Doubly Distorted Mirrors

Cyril U. Orji
School of Computer Science
Florida International University
Miami, Florida 33199
orji@geneva.fiu.edu

Jon A. Solworth
Department of EECS (M/C 154)
University of Illinois at Chicago
851 S. Morgan, Rm 1120 SEO
Chicago, Illinois 60607-7053
solworth@parsys.eecs.uic.edu

Abstract

Traditional mirrored disk systems provide high reliability by multiplexing disks. Performance is improved with parallel reads and shorter read seeks. However, writes must be performed by both disks, limiting performance.

Doubly distorted mirrors increase the number of physical writes per logical write from 2 to 3, but performs logical writes more efficiently. This reduces the cost of a random logical write to 1/3 of the cost of a read. Moreover, much of the write cost can be absorbed in the rotational latency of the reads, performing under certain conditions all the writes for free. Doubly distorted mirrors achieves a 135% performance improvement over traditional mirrors in the TP1 benchmark. Although these techniques require a disk cache for writes, the cache need not be safe nor is recovery time impacted very much.

1 Introduction

Disk mirroring, also called shadowing, is a technique for replicating 1 logical disk across 2 physical disks. The 2 physical disks are called the *mirrored set*, and each disk in a traditional mirrored set contains an identical disk image. By replicating data across different disks, computer systems become practically immune to data loss from disk failures, thereby improving data reliability. Moreover, if multiple controllers and paths are provided to mirrored disk sets, data availability can be improved in the event of disk and/or controller failures, dramatically improving reliability [BT85, Tan86].

Disk mirroring also increases the performance of I/O systems over single disk systems, RAID-5 [PCGK89], and parity striping [GHW90] both in throughput and response time, although at a higher cost. A read can be satisfied by either disk in the mirrored set, doubling read throughput versus single disks.

Bitton and Gray have shown analytically [BG88] and experimentally [Bit89] that mirrors decrease the average seek distance to service a random read. For a disk with n cylinders and using a *nearest arm algorithm*, traditional mirroring read seek distance is $n/5$ cylinders, versus $n/3$ cylinders using a single disk. (Intuitively, the 2 disk arms divide the total disk band into two regions). Hence, mirroring better than doubles read performance.

A major drawback to traditional mirroring is the cost of writes. A write request must be serviced by both disks in the mirror set to maintain data consistency on the disks. Bitton and Gray [BG88, Bit89] have found that the average maximum seek distance to service a write request in a traditional mirror environment is about $0.46n$ cylinders, versus only $0.33n$ cylinders for a write in a single disk system.[1] Hence the logical write rate on both disks of a mirrored system is less than that on a single (non-mirrored) disk.

The relatively poor write performance of mirrors is a problem because:

1. Disk cache sizes continue to increase relative to disk sizes. Since reads can be satisfied from the cache while writes must be written through to the disk for reliability reasons[2], the physical write to read ratio is increasing.

2. The write percentage limits the performance of mirrored systems. Given a write fraction w, the performance improvement is bounded by $1/w$. For example, for 70% writes ($w = .7$), the performance limit is 1.4 times a single disk.

Point one means that w is increasing over time, point two means that as w approaches 1, there is no performance gain over non-replicated systems. We

SIGMOD /5/93/Washington, DC,USA
© 1993 ACM 0-89791-592-5/93/0005/0307...$1.50

[1] The response time measures the longest completion time on any disk. A service time was not derived.

[2] A non-volatile store would enable a write back scenario but this has its own complications.

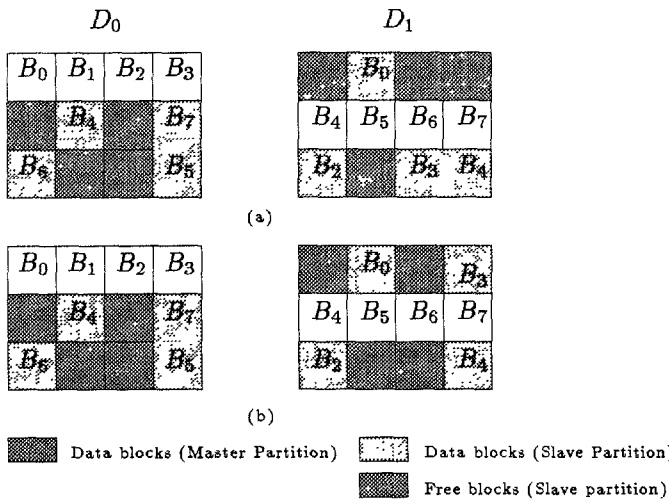

Figure 1: A distorted mirror

conclude that the performance impact of traditional mirrored systems is decreasing.

In a traditional mirror, the cost of a write is over twice that of a read. In this paper, we combine three different techniques to reduce the cost of a random mirrored write to 1/3rd that of a random read. Moreover, we shall show that depending on the read/write mix, the writes can even be performed for free. Although we use a disk cache, that disk cache is constructed out of ordinary RAM, and hence is neither fault tolerant – that is, it need not be replicated – nor must it be nonvolatile.

The Doubly Distorted Mirror use the following techniques:

- Distorted mirrors,
- Write-only disk caches, and
- Write-twice.

Distorted mirrors improved small write performance by almost a factor of 2 over traditional mirrors [SO91, SO92], reducing two random writes to one. Each disk in the distorted mirror set is split into two partitions: the *master* partition which contains the disk blocks in fixed locations, and the *slave* partition which contains the blocks in arbitrary locations. Let M_i (S_i) be the set of logical blocks in the master (slave) partition of disk i. Then $M_0 = S_1$ and $M_1 = S_0$, and $M_0 \cup M_1$ are the set of logical blocks in the mirrored set. A read can be performed on either the slave or master partition. A write must be written to both of its partitions; the master partition write requires a random access while the slave partition write is performed at any free address.

Figure 1a shows a distorted mirror with 8 logical blocks numbered $B_0 ... B_7$, with $B_0 ... B_3$ ($B_4 ... B_7$) mastered on disk D_0 (D_1). A Distorted Mirror write of B_i is written to block i on the disk it is mastered, and to any free block on the disk where it is slaved (at which time the previous slave location for the block is freed). A write of block B_3 overwrites the master block on D_0 (random seek), and writes any free block on D_1, freeing the previous slave copy of B_3, as shown in Figure 1b.

We shall call writes which can be written to any free sector, *write anywhere* blocks; in the language of recovery these blocks are **¬steal** [HR83]. Slave block location are recorded in the distortion map and the free blocks in the free map. To enable the slave blocks to be written without a seek and with very small rotational latency, some 20% of the disk is set aside for free space[3].

Distorted mirrors enables the slave write to be performed cheaply. We now describe how write-only disk caches and write-twice enable the master write to also be performed cheaply.

Write-Only Disk Caches (WODCs) uses a safe-RAM (non-volatile and fault tolerant) to delay writes until they could be performed opportunistically, *piggybacking* them during the rotational latency of reads [SO90]. (If there was insufficient bandwidth to perform these operations during the read rotational latency, cylinders with the most outstanding writes to them could be efficiently purged). WODCs dramatically reduce the cost of writes by eliminating seeks and rotational latency, but require a safe-RAM.

We remove the need for a safe-RAM by writing the block twice; once anywhere there is a free sector (immediately), and the second time where it is supposed to reside (delayed). This system is both write optimized (the first write) and read optimized for future large sequential reads (the second write). Although the block is written twice, neither a seek nor a rotational latency is incurred. Since these two factors account for about 90% of a random access, this savings is substantial. The block becomes safe at the point it is written to both the master and the slave disk, but before it is written in its final location on the master disk. Replicas are often used to improve read performance – this is the first replica use we know of which increases write throughput.

The contributions of doubly distorted mirrors are that

- only a small fraction of the master indices change for updates,
- the distorted map is kept efficiently and explicitly,
- efficient large sequential reads are possible,
- recovery time is modestly impacted,

[3] In a ten surface disk, this means that there is an average of 2 free blocks per sector. Since rotational latency and seek time typically consume over 90% of a random access, paying only the transfer time makes the second write almost free.

- small writes are performed very cheaply (in some cases for free), and
- cache can be volatile.

The above techniques are described in the context of mirrored systems, but apply to other types of disk organizations, for example chained declustering [HD90]. The rest of the paper is organized as follows. Section 2 describes the distorted mirror algorithm. In section 3, the simulation parameters are described and section 4 provides a performance comparison of doubly distorted mirrors versus both traditional mirrors and distorted mirrors. Section 5 describes recovery after system failure. Related work is described in section 6, and we conclude the paper is section 7.

2 The doubly distorted mirror

The doubly distorted mirror is described in two parts. The first describes the logical and physical address spaces, and the mappings between them. The second describes the algorithm for doubly distorted mirrors.

2.1 Logical and physical address spaces

In a doubly distorted mirror set, the logical address space is divided into two partitions, P_0 and P_1. The physical disk space is divided into 4 partitions; on disk i ($i = 0, 1$) there is a master partition, M_i, and a slave partition and S_{1-i}. Each of P_0, P_1, M_0, and M_1 are the same size while the partitions S_0 and S_1 are larger since they contain many free blocks to aid in the write anywhere operations. A block in logical partition P_i is said to be *mastered* in physical partition M_i on disk i. Blocks are mapped into the master partition in sequential order, while blocks in the slave partition are write anywhere.

There are two maps between the logical address space and the physical address space, the implicit map and the distortion or explicit map. The implicit map is an invariant mapping of logical to physical addresses; the explicit map is a dynamic mapping which is explicitly kept of logical to physical addresses.

A write is written to the slave partition of each disk, and is later written to its master partition.

2.1.1 Implicit map

The implicit map maps the logical address space partitions 1-1 with the master partitions; moreover the master partitions are kept in sequential order. The implicit map is as follows:

$$(P_i - X) \quad \rightarrow \quad M_i \qquad i = 0, 1 \tag{1}$$

Where X is the set of (write-twice) blocks in the disk cache which have been backed up to the disk containing the master partition, but not written in the master partition.

We have constructed each master partition to contain a contiguous range of logical block addresses, enabling almost all large transfers to be performed efficiently on a single disk[4], and that each cylinder is divided into master and slave partitions. Given the disk parameter, up to 100 blocks can be read continuously before performing a *twitch* (adjacent cylinder seek), supporting efficiently large sequential reads.

2.1.2 Explicit map

The explicit or distortion map keeps track of the currently live, write anywhere blocks, which are always written to the slave partition. The size of this map for the mirrored set is the size of the logical address space plus the size of the cache. The distortion or explicit map is as follows:

$$((X \cap P_i) \cup P_{1-i}) \quad \rightarrow \quad S_{1-i} \qquad i = 0, 1 \tag{2}$$

The slave partition is larger than a logical partition, so that there are many unused disk blocks which can be used for *write anywhere* blocks. In our simulations, we have used 20% free space. Figure 2 shows the logical and physical address spaces in a distorted mirror. These blocks in the free store are tracked in the free map, so that write anywhere sectors could be rapidly located.

2.2 Algorithm

The read and write scheduling algorithms are shown in Figure 3. We describe in detail writes; reads are only interesting in that they perform part of the work of the write.

Consider writing block B_3, which is mastered on D_0. Figure 2(a) shows the mirrors before the write. When the write arrives, it is immediately written anywhere to the slave partitions of both disks, as shown in Figure 2(b) as well as cached. Sometime later, B_3 is written to its fixed location in the master partition on D_0 and the D_0's slave copy of B_3 is freed (Figure 2(c).

The program is shown in Figure 3. When a write is requested, it is enqueued to be written to each of the slave partitions, and is also put in the cache. The physical slave writes are then delayed until one of the following conditions hold:

- A physical read is requested, at which time queued writes are piggybacked onto the read (see below).

[4] An alternative is to use a finer interleaving, for example mapping odd numbered blocks to one partition and even numbered blocks to another; this increases bandwidth for a single request large request since the request can be performed on multiple disks but also increases overheads.

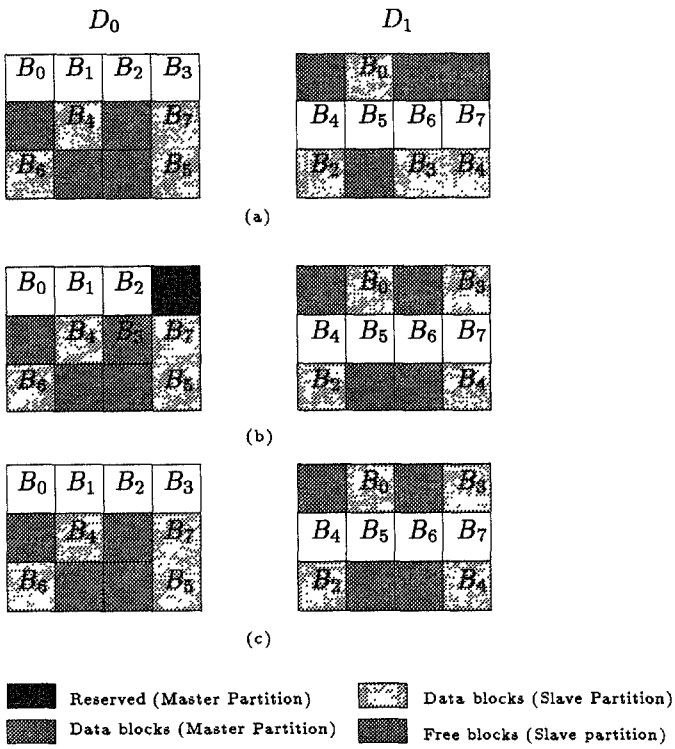

Figure 2: Doubly Distorted Mirror

```
write(addr, block)
    for i= 0..1 do
        if disk[i].writeq.full() then
            disk[i].writeq.purge();
        endif;
        disk[i].writeq.enqueue(addr, block);
    endfor;
    Cache(addr, block);
endproc;

read(addr, block)
    if addr ∈ cache then
        return(CacheBlock(addr));
    endif;
    let d be the disk that the read is scheduled on
    l = disk[d].scheduleRead(addr, block);
    disk[d].schedulePiggybacks(cylinder[l]);
    disk[d].scheduleWriteAnywhere(l);
    perform writes, recording which blocks written
    which free block, and update the distortion maps
endproc;
```

Figure 3: Doubly distorted mirror algorithm

- The queue is full, then a write anywhere purge occurs, writing out all the blocks in the queue to free sectors in the current cylinder.
- A checkpoint occurs, in which case the transactions unwritten blocks in the queue are sent to disk[5].

By far, the most common action is the first one, since there is usually sufficient bandwidth to piggyback almost all of the writes on the reads.

The physical master writes are cached and delayed until one of the following conditions holds:

- A read is requested to the cylinder containing the block in the master partition. The write is then piggybacked on the read.
- The cache fills up, in which case the cylinder with the most dirty blocks is purged, or written out[6].

When a read is requested, the algorithm checks whether the block is in the cache, in which case no physical I/O is necessary. Otherwise, the read is scheduled by nearest free arm algorithm. In addition to scheduling the read, any dirty blocks in the cache which implicitly map to that cylinder and which will be passed during the rotational latency are scheduled. After the implicit mapped blocks are scheduled, the write anywhere blocks queued by the write() are scheduled to any free sector passed[7]. Each write anywhere block is recorded in the distorted map. Dirty blocks may not be removed from the cache until all of the associated writes complete[8].

Doubly distorted mirrors uses a log to record changes to the distortion map. We have also assumed in the TP1 benchmark that changes to the data were also logged as in a traditional mirrored system, to reduce response time — although this may not be necessary in a real system. Because of the continuous slave writes, the amount of data log that is relevant is much shorter resulting in faster recoveries. In fact, one advantage to doubly distorted mirrors is that the system is continuously checkpointed (using the write anywhere queues), and hence periodic checkpoints disappear. In any event, using group commit [DKO+84, Gra79] log writes are so efficient that their performance impact is small.

[5]Because of the continuous writing of blocks, checkpointing will have much fewer blocks to write (on the order of the length of the write anywhere queue.) Moreover, no seeks are required.

[6]Using a write-only disk cache of size c, with b blocks per cylinder, the expected number of blocks written is $2bc$, and the write is quite efficient. For more details, see [SO90]

[7]If the write anywhere queue is over half full, extra rotational beyond the read point is incurred, if needed, to keep the physical writes in balance with the physical reads.

[8]This guarantees that the latest value of a block is either in the cache or has been totally written to disk. Hence the remaining reads may be performed before queued writes.

2.3 Memory data structures

Two in-memory data structures are required for distorted mirrors:

FreeMap This is a bit map, with 1 bit for each non-master block in the mirrored set.

DistortedMap A map whose domain is logical locations, and whose range is the address space of the distorted map.

The free map is needed to find legal places to write slave blocks. The distorted map is used to schedule reads and to update the free map. There is a distorted map per slave partition. When logical disk block l is to be written, the following sequence of actions occur:

1. Select a free block f from the FreeMap
2. Release DistortedMap[l] when update committed
3. DistortedMap[l] is set to f

Because we use a write ahead log for fixed location writes, and because distorted writes do not overwrite valid data, there is no need for undo information in the log [HR83]. The main memory overhead of these structures is measured as a percentage of disk space.

The FreeMap requires one bit per slave block. Since we assume 4K bytes/block (32K bits), the ratio is 1:32K or about .003% is needed for the FreeMap. The DistortedMap is a little larger, about .05% of total buffer space[9].

These numbers are small compared to current disk cache (disk buffer space) which is on the order of 1–2% and hence these maps easily fit in main memory.

3 Simulation model

We wrote a simulator to compare the expected performance of the distorted mirrors, doubly distorted mirrors, and the traditional mirror. We summarize the disk parameters in Table 1.

The workload is characterized by the types of requests (whether a read or a write), the size of the request in blocks and the disk full factor. The request size in blocks will characterize disk performance in terms of random and sequential access performance [Tur88]. I/O requests are externally triggered and arrive at the controller in a FIFO order. The request characteristics fit both OLTP [A+85] and UNIX file system [ODH+85].

No of Cylinders	1000
No of Surfaces	10
Disk Full Factor	80%
Blocks per Track	10
Average Latency	8.33 ms
Average Seek	13.50 ms

Table 1: **Disk Parameters**

The workload parameters can be described by the read/write mix and request size. The read/write mixes simulated were 0%, 30%, 50%, and 70%, read. The request size was scaled from 1 block to 256 blocks in powers of two, modeling very small (4Kb) to large sequential transfers (1Mb). The disk full factor used in the experiment was 80%.

In each simulation, the request size is kept constant and simply grows by powers of two for each run. The requests used in the experiments are uniformly distributed over the number of tracks in the disk. 20,000 requests are serviced in each run of the experiment, except for doubly distorted mirrors in which 200,000 requests were needed to reach steady state on the cache.

Finally the I/O system is modeled as a single-queue, single-server queuing system with an events scheduler and controller. The disk controller keeps mapped information on the disk blocks and determines which of the disks in the mirror set will service a request. The scheduler arranges the requests in the order they will be serviced and forwards the request to the appropriate disk. Read requests are scheduled by shortest seek time if both arms are free.

When not otherwise specified, we have used a 2% disk cache for doubly distorted mirrors.

4 Results

In this section the results obtained in simulating the traditional, distorted and doubly distorted mirror schemes are presented. First the effect on single block writes is examined. Then two important metrics for measuring disk system performance are discussed; service and response time. For doubly distorted mirrors, there are two ranges of transfer sizes; small sizes in which doubly distorted techniques are used and large sizes in which distorted mirrors techniques are used. Finally, doubly distorted mirrors performance is discussed for the TP1 benchmark.

[9] The DistortedMap is represented as two data structures, the direct DistortedMap which maps locations l to physical addresses on disks which they are not mastered, and the hashed DistortedMap which maps addresses which have been updated on the mater disk in the slave partition but not the master partition. The hashed distorted map is very small since it is proportional to the disk cache size. The computation for the direct distorted map follows: Four bytes are sufficient to represent 4×10^9 blocks in a mirrored set (which could contain 16×10^{12} bytes) well in excess of current needs. This is .1% overhead of main memory. Since the map only exists for distorted map entries, we get a further factor of almost 2 reduction to .05% overhead.

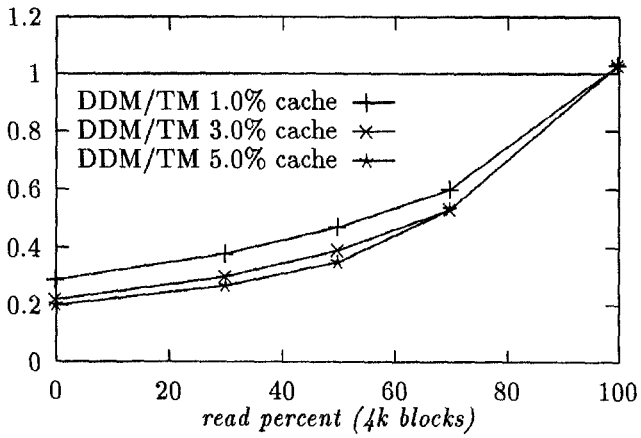

Figure 4: **Service time ratio** of DDM vs. TM for single block requests

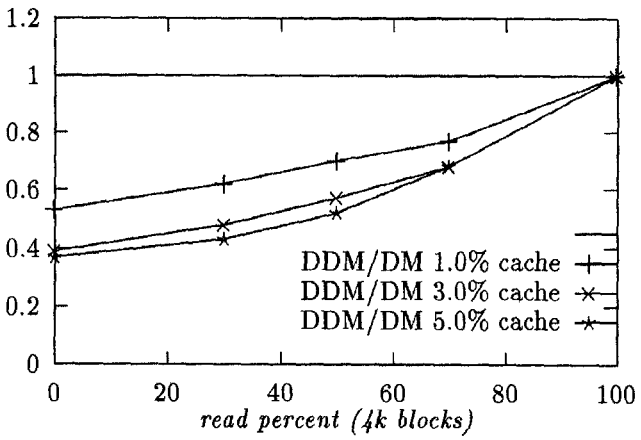

Figure 5: **Service time ratio** of DDM vs. DM for single block requests

4.1 Single block write time

In this section we compare the effect of random writes on doubly distorted mirrors (DDMs) vs. those of traditional mirrors (TMs) and distorted mirrors (DMs).

The request sizes in blocks are shown on the horizontal axis while on the vertical axis we show the ratio of the service times of the DDM to those of the TM (DM). Values on the vertical axis are obtained as the ratio service times. The horizontal solid line at the vertical distance 1.0 represents the normalized values for the TM (DM). Therefore, regions of the graph under this horizontal line show where the DDM scheme outperforms the TM (DM). Above this line, the TM (DM) shows better performance.

In Figures 4 and 5 the effect of DDMs is compare to TMs and DMs for various read/write ratios. The service time is the average cost per single-block I/O.

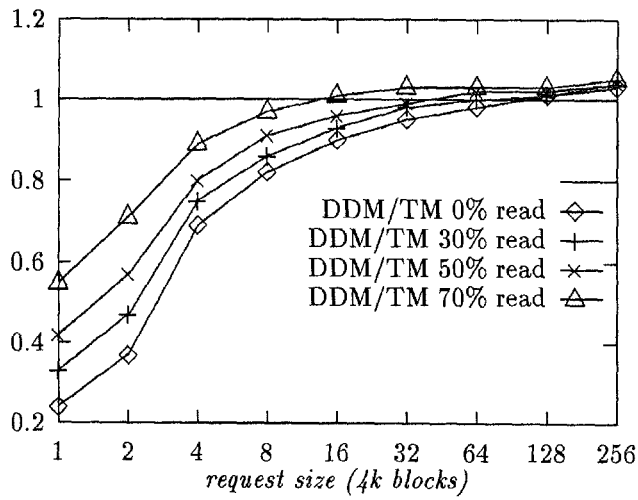

Figure 6: **Service time ratio** of DDM to TM

DDMs achieve up to a 5 fold performance improvement over TMs and up to a 2.5 fold improvement over DMs. On single block requests, DDMs outperform both DMs and TMs, except at 100% reads.

4.2 Service time

One of the metrics for evaluating the performance of the I/O systems is the average time for servicing a request. In Figure 6, semi-log graphs show the service times ratios for the DDM against corresponding values for the TM for various read/write ratios. These graphs assume a 2% write cache.

There are two distinct size ranges in these experiments, small and large block transfers. For small transfers from 1 to about 16 blocks (4K to 64K bytes), DDMs are superior to TMs. Beyond this transfer size, TM is better. The advantages of DDMs for small writes are:

1. With sufficiently large caches, DDM master write cost is zero, since these costs are incurred by the read. With smaller caches, only some of the blocks are piggybacked: the rest must be purged and this requires a seek, although in general a single seek time can be amortized over multiple writes.

2. DDM slave writes are also performed very cheaply. Almost all slave writes are piggybacked on the rotational latency of reads. This effect does not depend on cache size since these blocks can be written anywhere; it depends only on the read-write ratio and the number of blocks per track (which determines maximum piggybacking).

Large block transfers incur a penalty of up to 4.5% over traditional mirrors since:

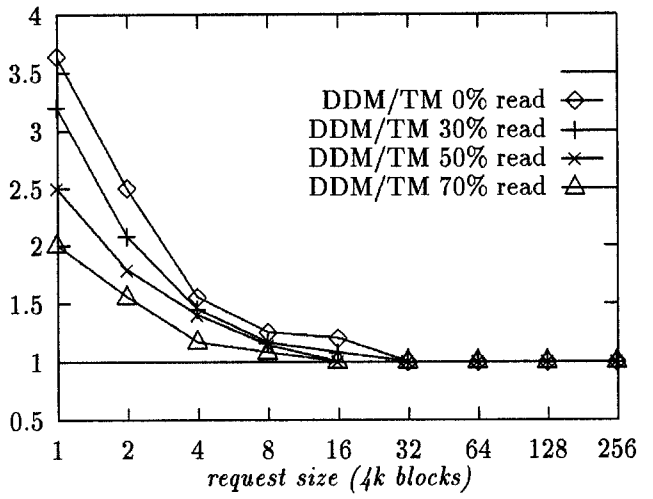

Figure 7: **Throughput ratio** of DDM vs. TM

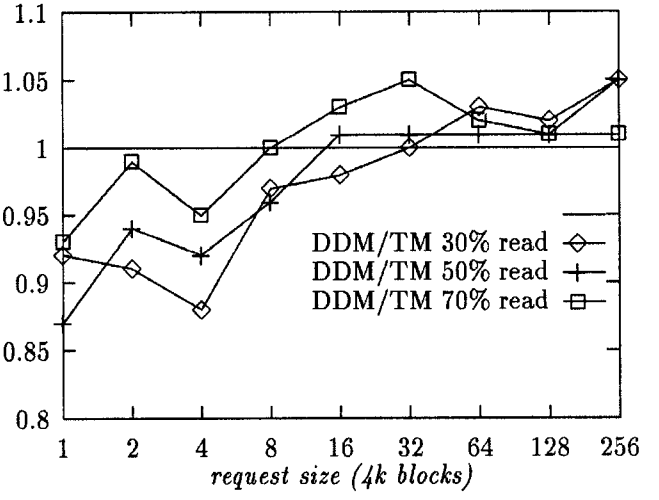

Figure 8: **Read response time ratio** of DDM vs. TM

1. Multi-block (sequential) read requests are scheduled on the master disk, the disk arms do not segregate the disk band into disjoint bands. Hence large reads seek further in the DDM.

2. For large block transfers, the DDM master disk must do over twice the track-to-track seeks as the TM.

3. The DDM slave disk has at least as many twitches as the master; it also has reduced transfer rate due to lack of free blocks within sectors.

Nevertheless, large block writes are so efficient that for most workloads performance is completely determined by small writes.

4.3 Throughput

Another way of looking at the response time is as its throughput or I/Os per second (IOPS) at 50% disk utilization. For write intensive applications, DDM can support higher loads for a given disk utilization. The relative throughput graphs for a 50% disk utilization are shown in Figures 7. Figure 7 shows that for one block requests at 0% read, the DDM throughput is about 3.75 times that of the TM. This advantage tails off for multi-block requests and as the read ratio increases.

4.4 Response time

The response time, or time it takes for an operation to complete depend on both the queueing time and on the maximum of the two service times. We separate read response time from write response time, since an application is only delayed by read response time. Recovery, which is effected by write response time is the same as in traditional systems, since it is controlled by the write response time of the log. DDM read response time varies from 13% better to 5% worse against TM. The response time graphs are shown in Figures 8. The DDM read response time is better than TM when the request size is less than 16.

Write response time is several times larger than read response time and depends on the write anywhere queue length. Given the maximum queue length of 20 used in the simulations, the average queue length is half that, or 10 (see Section 2.2). Assuming 50% read, and hence 2 slave writes per each master write, write response time is about 5 times that of read response – on the order of 100ms, and hence is quite small. In the next section, which considers the TP1 benchmark a data log is used so that commit response time is independent of write response time.

At any rate, we believe that read response time differences of this magnitude are not meaningful to the application, and that the large increases in throughput more than compensates by reducing queuing time.

4.5 TP1 benchmark

In the previous section, we discussed relative performance. In this section we discuss performance of mirroring on the TP1 benchmark.

The average disk throughput can be computed given a read/write mix and a mix of block sizes. Then the throughput is:

$$blocks/second = \frac{1}{\sum_b R_b/T_b}$$

where R_b is the number of b-block transfers on average per transaction and T_b is the throughput for that transfer at the appropriate read/write mix. Consider the effect of the Datamation TP benchmark [A+85]. In the Tandem system, this benchmark results in a 50%

read percent	measure	TM	DM	DDM
50%	IOPS	58	85	139
50%	TPS	28	41	66
30%	IOPS	51	82	155
30%	TPS	34	55	101

Table 2: Random I/O per second (IOPS) and transactions per second (TPS) of mirrored sets

read/write mix, 1 physical read and 1 physical write are needed per transaction, plus .4 log writes [The88]. DDM's 139 IOPS (I/Os per second) achieves an 140% improvement over TM's 58 IOPS, and a 64% improvement over DM's 85 IOPS (see Table 2). In terms of TP1 rates, DDM's 66 TPS is an 135% improvement over TM's 28 TPS, and a 60% improvement over TM's 28 TPS. The slight decline from IOPS improvements to TPS improvement is due to the log writes, which essentially are performed at the same rate in all mirroring techniques — but it is random I/O performance which dominates TP1.

If the disk cache is large enough to hold those active accounts against which half the transactions apply, then the read percentage drops to 33%[10]. The transaction rate improve with all three mirroring techniques. More interesting is that the relative advantage of DDM increases. On TP1, the DDM improvement reaches 197% over TM and 83% improvement over DM.

5 System recovery

We consider two different failures and their effect on system recovery:

1. Controller failure
2. Disk failure

In the case of controller failure, the data on each disk is not corrupted but the DistortedMap and FreeMap are lost and must be reconstructed. In addition, the loss of the disk cache places an extra (and perhaps heavy) load on the system. The loss of the disk cache, however has the same effect in all three mirrored systems and hence is a neutral issue.

If a disk fails, then the map remains intact and hence each block (slave or master) is directly accessible on the remaining disk. Given this disk, a spare disk must be reconstructed from the remaining mirrored disk to restore mirroring protection. This is more expensive in DM and DDM since the distortion map means that a direct disk-to-disk copy cannot be used.

5.1 Controller failure

If a safe memory is associated with the controller which will store the disk map, then a controller failure does not result in lost mapping information. This may not be sufficient for all applications since the controller reliability must be sufficiently large so that the map is not corrupted by other controller failures.

Assuming that controller failure is possible, we describe an incremental checkpoint recovery technique which writes out the DistortedMap periodically and journals the changes between checkpoints. (The FreeMap can be constructed directly from the DistortedMap). Hence, each journal entry contains the logical disk block number l, the old distorted map entry for l, and the new distorted map entry for l.

If the DistortedMap is checkpointed every 5 minutes, and writes occur at the rate of 200 per second, then there will be a maximum of 120,000 journal entries[11]. Each of these entries is 12 bytes for a total journal size of 1,440,000 bytes, resulting in a restoration time of less than a second. The periodic checkpoints would also require less than a second.

Given the FreeMap and the DistortedMap, the master blocks written to the distorted partition can be determined. These master blocks can be (after recovery) read in and then written back to their fixed locations. These reads may be piggybacked on other reads, and hence can be performed at low cost.

5.2 Disk failures

A disk failure represents a more serious loss of performance than a controller failure. First, slightly more than half the bandwidth of the disk system has been lost. Second, a spare disk must be restored which will use up additional bandwidth reading the remaining disk drive. Nonetheless, since the distorted map is memory resident, there is no loss of data availability.

One technique which can significantly reduce this second cost is disk caching. In very large systems, containing hundreds of mirrored disk sets each with a 1 or 2 percent cache, it may be possible to temporarily borrow .5% cache from each disk to entirely cache the disk in main memory. The time to restore the memory is the time to read in all the blocks from the main disk, and to do a sequential write of the secondary disk.

Even if only a small disk cache is available, a significant performance advantage can be gained. For

[10] Note that this may be significantly less than half the total accounts, since some accounts will have disproportionate number of transactions posted against them

[11] Note that the data is continuously checkpointed by the write anywhere queues, and hence don't require a large flush during checkpoints.

example if c is the fraction of disk cached, and b is the number of blocks per cylinder, approximately $2bc$ blocks can be written at a time [SO90]. For a doubly distorted mirror with a 80% disk full factor, $b = 100$ and $c = .02$, an average of 4 blocks can be written per disk access as the slave portion of the surviving disk is reorganized into a master disk. In addition, 4 master to slave blocks could be piggybacked on the master reorganization enabling 8 blocks to be written per random track write, which takes 30ms. Hence, the 80,000 blocks to be copied would require 300,000ms or 300 seconds, a 77% increase over the 170 seconds to copy a traditional mirror which was 80% full.

As disk caches continue to increase in size, the copy differential decreases, narrowing the cost of recovery for distortion techniques.

6 Related work

Disk mirroring is not the only technique for increasing reliability and availability of data. We shall review several other techniques which have been proposed. These techniques differ in the reliability provided, storage overhead, and performance.

The distortion techniques are applicable not only to mirrors, but to any organization which replicates writes — rather than using parity. For example, interleaved declustering [Ter85], chained declustering [HD90] as used in the GAMMA database machine [DGS+90], as well as the declustering in the Bubba database machine [CABK88] .

However, RAID systems [PCGK89] and Parity striping [GHW90] requires the (parity) writes to occur in fixed making it unclear how to apply distortion techniques, and increasing the cost of multiple writes.

The distorted map techniques dramatically decrease the cost of small writes by a *write-anywhere* policy of writing data blocks to any free sector. This is not a new idea, and is used extensively in the UNIX file system. For example in the Berkeley 4.3 file system [LMKQ88], data blocks are written anywhere although directory and indici blocks (called inodes) are in constrained locations. In the log-structured file system (LFS) [RO91], the constraint on inode and directory blocks are removed as this information is copied over.

IBM has a mirrored UNIX filesystems[MO90]. In this case, the file system is replicated but not duplicated. However, this method does not allow the traditional advantages of explicit disk management for database systems with its efficient large block transfers. Instead, in distorted mirrors, some blocks are mapped to variable locations and some to fixed locations. This combination of UNIX filesystem "write anywhere" with traditional database management semantics, blends the advantages of both techniques.

7 Conclusions

Doubly distorted mirror provides an effective method for performing both small block reads and writes as well as large sequential transfers. In particular, for doubly distorted mirrors we have shown that:

- Single block writes improve by a factor of 3.
- Single block writes can often be piggybacked onto reads for free, effectively reducing the cost of doing I/O to just the cost of the cache-miss reads.
- TP1 benchmark improves by 135% over traditional mirrors, and 60% over distorted mirrors.
- Large block transfers are only slightly less efficient than in traditional mirrors.

Doubly distorted mirroring increases in importance as technological advances bring larger disk caches. Moreover, with large disk caches, the write ratio increases improving the relative advantage of doubly distorted mirrors — and this is the traditional mirror's achillies heel.

We believe that distortion techniques complement clustering techniques, and when blended together give the best of both worlds. This is a topic for future research.

Distorted mirrors also have the advantage that they are application independent. A distorted mirror system can replace a traditional mirrored system without application changes.

Acknowledgements

Part of this work was supported by NSF Grant CCR-9208631. The authors thank the referees for their many useful comments.

References

[A+85] Anonymous et al. A Measure of Transaction Processing Power. *Datamation*, 31. No. 7:112–118, April 1985.

[BG88] 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.

[Bit89] 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.

[BT85] K. Bates and M. TeGrotenhuis. Shadowing Boosts System Reliability. *Computer Design*, April 1985.

[CABK88] G. Copeland, W. Alexander, E. Boughter, and T. Keller. Data Placement In Bubba. In *Proceedings of the International Conference of the ACM SIGMOD*, pages 99 - 108, Chicago, Illinois, June 1988.

[DGS+90] D. DeWitt, S. Ghandeharizah, S. Schneider, H. Hsiao, and R. Rasmussen. The Gamma Database Machine Project. *IEEE Transactions on Knowledge and Data Engineering*, 2, No. 1:44–61, March 1990.

[DKO+84] D. DeWitt, R. Katz, F. Olken, L. Shapiro, M. Stonebraker, and D. Wood. Implementation Techniques for Main Memory Database Systems. In *Proceedings of the International Conference of the ACM SIGMOD*, pages 1–8, Boston, Ma., June 1984.

[GHW90] J. Gray, B. Horst, and M. Walker. Parity Striping of Disc Arrays: Low-Cost Reliable Storage with Acceptable Throughput. Technical Report 90.2 Part Number 39596, Tandem Computers, Inc., 1990.

[Gra79] J. Gray. Notes on Data Base Operating Systems. In *Operating Systems, An Advanced Course*, pages 393–481. Springer-Verlag, 1979.

[HD90] H. Hsiao and D. DeWitt. Chained Declustering: A New Availability Strategy for Multiprocessor Database Machines. In *Proceedings of the IEEE International Conference on Data Engineering*, pages 456–465, Los Angeles, California, February 1990.

[HR83] Theo Haerder and Andreas Reuter. Principles of transaction-oriented database recovery. *ACM Computing Surveys*, 15(4):287–317, December 1983.

[LMKQ88] S. Leffler, M. McKusick, M. Karels, and J. Quarterman. *The Design and Implementation of the 4.3BSD UNIX Operating System.* Addison Wesley, Reading, Massachusetts, 1988.

[MO90] J. M. Mott and J. C. O'Quin. Auxiliary Storage Management in the AIX Operating System. IBM Risc System/6000 Manual, 1990.

[ODH+85] J. Ousterhout, H. Da Costa, D. Harrison, J. Kunze, M. Kupfer, and J. Thompson. A Trace-Driven Analysis of the UNIX 4.2 BSD File System. In *Proceedings of the Symposium on Operating Systems Principles*, pages 15–24, December 1985.

[PCGK89] D. Patterson, P. Chen, G. Gibson, and R. Katz. Introduction to Redundant Arrays of Inexpensive Disks (RAID). In *Proceedings of the IEEE Computer Society International Conference (COMPCON)*, pages 112–117, San Francisco, California, February 1989.

[RO91] M. Rosenblum and J. Ousterhout. The LFS Storage Manager. *Proceedings of the Summer USENIX Conference*, pages 315 – 324, June 1991.

[SO90] Jon A. Solworth and Cyril U. Orji. Write-Only Disk Caches. In *Proceedings of the International Conference of the ACM SIGMOD*, pages 123–132, Atlantic City, New Jersey, May 1990.

[SO91] Jon A. Solworth and Cyril Orji. Distorted mirrors. In *Parallel and Distributed Information Systems*, pages 10–17. IEEE/ACM, December 1991.

[SO92] Jon A. Solworth and Cyril Orji. Distorted mapping techniques to improve the performance of mirrored disk systems. *Distributed and Parallel Databases: An International Journal*, 1992. to appear.

[Tan86] Tandem. Configuring Disks. *Tandem Systems Review*, December 1986.

[Ter85] Teradata Corporation. *DBC/1012 Database Computer System Manual Release 2.0, Document No. C10-0001-02*, November 1985.

[The88] The Tandem Database Group. NonStop SQL, A Distributed, High-Performance, High-Availability Implementation of SQL. In *Proceedings of the International Conference of the ACM SIGMOD*, pages 337 - 341, Chicago, Illinois, June 1988.

[Tur88] C. Turbyfill. Disk Performance and Access Patterns for Mixed Database Workloads. *Database Engineering*, 11 No. 1:48–54, March 1988.