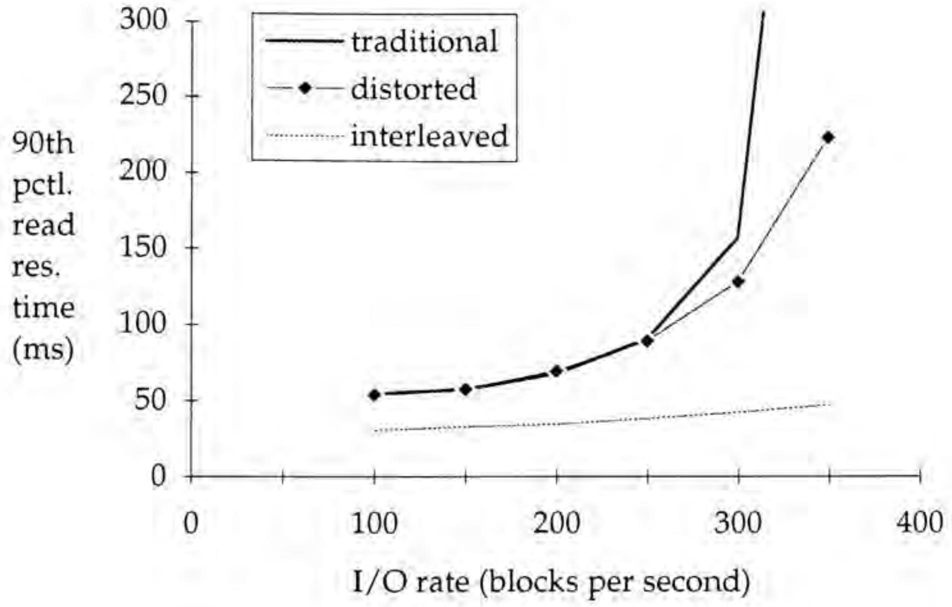

Figure 15: 90th Percentile Read Response Time vs I/O Rate (75% read: The system is in rebuild mode and response time is measured over *only those application read requests that map to the mirror set with a failed disk.* The rebuild unit is block)

increases for distorted and traditional mirrors are 30.9% and 17.88% respectively. However, if we consider only requests that map to the failed set, the response time increases for interleaved declustering, distorted mirror and traditional mirror are 19.3%, 167.8% and 119.2% respectively (Figures 14 and 15). This again demonstrates the ability of interleaved declustering to balance its load in failure mode.

Block rebuild time is graphed in Figure 16. Over the range of I/Os simulated, interleaved declustering has the best rebuild time. For the three schemes, an increase in I/O rate means reduced bandwidth for the rebuild process thereby increasing rebuild time.

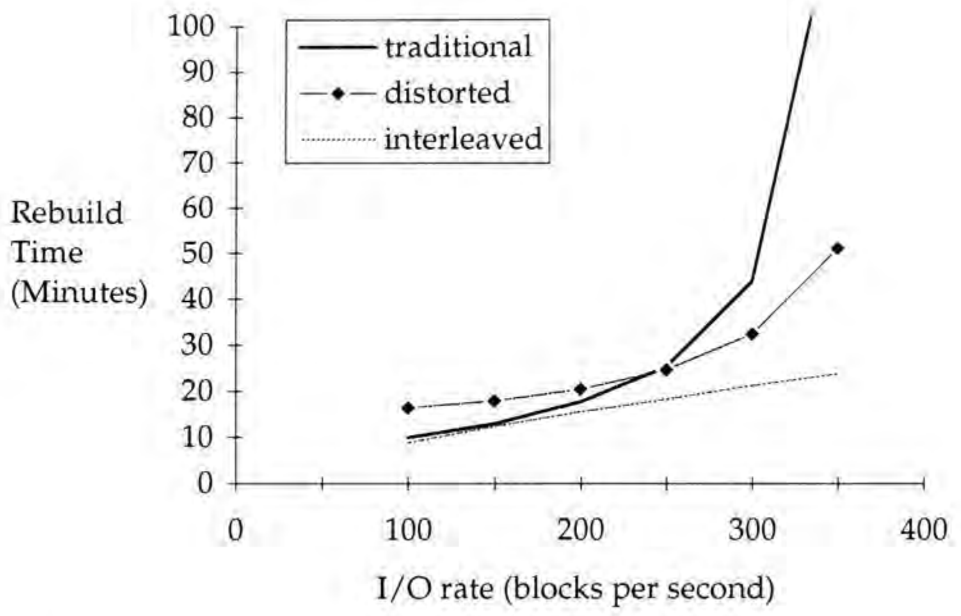

Figure 16: Rebuild Time in minutes vs I/O Rate (75% read: Total time to rebuild failed disk on a replacement disk using block rebuild unit)

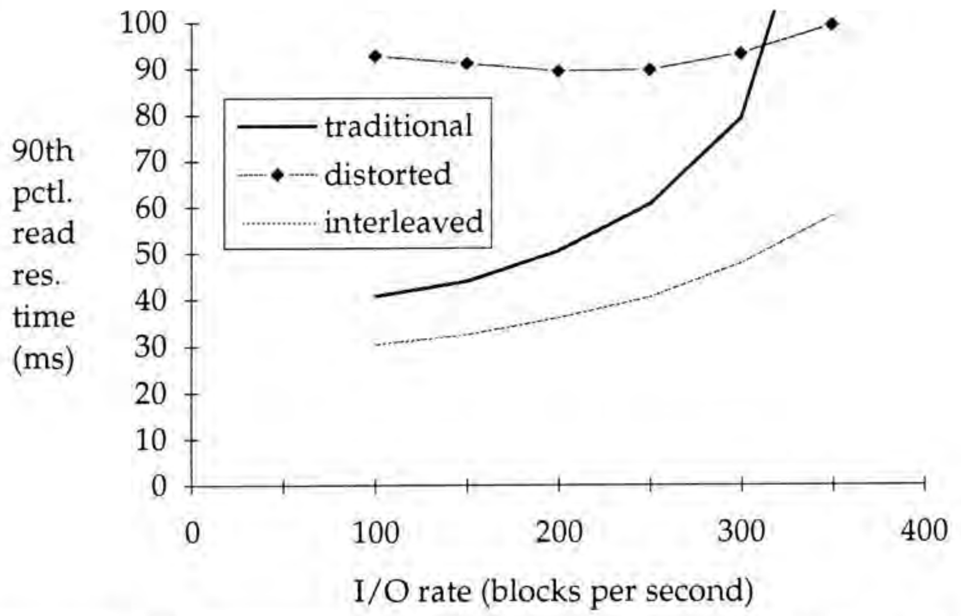

Figure 17: 90th Percentile Read Response Time vs I/O Rate (50% read: The system is in rebuild mode and response time is measured *over all the application read requests. This includes requests that map to the mirror set with a failed disk.* The rebuild unit is track)

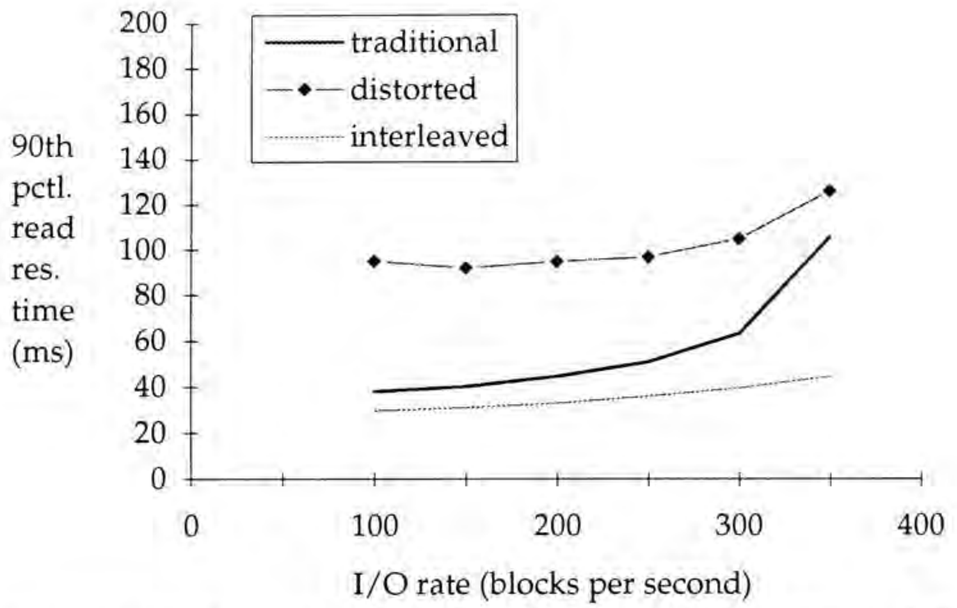

Figure 18: 90th Percentile Read Response Time vs I/O Rate (75% read: The system is in rebuild mode and response time is measured *over all the application read requests. This includes requests that map to the mirror set with a failed disk.* The rebuild unit is track)

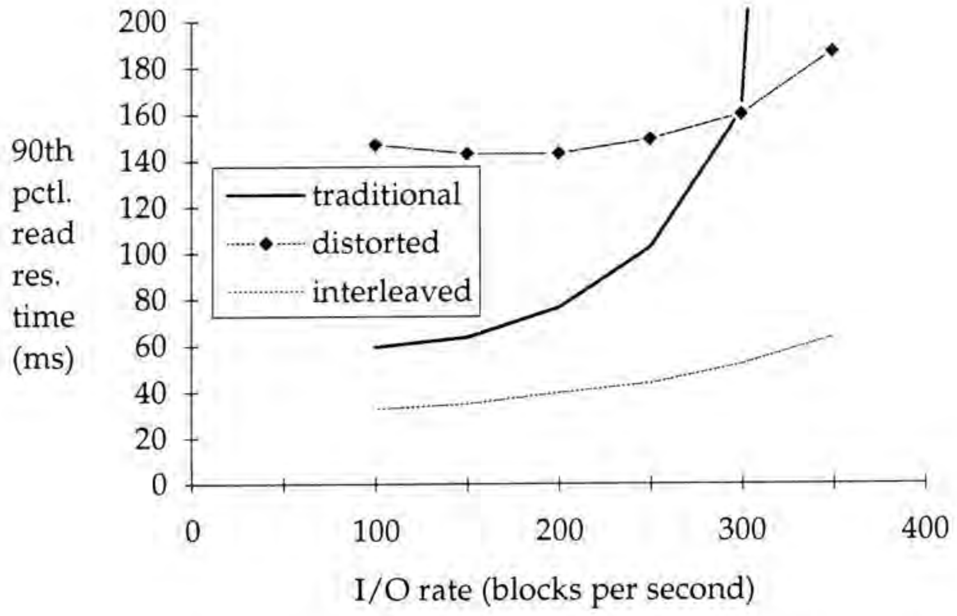

Figure 19: 90th Percentile Read Response Time vs I/O Rate (50% read: The system is in rebuild mode and response time is measured over *only those application read requests that map to the mirror set with a failed disk.* The rebuild unit is track)

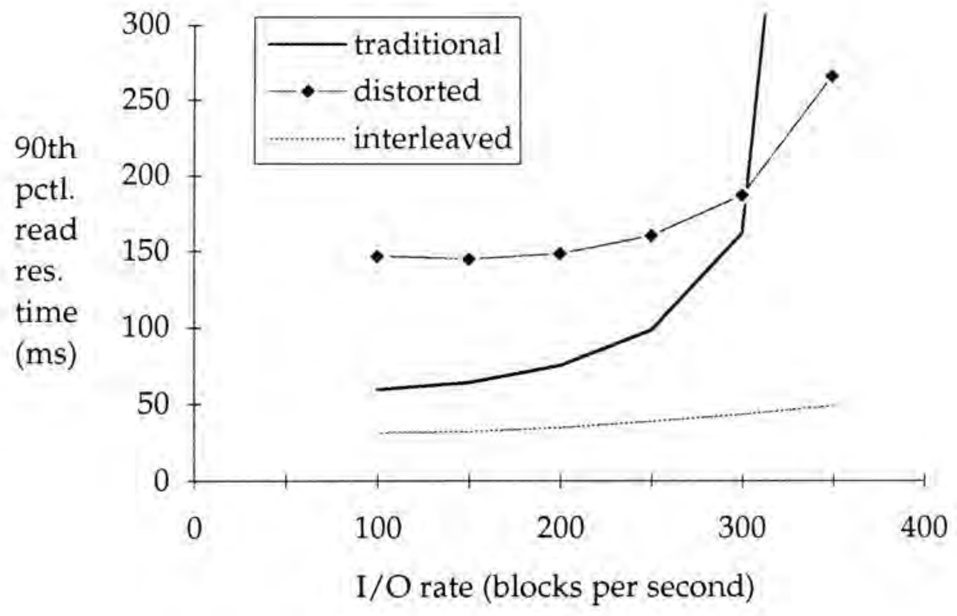

Figure 20: 90th Percentile Read Response Time vs I/O Rate (75% read: The system is in rebuild mode and response time is measured *over only those application read requests that map to the mirror set with a failed disk.* The rebuild unit is track)

The graphs for track rebuild unit are shown in Figures 17, 18, 19, 20 and 21 while the graphs for the cylinder rebuild unit are shown in Figures 22, 23, 24, 25 and 26. The two sets of graphs have identical patterns. Considering the response time curves (Figures 18 and 23), we note that at low I/O rates (below 300 I/Os per second), distorted mirror has the worst performance. This is due to the random disk accesses to rebuild a track or cylinder in the master partition of the replacement disk (see Chapter 2, Section 2.2.2). Our results show that, if response time of application requests is a major consideration, track and cylinder rebuild units are inappropriate in distorted mirror. Interleaved declustering has the best performance of the three schemes. Traditional mirroring has worst

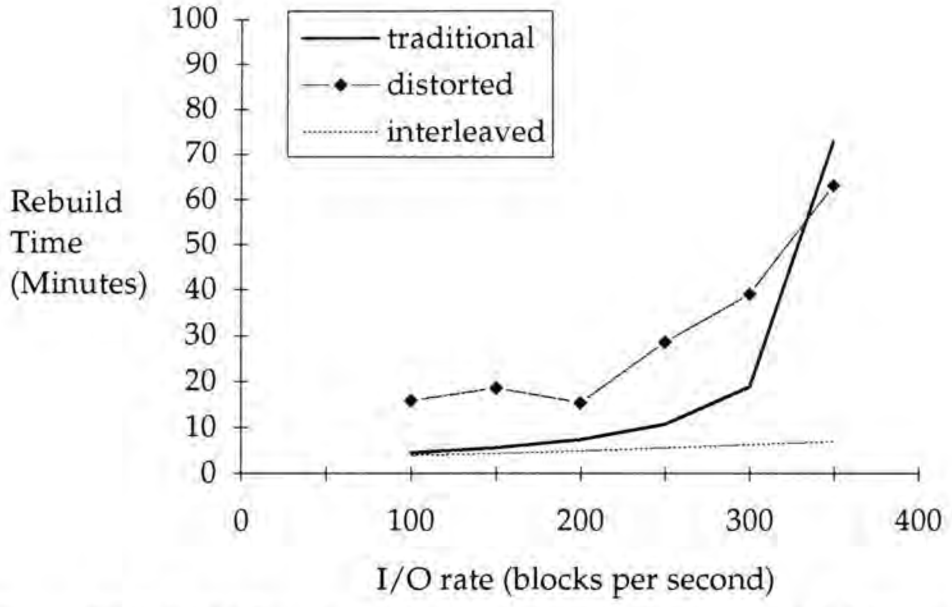

Figure 21: Rebuild Time in minutes vs I/O Rate (75% read: Total time to rebuild failed disk on a replacement disk using track rebuild unit)

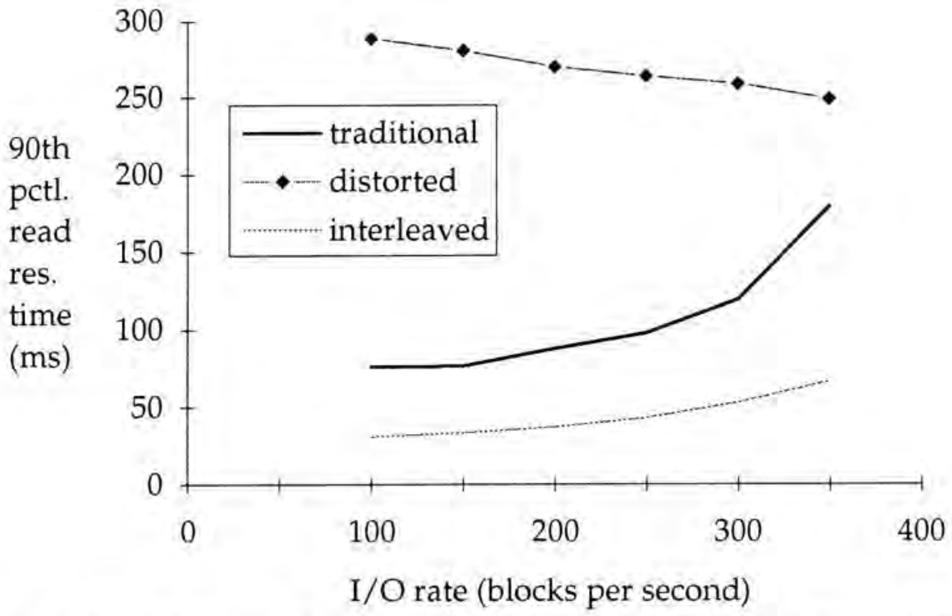

Figure 22: 90th Percentile Read Response Time vs I/O Rate (50% read: The system is in rebuild mode and response time is measured *over all the application read requests. This includes requests that map to the mirror set with a failed disk.* The rebuild unit is cylinder)

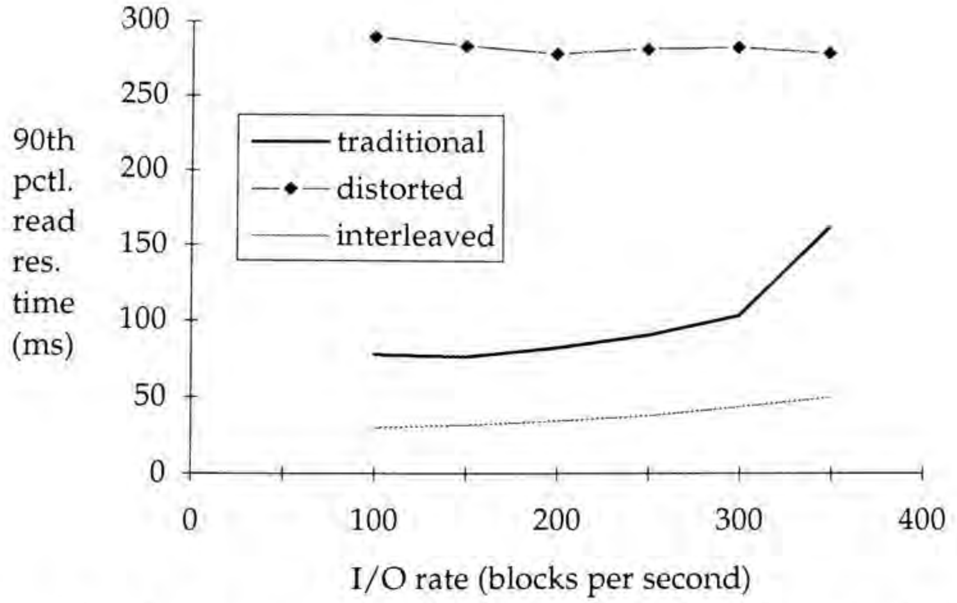

Figure 23: 90th Percentile Read Response Time vs I/O Rate (75% read: The system is in rebuild mode and response time is measured *over all the application read requests. This includes requests that map to the mirror set with a failed disk*. The rebuild unit is cylinder)

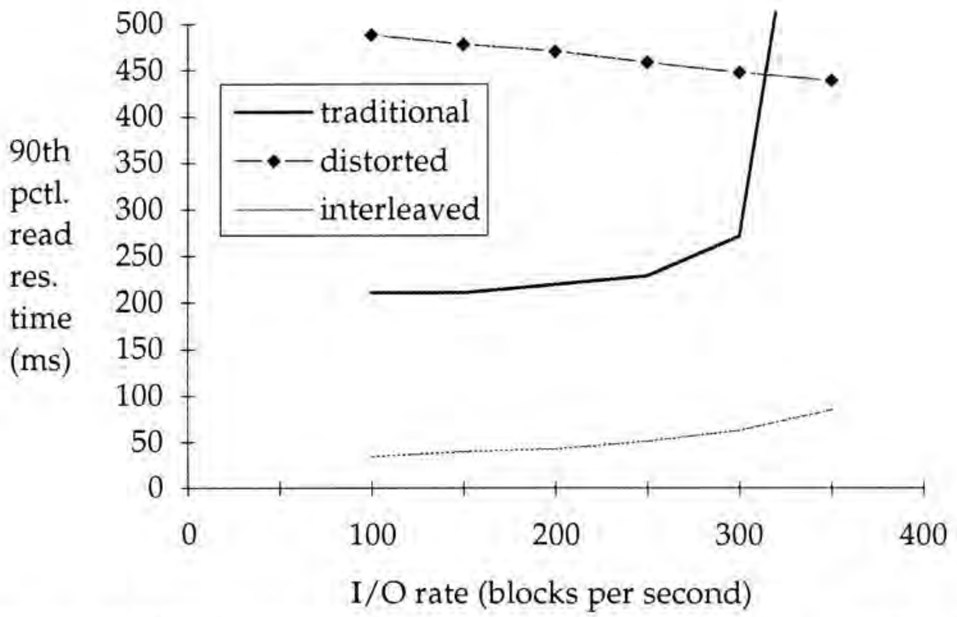

Figure 24: 90th Percentile Read Response Time vs I/O Rate (50% read: The system is in rebuild mode and response time is measured over *only those application read requests that map to the mirror set with a failed disk*. The rebuild unit is cylinder)

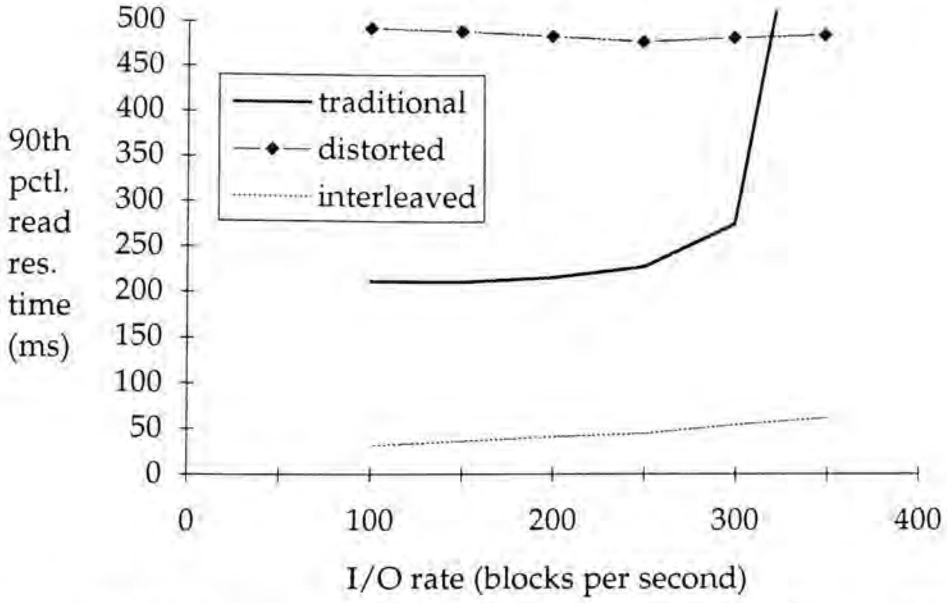

Figure 25: 90th Percentile Read Response Time vs I/O Rate (75% read: The system is in rebuild mode and response time is measured over *only those application read requests that map to the mirror set with a failed disk.* The rebuild unit is cylinder)

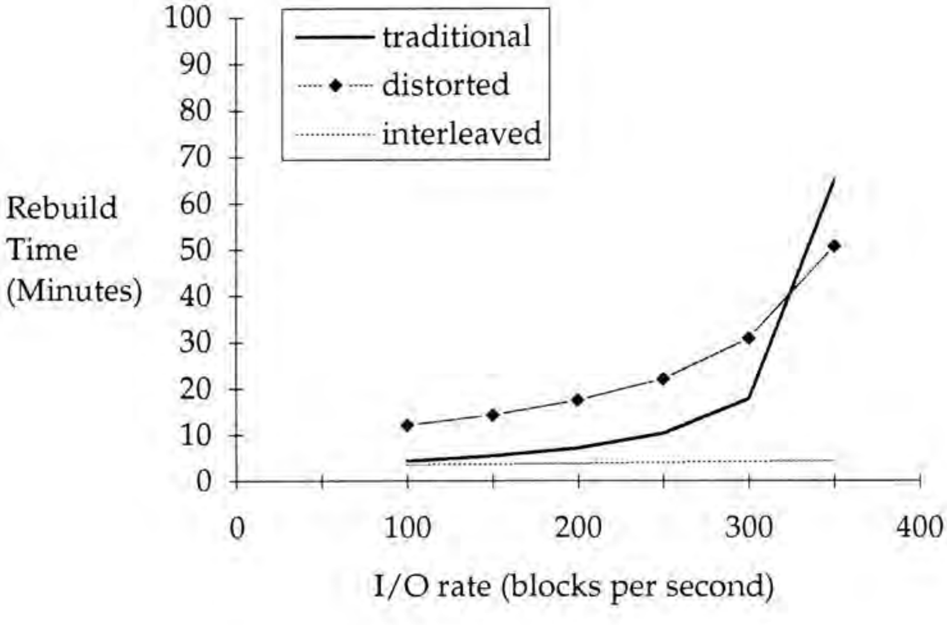

Figure 26: Rebuild Time in minutes vs I/O Rate (75% read: Total time to rebuild failed disk on a replacement disk using cylinder rebuild unit)

performance with high I/O rates.

The rebuild time graphs (Figures 21 and 26) are identical. For almost the entire range of I/O rates simulated, distorted mirror has the worst rebuild time for the same reasons explained above. Interleaved declustering rebuilds fastest. Traditional mirror has reasonable rebuild time at low I/O rates and starts to degrade as I/O rates increase.

In Tables 3 and 4 we summarize the response and rebuild time data for the three schemes. Table 3 shows the results with no redirection of read enhancement to any of the algorithms, while Table 4 shows corresponding results when read redirection is enabled. In general, we conclude from the two tables that interleaved declustering has the best performance of the three schemes. For example, from Table 3, using cylinder rebuild unit, there is only a 19.9% increase in response time for requests that map to the set with a failed disk when compared to the 90th percentile response time for all application read requests. This compares with 161.7% in traditional mirror and 73.4% in distorted mirror.

But it is interesting to compare the two tables. Two points can be made. The first is that distorted mirror benefits most from redirection of read. Since the slave partition can be rebuilt in a relatively short time, there is benefit in making the blocks available for servicing incoming application requests. The second point is similar to the findings of Holland and Gibson [8]; all algorithms do not benefit from redirection of reads. Interleaved declustering was able to balance its workload such that the disks were never saturated, hence there was no need to redirect read requests.

When the tradeoffs between response and rebuild times are considered, we arrive at conclusions similar to those in [10]. Using block rebuild unit

	Traditional Mirroring			**Distorted Mirroring**			**Interleaved Declustering**		
	block	*track*	*cyl*	*block*	*track*	*cyl*	*block*	*track*	*cyl*
90% response time (ms)	42.01	44.60	83.00	39.00	95.00	279.35	33.38	33.02	34.59
max. response time (ms)	68.4	75.50	217.23	70.80	149.23	484.61	34.81	34.63	41.50
rebuild time (minutes)	17.7	7.4	7.1	20.50	15.40	13.50	15.5	4.97	3.83

Table 3: Response and Rebuild Time Data at 200 I/Os Per Second (*75% read. The system is operating in rebuild mode. 90% response time is over all application read requests. Max. response is 90% response time over requests that map to the set with the failed disk. Rebuild time is wall clock time in minutes to rebuild a failed disk. There was no redirection of reads during the rebuild mode; i.e. the replacement disk did not service any incoming application request. The capacity of each disk is approximately 320 mbytes.)*

	Traditional Mirroring			**Distorted Mirroring**			**Interleaved Declustering**		
	block	*track*	*cyl*	*block*	*track*	*cyl*	*block*	*track*	*cyl*
90% response time (ms)	39.10	41.50	74.98	35.26	41.28	66.65	32.80	32.76	34.14
max. response time (ms)	53.77	60.91	210.88	51.07	112.81	393.78	36.50	36.72	47.60
rebuild time (minutes)	16.38	6.81	5.57	19.17	15.49	11.82	16.28	5.2	3.97

Table 4: Response and Rebuild Time Data at 200 I/Os Per Second (*75% read. The system is operating in rebuild mode. 90% response time is over all application read requests. Max. response is 90% response time over requests that map to the set with the failed disk. Rebuild time is wall clock time in minutes to rebuild a failed disk. Read requests are redirected to the replacement disk whenever it is possible and efficient to do so. The capacity of each disk is approximately 320 mbytes.)*

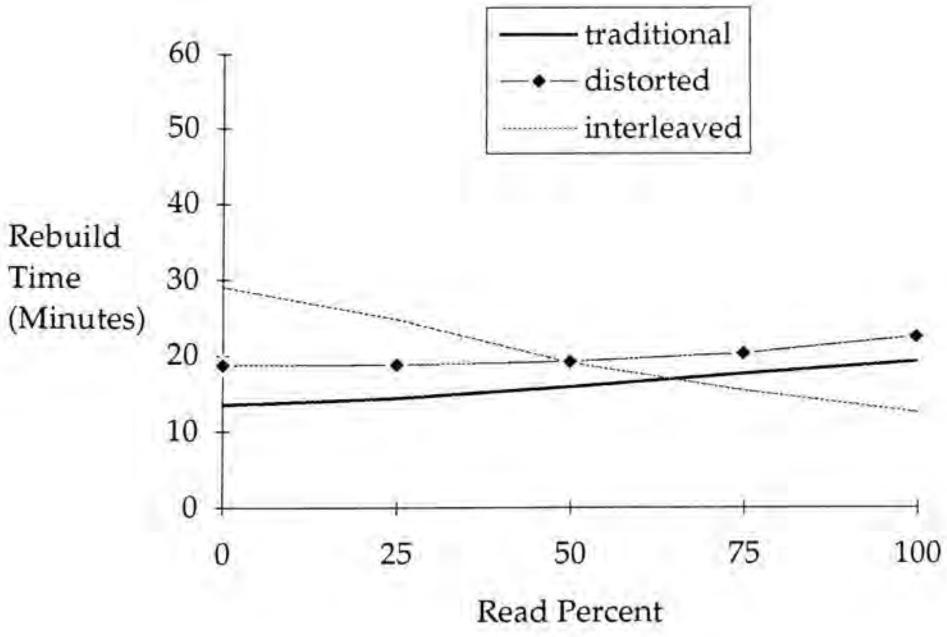

Figure 27: Rebuild Time in minutes vs Read Ratio (I/O rate: 200 per second. Total time to rebuild failed disk on a replacement disk using block rebuild unit)

minimally increases the response time of application read requests. This increase is significant when cylinder rebuild is used. However, cylinder rebuild offers the best performance if rebuild time is the primary consideration. Our experiments show that track rebuild is a good compromise that balances response time and rebuild time demands.

3.5 Other Read/Write Ratios

In this section, we briefly show results that reflect behavior of the schemes at other read/write ratios. We discuss rebuild time data using block, track, and cylinder rebuild unit. We show only data at 200 and 300 I/Os per second.

Figures 27, 28 and 29 show the rebuild time at 200 I/Os per second, while

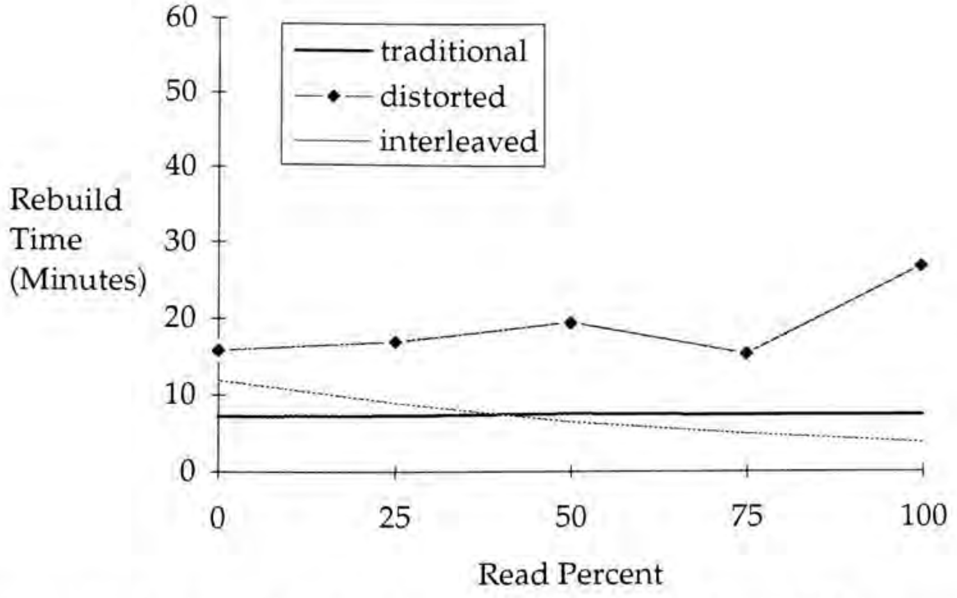

Figure 28: Rebuild Time in minutes vs Read Ratio (I/O rate: 200 per second. Total time to rebuild failed disk on a replacement disk using track rebuild unit)

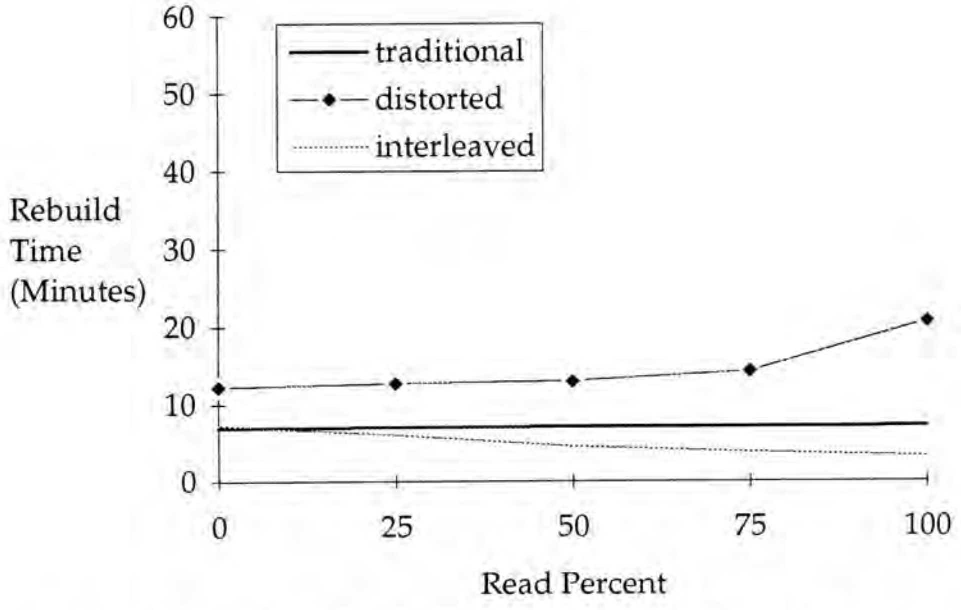

Figure 29: Rebuild Time in minutes vs Read Ratio (I/O rate: 200 per second. Total time to rebuild failed disk on a replacement disk using cylinder rebuild unit)

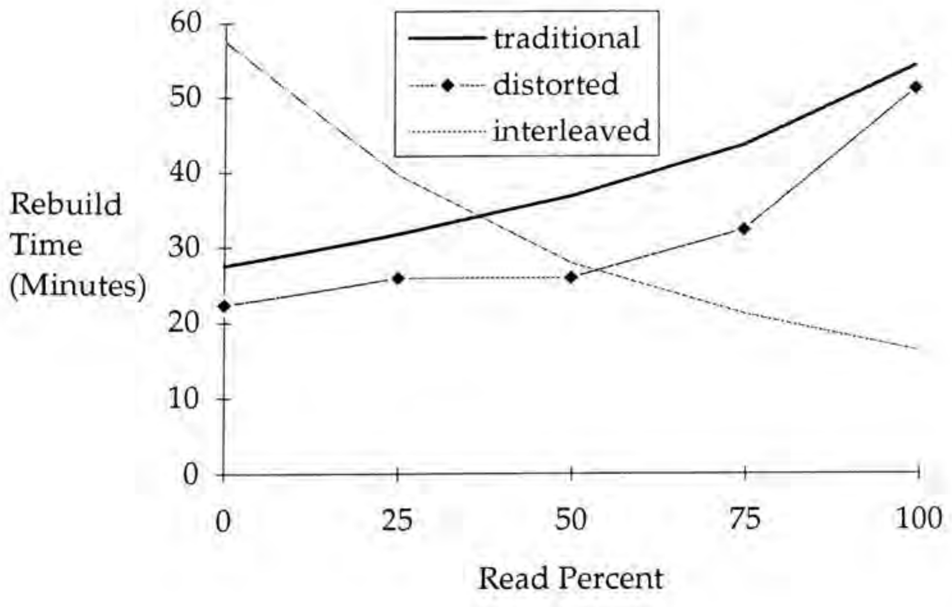

Figure 30: Rebuild Time in minutes vs Read Ratio (I/O rate: 300 per second. Total time to rebuild failed disk on a replacement disk using block rebuild unit)

Figures 30, 31 and 32 show the rebuild time at 300 I/Os per second. The two sets of graphs illustrate two different patterns for the three schemes; traditional and distorted mirrors have identical behavior while interleaved declustering has a different (opposite) behavior.

The behavior of traditional and distorted mirrors is intuitive. An increase in read ratio will place high demands on the bandwidth of the survivor disk since the disk services all read requests. This degrades rebuild performance. Since interleaved declustering distributes the workload of the failed disk over many disks, we expected its performance to be relatively better than the performance of traditional and distorted mirrors. It was, however,

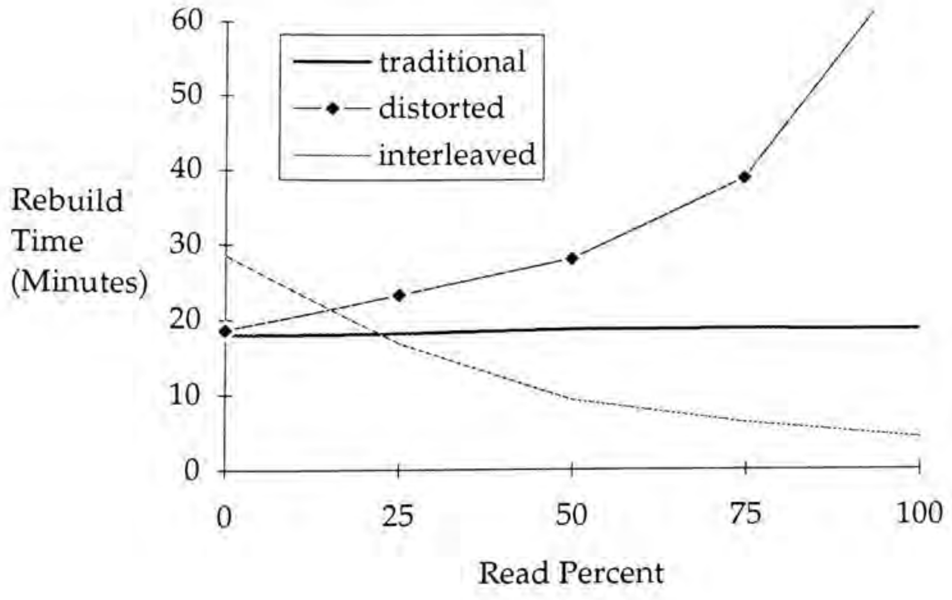

Figure 31: Rebuild Time in minutes vs Read Ratio (I/O rate: 300 per second. Total time to rebuild failed disk on a replacement disk using track rebuild unit)

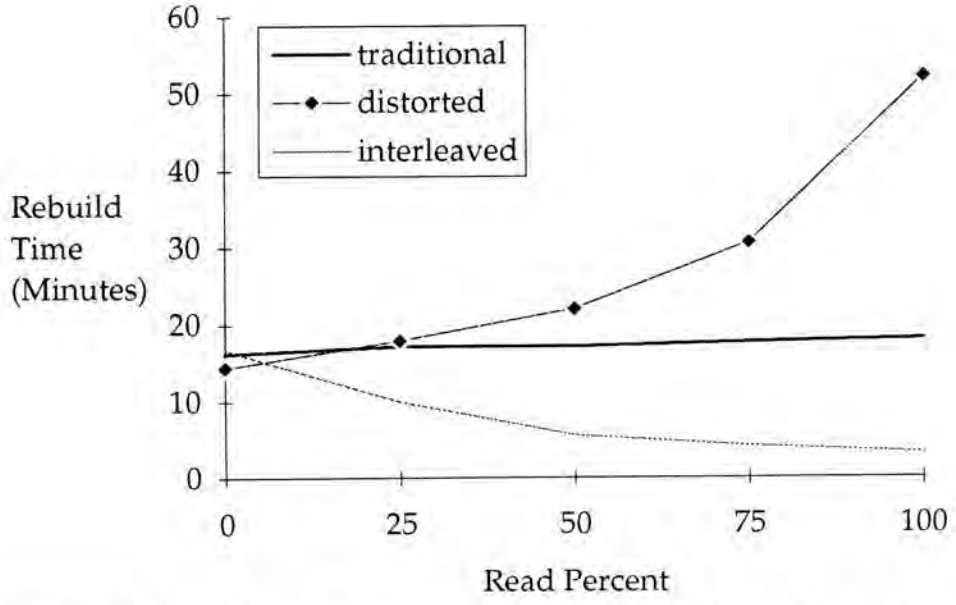

Figure 32: Rebuild Time in minutes vs Read Ratio (I/O rate: 300 per second. Total time to rebuild failed disk on a replacement disk using cylinder rebuild unit)

not immediately clear why the performance of interleaved declustering was worse than corresponding performance for traditional and distorted mirrors at high write (low read) ratios. For example, with 0% read (100% write) and using a block rebuild unit at 200 I/Os per second, the rebuild time in interleaved declustering is 29.21 minutes; in traditional mirror it is 13.50 minutes and in distorted mirror it is 18.71 minutes.

The reason for this behavior can be explained as follows. 16 disks were used in the experiment. For traditional and distorted mirrors, this is equivalent to 8 mirror sets. Thus there is a 1:8 chance that a write request will map to the set with a replacement drive. While the request is being serviced, the drive is unavailable for rebuild operations. However, the 16 disks are divided into 2 clusters in interleaved declustering. Thus there is a 1:2 chance that a write request will map to the cluster with the replacement drive, increasing the probability that the replacement drive will be unavailable for rebuild operation. Thus, the higher the write ratio, the higher the chance that in interleaved declustering, the replacement disk would be unavailable for rebuild operations. We note that at high read ratios, interleaved declustering outperforms traditional and distorted mirrors.

CHAPTER 4
CONCLUSION

In this final chapter, we summarize the major contributions of this thesis and point directions for future research.

4.1 Contributions

In this thesis we have studied three mirroring schemes; traditional mirroring, interleaved declustering and distorted mirroring. Each one of these techniques is able to sustain a single disk failure and provides resiliency to disk failure. However, while each one of these techniques improves the probability of data remaining available in the event of failure, each has significant limitations. While traditional mirroring offers the highest level of availability, it is unable to balance the load in the event of a disk failure. Interleaved declustering provides tradeoffs between availability and performance in the event of a disk failure. As the cluster size is increased, the probability of two failures rendering data unavailable is increased while the imbalance among the disks in the event of a failure decreases. Distorted mirrors suffers from similar drawbacks as traditional mirroring. In addition, it uses more disk space to store data than the other two schemes. However, it is able to perform writes with reduced seek and latency.

We have examined the performance of the three mirroring schemes under three operating conditions — normal, degraded and rebuild modes. Using a synthetic workload that closely models transaction processing environments, we studied various algorithms for rebuilding a failed disk in these mirroring schemes using three rebuild units identified in [10]. Our findings support findings in related studies.

- Because writes are inefficient in traditional mirrors, disks used in this configuration saturate rapidly with increasing I/O rate.

- Interleaved declustering has the best performance for most of the workload studied. However, if the workload is dominated by writes, the performance of interleaved declustering in rebuild mode is worse than the performance of traditional and distorted mirrors.
- Redirection of reads is not beneficial in rebuild algorithm if the disks are not saturated.
- For the 320 mbytes disks simulated in the experiments, the rebuild time was less than 30 minutes with all the rebuild units and configurations studied (at I/O rates of less than 300). As reported in other studies, cylinder rebuild unit provides best performance if rebuild time is of primary concern. The tradeoff is higher response time for application requests. Track rebuild unit provides a good balance between response time and rebuild time demands.

We did not factor in cost considerations in this study. For example, the logical disk in distorted mirror is slightly less than the logical disk in the other schemes studied. In addition, distorted mirror keeps an in-memory data structure that maintains the mapping between master and slave blocks. Interleaved declustering also needs a similar mapping for the backup blocks in a cluster.

4.2 Future Directions

There are many other possible algorithms, and variations to the algorithms studied here, that need to be investigated. We plan to pursue these as future extensions to this study. For example, if cost is a secondary issue and systems can maintain two spare disks on-line, it would be interesting to see how performance would differ when both disks in a mirror set are replaced when one of them fails. But this may not be straightforward (or actually feasible) with interleaved declustering; should all the disks in a cluster be replaced when one of them fails? In the experiments, the size of buffer space was fixed to the size of the rebuild unit. We would like to see how sensitive performance is to variation in buffer size. As suggested else where in this thesis, we suspect that distorted mirror could benefit from a large buffer space during rebuild operations.

References

[1] G.M. Amdahl. Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. In Proceedings AFIPS 1967 Spring Joint, Computer Conference, pages 483-485, Atlantic City, New Jersey, April 1967.

[2] K. Bates, and M. TeGrotenhuis: Shadowing Boosts System Reliability. Computer Design, pages 129-137, April 1985.

[3] D. Bitton, Arm Scheduling in Shadowed Disks. In Proceedings of the IEEE Computer Society International Conference (COMPCON), pages 132-136, San Francisco, California, February 1989.

[4] D. Bitton, and J. Gray. Disk Shadowing. In Proceedings of the International conference on Very Large Data Bases, pages 331-338, Los Angeles, California, September 1988.

[5] J. Chandy and A. Narasimha Reddy. Failure Evaluation of Disk Array Organizations. Technical Report RJ 8706 (78268), IBM March 1992.

[6] D. DeWitt, S. Ghandeharizah, S. Schneider, H. Hsiao, and R. Rasmussem. The Gamma Database Machine Project. IEEE Transactions on Knowledge and Data Engineering, 2, No. 1:44-61, March 1990.

[7] J. Gray, B. Horst, and M. Walker. Parity Striping of Disk Arrays: Low-Cost Reliable Storage with Acceptable Throughput. In Proceeding of the International Conference on Very Large Data Bases, pages 148-161, Brisbane, Australia, 1990.

[8] M. Holland and G. Gibson. Parity Declustering for Continuous Operation in Redundant Disk Arrays. In Proceeding of the Fifth International Conference on Architecture Support for Programming languages and Operating Systems, pages 23-35, October 1992.

[9] R. Hou, Y. Menon, and Y. Patt. Balancing I/O Response Time and Disk Rebuild Time in a RAID5 Disk Array. In Proceeding of the Hawaii International Conference on System Sciences, pages 70-79, 1993.

[10] R. Hou and Y. Patt. Comparing Rebuild Algorithm for Mirrored and RAID5 Disk Arrays. In Proceeding of the International Conference of the ACM SIGMOD, pages 317-326, Washington D.C., May 1993.

[11] H. Hsiao and D. DeWitt. Chained Declustering: A New Availability Strategy for Multiprocessor Database Machines. In Proceeding of the IEEE International Conference on Data Engineering, pages 456-465, Los Angeles, California, February 1990.

[12] M. Kim.: Synchronized Disk Interleaving. In IEEE Transaction on Computers, Vol. C-35, No. 11, November 1986.

[13] J. Menon, D. Mattson, "Performance of Disk Arrays in Transaction Processing Environments", 12th International Conference on Distributed Computing Systems, 1992.

[14] J. Menon, and D. Mattson. Comparison of Sparing Alternatives for Disk Arrays. In the Proceeding of 19th International Symposium on Computer Architecture, pages 318-329, May 1992. Also published as Computer Architecture News, Vol 20, No. 2.

[15] R. Muntz and J. Lui. Performance Analysis of Disk Arrays Under Failure. In Proceeding of International Conference on Very Large Data Bases, pages 162-173, Brisbane, Australia, 1990.

[16] C. Orji and J. Solworth. Doubly Distorted Mirrors. In Proceedings of the international Conference of the ACM SIGMOD, pages 307-316, Washington D.C., May 1993.

[17] C. Orji, Mark A. Weiss, and J. Solworth: Improved Traditional Mirrors. In proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms. pages 329-344, Chicago, Illinois, June 1993.

[18] J. Ousterhout and F. Douglis. Beating the I/O bottleneck: A Case for Log-Structured File Systems. Tech. Rep., Dept. of EECS, Univ. of California, Berkeley, Aug. 1988.

[19] D. A. Patterson, G. Gibson, and R.H. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). In Proceedings of the International Conference of the ACM SIGMOD 1988, pages 109-116, Chicago, Illinois, June1988.

[20] C. Polyzios, A. Bhide, and D. Dias. Disk Mirroring with Alternating Deferred Updates. In Proceeding of the International Conference on Very Large Data Bases, Dublin, Ireland, 1993.

[21] J. Solworth, and C. Orji: Write-Only Disk Caches. In of the proceedings

International Conference of the ACM SIGMOD, pages 123-132, Atlantic City, New Jersey, May 1990.

[22] J. Solworth, and C. Orji: Distorted Mirrors. In First International Conference on Parallel and Distributed Information Systems, pages 10-17, Miami, Florida, December 1991.

[23] J. Solworth and C. Orji. Distorted Mapping Techniques to Improve the Performance of Mirrored Disk Systems. Distributed and Parallel DataBases: An International Journal, 1(1):81-102, 1993.

[24] Tandem. Configuring Disks. Tandem Systems Review, December 1986.

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