MimdRAID: Low Latency Secondary Storage*

Randolph Y. Wang† Thomas E. Anderson‡ Yuqun Chen†
Arvind Krishnamurthy§ Kai Li† Xiang Yu†

Abstract

Disk arrays can provide scalable bandwidth with increasing number of disks, but they do not address latency. In this paper, we study how to improve secondary storage read latency by systematically increasing the ratio between disk heads and usable capacity. The most direct technique is to build "faster" disks by altering disk geometry. In addition, as an alternative to building such disks, we also describe how we can emulate these faster disks using multiple conventional disks. We show that all these techniques are governed by a common principle that the overhead-independent part of the latency improves by a factor of the square root of the amount of extra resources. We evaluate some of these latency reduction techniques on a prototype implementation which we call MimdRAID. On a 12-disk system, for random reads, we achieve a 2× improvement in latency and 21× improvement in throughput compared to a single drive. This is 80% to 95% better than the throughput achieved on a conventional mirrored system.

1 Introduction

In this paper, we set out to answer a simple question: how do we minimize the latency of small reads to disk?

Although considerable effort in disk array research [5, 21, 30] has resulted in vast bandwidth improvement, low latency remains an elusive goal. More recently, novel logging techniques promise to dramatically lower the write latency [4, 29]. However, high read latency remains a major weakness of secondary storage systems.

*This work is supported in part by the Scalable I/O project under the DARPA grant DABT63-94-C-0049 and by the National Science Foundation under grant CDA-9624099.

†Department of Computer Science, Princeton University, {rywang,yuqun,li,xyu}@cs.princeton.edu.

‡Department of Computer Science and Engineering, University of Washington, Seattle, tom@cs.washington.edu.

§Department of Computer Science, Yale University, arvind@cs.yale.edu.

The performance of small synchronous disk reads impacts the performance of important applications such as persistent object stores [14] and database applications [27, 28]. There are three existing approaches to improve read latency: careful data placement exemplified by Unix FFS [19] and reorganization [18], aggressive prefetching such as "transparent informed prefetching" [3, 13, 22], and aggressive caching such as "cooperative caching" [6, 7]. These techniques work best when one can accurately predict the access pattern, either statically or dynamically. We are interested in the complementary question of how to improve performance in the absence of accurate predictions.

1.1 Technology Trends

The gist of our technique is systematically trading off capacity for lower latency. Several technology trends have simultaneously enabled and necessitated this approach.

The key trend is the phenomenal growth of disk areal density. At an annual rate of 60% [11], this growth has resulted in a dramatic improvement in disk capacity, bandwidth, and cost.

This trend has several implications. Since disk latency has been improving at only 10% per year [17], disks are becoming increasingly unbalanced in terms of the relationship between capacity and latency. A similar phenomena is happening to the memory subsystem [23]. To address this imbalance, Wood and Hill proposed the so called *Converse Amdahl's Dictum*: each megabyte of memory should be accompanied by one MIPS of processing power [31]. In other words, we can achieve better cost/performance by increasing the processing to memory capacity ratio. We believe that a similar insight applies to secondary storage systems: instead of blindly conforming to form factors, we can achieve a more balanced system by increasing the disk head to capacity ratio.

Database vendors today have already recognized the importance of building a balanced secondary storage system. For example, in order to achieve high performance on TPCC [28], vendors configure

systems based on the number of disk heads instead of capacity. To achieve $D\times$ the bandwidth, the heads form a D-way mirror, a D-way stripe, or a combination of the two (as was done in Petal [15]). What is not well understood is how we can systematically translate the excess capacity into improved latency, which can in turn lead to improved throughput, reducing the number of heads required.

Another important technology trend is the relationship between memory and disks. The areal density of memory has been improving at a steady rate of 40% per year [11], a rate that is far behind that of the growth in disk density (60%), which has benefited significantly from recent innovations in disk sensing and recording technology. This disparity also manifests itself in terms of cost per megabyte: the gap between memory and disk is roughly two orders of magnitude today and growing.

A naive answer to the disk read latency problem is that it will be addressed in time by the growing memory caches. However, file system implementors are consistently surprised by the low cache hit rate despite the increasing amounts of memory [1]. We believe that the widening gap between memory and disk costs can only make this problem worse. The upside of this growing disparity is that it presents an opportunity for the development of a range of storage alternatives that can fill the gap between memory and disk in terms of cost/performance.

1.2 Background

A number of recent RAID systems are designed with the intent to study the tradeoff between capacity and performance. We highlight two such systems that have inspired our work, while deferring the rest of the related work to a later section.

Petal is a network RAID that exports virtual disk interfaces to network clients [15]. To provide better small write performance, better load balance, and easier crash recovery, Petal chooses mirroring instead of a RAID-5 organization. The HP AutoRAID system makes the tradeoff between capacity and performance more explicit by incorporating both mirroring and RAID-5 into a two-level hierarchy [30]. The mirrored upper level provides advantages similar to those of Petal at the expense of consuming more storage, while the RAID-5 lower level is more frugal in its use of disk space.

We believe that these systems represent steps in the right direction in their explicit recognition of the tradeoff between capacity and performance. However, a large number of questions remain unanswered. The primary advantage of mirroring exploited by these systems is its low *write* latency; how can we extract low *read* latency by exploiting this tradeoff? Are there techniques other than mirroring that can accomplish this goal? Why do we have to stop at two copies? How can we build better disks that are more latency-friendly? What is the relationship amongst these different latency reduction techniques? In short, we believe Petal and AutoRAID represent two interesting design points in a vast design space that beckons a systematic exploration.

1.3 Contributions

MimdRAID is a low latency secondary storage system that systematically trades off capacity for performance. During its design, we have made the following contributions. We have designed a number of latency reduction techniques that exploit excess capacity. Two of these, which we call *splitting* and *replicated splitting*, offer throughput and simplicity advantages that traditional mirroring does not possess. We have identified and quantified how disk geometry can be altered to improve disk latency. We show that all our latency reduction techniques share a simple rule-of-thumb, which we call the *Square Root Rule:* speedup of the overhead-independent part of the latency equals $\sqrt{x}$, where x can be any one of the following: the number of excess disks, the reduction of platter area, or the number of heads per surface. By combining splitting and rotational replication, we can achieve comparable latency improvement as mirroring with far fewer replicas, while delivering super-linear scale-up in throughput as we increase the number of disks.

To evaluate some of these latency reduction techniques, we have implemented a prototype MimdRAID system. In the process, we have developed a calibration method that provides accurate prediction of disk head positions, a crucial requirement for most of our latency reduction techniques. On a 12-disk system, we achieve a $2\times$ improvement in latency and $21\times$ improvement in throughput for random small reads.

The remainder of the paper is organized as follows. Section 2 presents a range of latency reduction techniques. We provide simple analytical models and analyze the advantages and disadvantages of each approach. Section 3 validates the Square Root Rule using simulations. The simulations shed light on cases that are too complex to model analytically or cases that are impossible to implement. Section 4 describes our prototype MimdRAID implementation. We explain how we perform disk head position calibration. We compare the latency and

throughput results of a range of strategies. Section 5 describes some of the related work. Section 6 concludes.

2 Latency Reduction Techniques

In a well-constructed RAID system, large I/O bandwidth scales linearly as we increase the number of disks. What is not well understood is how latency improves in response to the addition of resources. There are three main contributing factors to disk latency: seek delay, rotational delay, and overhead. In this section we consider the first two factors. We would like to understand, for example, how much improvement one can expect if we double the number of disks, halve the diameter of disk platters, or double the number of heads per surface. We use simple analytical models to study how to best take advantage of the extra resources and evaluate the different alternatives.

2.1 Using Extra Disks

Disks are cheap. The simplest form of introducing extra resources into the system is to add more disks. The challenge is how to effectively turn extra disks into low read latency. Mirroring systems such as Petal [15] and AutoRAID [30] represent an important data point. In this section, we explore alternatives other than simple mirroring and how the latency improvement scales as we increase the number of extra disks beyond two. We first explore a number of techniques that reduce seek distance; then we study how to reduce rotational delay; and finally, we combine the seek and rotation reduction techniques.

2.1.1 Techniques for Reducing Seek Distance

We start by defining the following abstract problem statement: suppose the maximum seek distance on a single disk is S, the total amount of data in question fits on a single disk, but we are given D disks. Suppose we perform random seeks. The challenge is to devise different ways the D disks can be effectively employed to reduce seek distance. We would like to know the average seek distance of these alternatives. We use seek distance to simplify our presentation. Of course, seek latency is not a simple linear function of seek distance [25]. We examine seek latency in later sections.

As a base case, it can be shown that the average seek distance for random reads on a single disk S_0 is[1]:

$$S_0 = \frac{S}{3} \tag{1}$$

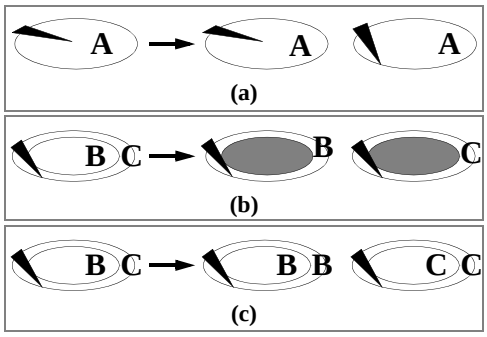

Figure 1: Techniques for reducing seek distance. Capital letters represent a portion of the data. To the left of the arrows, we show how data is (logically) stored on a single disk. To the right, we show different ways that data on the single disk can be distributed on two disks: (a) mirroring, (b) splitting, and (c) replicated splitting.

Mirroring

Mirroring can reduce seek distance because we can choose the disk head that is closest to the target sector in terms of seek distance (shown in Figure 1(a)). Assuming that we do not enforce identical layout on the D replicas, we can prove that the average seek distance on a D-way mirror S_m is:

$$S_m = \frac{S}{2D+1} \tag{2}$$

We can achieve slightly better result than Equation (2) if we insist that all replicas have identical layout. Such a requirement, however, would prohibit write optimization techniques that rely on data location independence.

Splitting

Unlike mirroring, *splitting* involves no data replication. Figure 1(b) illustrates a *two-way split*. Data on the original single disk is partitioned into two disjoint sets: B and C. We store B on the outer edge of the first disk and C on the outer edge of the second disk. The space in the middle of these two disks is simply discarded! As a result, the disk heads are restricted within a smaller region. This is called disk splitting because the single large disk is in effect split into two smaller disks. Discarding space in the middle of the disk is a key feature that distinguishes splitting from other data distribution

[1] In the interest of brevity, we omit all the proof details in this paper. Interested readers can find them in the technical report.

techniques. Otherwise, splitting is identical to striping. We can prove that the average seek distance S_s is:

$$S_s < \frac{S}{3D} \quad (3)$$

Note that we have used the inequality "<" in Formula (3). This is because in reality, the outer tracks on a disk store more data than the inner ones. As a result, the difference between S_s and S_0 should be a factor that is larger than D.

Splitting by itself does not provide reliability in the face of disk failures. There are two ways of addressing this issue. The first is a combination of mirroring and splitting: when $D > 2$, we replicate the data *once* while splitting $D - 1$ ways. This is based on the observation that it usually does not take D copies to provide sufficient protection; two are sufficient. The second possible solution to reliability is to use the center of the disks to store *secondary* copies, copies that exist solely for reliability considerations. For example, in Figure 1(b), we will store a copy of C in the center of the first disk and a copy of B in the center of the second disk. This is analogous to mirroring in that each disk contains a complete replica of *all* data. The disadvantage of this approach is that during periods of mixed reads and writes, the writes will force some of the disk heads into the secondary copies at the center of disks, thus degrading performance of reads to the primary copies on these disks. We therefore assume the first approach.

It is easy to see that splitting is strictly better than mirroring. One reason is that splitting should offer better latency than mirroring. The difference between S_m of Equation (2) and S_s of Formula (3) increases as we increase the number of disks D.

Perhaps the even more important reason to prefer splitting to mirroring is that splitting provides better write throughput. In order to deliver the full benefit of seek distance reduction using D mirrored disks, we must perform D physical writes for every single logical write. Of course, some of these writes can propagate in the background; but reads to data whose propagation has not completed cannot enjoy the full benefit of latency reduction offered by a D-way mirror. A high degree of replication also complicates implementation due to the effort in maintaining consistency of the replicas. For splitting, only one (if reliability is not a concern) or two (if reliability is a concern) writes need to occur, so latency improvement is achieved without sacrificing write throughput.

Replicated Splitting

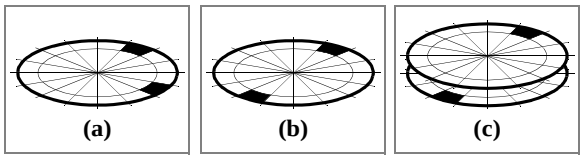

Figure 2: Techniques for reducing rotational delay. (a) Randomly placed replicas. (b) Evenly spaced replicas. (c) Replicas placed on different tracks.

Splitting, as discussed in the last section, reduces seek distance by not using the center of the disks. Our next technique uses this space to further improve latency. Under *replicated splitting*, as shown in Figure 1(c), we replicate the data *within* each disk a number of times. Assuming constant track capacity, and depending on the number of copies we make (which is less than D), we can prove that the average seek distance of replicated splitting S_{rs} falls within the following bound:

$$\frac{S}{4D} < S_{rs} < \frac{S}{3D} \quad (4)$$

The upper bound is simply S_s. The lower bound requires more explanation. Intuitively, replicated splitting can further reduce seek distance beyond splitting because for most of the time, there is an identical copy of the target sector on each side of the disk head so that the head can choose the closer one. As we increase the number of copies, the average seek distance quickly approaches the lower bound when there is always a copy in either direction. In practice, we have found that making more than two copies provides virtually no additional benefit.

As disk drives acquire more intelligence in the future, it is possible that such propagations can be controlled by the embedded controller inside the drive without outside intervention. Such an optimization can take advantage of the "free" bandwidth between the head and the platters without consuming valuable interconnect bandwidth and RAID controller processing. Unlike this *intra-drive propagation*, traditional mirroring relies on *inter-drive propagation*, which cannot avoid crossing the interconnect between disks.

2.1.2 Techniques for Reducing Rotational Delay

Reducing seek distance alone is not sufficient. Another important contributing factor to latency is rotational delay. The general approach is to replicate data at different rotational positions. By choosing a replica that is rotationally closer than others, this approach has the effect of emulating a disk that

spins faster. Of course, this assumes that the rotational position of the disk head is available to the entity that chooses the target sector. Replication for reducing rotational delay can of course increase seek distance by pushing data farther apart. We ignore this effect for the time being in this section.

In the following discussions, we assume that the full rotational delay on a single disk is R and we replicate data D times. As a base case, the average rotational delay on a single disk R_0 is simply half of a full rotation:

$$R_0 = \frac{R}{2} \tag{5}$$

Random Rotational Replication

A naive approach is to place replicas at random rotational positions. There are several scenarios under which this arrangement can occur. One is a mirrored system whose spindles are not synchronized. Another is when we schedule the writes of different replicas at different times at "convenient" but random rotational positions either on a single disk (shown in Figure 2(a)) or on different disks. Using D rotational replicas, we can prove that the average rotational delay R_r to the first such replica is:

$$R_r = \frac{R}{D+1} \tag{6}$$

It is easy to see why *random rotational replication* may not be the best way of placing the extra copies. For example, in Figure 2(a), the two random replicas partition a track into two differently sized arcs. The probability of the disk head's falling in the larger arc is larger than in the smaller one. Therefore, the probability of incurring the longer delay is also larger.

Even Rotational Replication

To address the shortcoming of random rotational replication, under *even rotational replication*, we place the D replicas $360/D$ degrees apart from each other. This can be done on a mirrored system whose spindles are synchronized or on a single disk when we carefully place the replicas (shown in Figure 2(b)). We can prove that the average rotational delay R_e under this approach is:

$$R_e = \frac{R}{2D} \tag{7}$$

Intuitively, R_e is simply scaling down R_0 linearly by a factor of D. This is so because even rotational replication effectively shortens the length of a track. For a larger replication factor D, the difference between random and even rotational replication is almost a factor of two.

Rotational Replication on Different Tracks

Figures 2(a) and (b) have illustrated the concept of rotational replication by making copies within the same track. Unfortunately, this decreases the bandwidth of large I/O. For example, under even rotational replication (shown in Figure 2(b)), during large sequential reads, we are forced to switch tracks D times more frequently than without replication. Today, on a state-of-art disk, a track switch is roughly 0.5-1 ms while the full rotational delay is about 6 ms. Under these conditions, a high degree of rotational replication may significantly degrade large I/O bandwidth.

In addition to bandwidth, replicating within a track may also negatively impact read latency. A higher degree of rotational replication effectively reduces the length of a track, thus increasing the likelihood of track switches (depending on the nature of locality).

To avoid unnecessary track switches, we place the replicas on different tracks, within the same cylinder (shown in Figure 2(c)), elsewhere on the same disk, or on different disks whose spindles are synchronized.

2.1.3 Combining Techniques for Reducing Both Seek and Rotational Delay

In the past sections, we have discussed techniques for reducing seek distance and rotational delay in isolation. Now we consider the combined effect. The combined effect, however, is more complex than it might first appear: choosing a replica that is closest in seek distance may not result in the optimal rotational delay. Also, the total latency is not necessarily a sum of the seek delay and rotational delay; instead, they may overlap. We consider two alternatives in this section. We defer their quantitative evaluation until later sections. Here we consider a number of issues qualitatively.

Inter-disk and Intra-disk Approaches

The first alternative is mirroring on disks whose spindles are carefully synchronized out of phase. This reduces both seek distance (as shown by Equation (2)) and rotational delay (as shown by Equation (7)). We consider this the *inter-disk* approach because the replicas are on different disks.

The second alternative is a combination of splitting (or replicated splitting) (as shown by Equation (3) or (4)) and rotational replication (as shown by Equation (7)). We call this the *intra-disk* approach. If we choose to use replicated splitting, then there are two types of replicas. The purpose

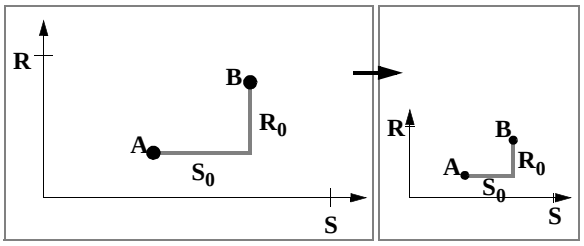

Figure 3: The two dimensions involved in latency: seek distance on the x-axis and rotational delay on the y-axis. In order to halve the "distance" between two random sectors A and B, in general, we must halve the "distance" in each dimension.

of replicas that are in the same cylinder is to reduce rotational delay. The purpose of replicas that are in cylinders that are far apart on the same disk is mainly seek distance reduction, although they can also be carefully placed in such a way that they contribute to rotational delay reduction as well. Both of these types of replicas are within the same disks, hence the name intra-disk approach.

The First Square Root Rule

Both the inter-disk and intra-disk approaches share a number of common elements. One is that there exists a tension between reducing rotational delay and reducing seek distance. For example, if we raise the degree of replication within a cylinder to reduce rotational delay, we push data belonging to different cylinders farther apart from each other, raising seek distance.

Figure 3 shows that there are two dimensions contributing to the total latency: seek distance and rotational delay. In order to improve the overall latency by a certain factor, we must at least improve each dimension by the same factor. This leads to our first rule of thumb.

Rule 1 *By using D disks, we can improve the overhead-independent part of random read latency by a factor of $\sqrt{D}$.*

Of course, when there is locality, we can do better than Rule 1: the contribution of rotational delay may be greater than that of seek distance when locality exists. By devoting an extra disk to the dimension that can offer the most gain as we increase the number of disks, we can best take advantage of the resources.

Advantages of the Intra-disk Approach

The intra-disk approach inherits the advantages of disk splitting compared to mirroring as discussed in Section 2.1.1. In order to achieve a factor of $\sqrt{D}$ improvement, the intra-disk approach requires only $\sqrt{D}$ rotational replicas while the inter-disk case requires D replicas. Consequently, propagating replicas is less costly under the intra-disk approach.

An even more important difference is the throughput of queued requests that can be reordered. Under the intra-disk approach, all replicas exist on a single drive; so scheduling is trivial: requests are simply sent to the drive that is solely responsible for the data. Each drive queues requests locally and performs scheduling locally. A good local scheduler considers both seek distance and rotational replicas to maximize throughput. Furthermore, each drive is only responsible for a subset of the data, so we can reduce the amount of head movement by avoiding having to pass over the remainder of the data.

In contrast, under mirroring, any request can be scheduled for any individual drive queue so the global scheduling is complicated. Furthermore, because each disk contains all the data, a naive layout policy or scheduling algorithm will result in a greater amount of head movement than the intra-disk approach as the head has to pass over the entire data set as opposed to a subset.

Traditionally, the best throughput one can hope for with a mirrored system is linear scale-up as we increase the number of disks. In contrast, because the latency of individual requests improves as we add disks to an intra-disk system, we can achieve super-linear scale-up.

2.2 Altering Disk Geometry

Much of disk drive design is based on ad hoc historic reasons such as form factors. As disk capacity rapidly increases, disks are becoming increasingly unbalanced in the relationship between capacity and latency. The techniques of using extra disks to reduce latency are in effect emulating faster disks using existing slow disks. A more direct and more cost-effective approach to balancing capacity with lower latency is to simply build "better" drives to start with so we do not have to resort to either replication or discarding space. As array manufacturers such as HP [30] and EMC [2] explore intermediate storage levels between RAID-5 and memory, there is an added incentive for us to seriously examine the implication of "faster" drives.

2.2.1 Reducing Disk Diameter

By using only a portion of the disk, splitting is in effect emulating a smaller disk using a large disk. A more direct approach is to simply reduce the disk

diameter. Reducing disk diameter has an immediate effect on both maximum seek distance and rotational speed. The effect on seek distance is obvious. There are also secondary benefits such as the fact that a smaller arm can be made more rigid and is easier to control. The effect on rotational speed, however, is less obvious and requires more explanation.

Constraints on the Increase of Rotational Speed

There are two possible obstacles to increasing rotational speed: the first is the maximum rate at which the internal data channel electronics is able to lift bits off the platter; the second is power consumption. We consider each of these constraints in turn.

Today, the drive data channels are running near 200 MHz. Some of the internal clocks run at twice that rate. Fast clock rates are needed to keep up with the rapid linear bit density growth [10]; about half of today's areal density growth is devoted to the linear bit density growth, while the remaining half is devoted to track density growth. The implication of the high linear bit density is that given a particular generation of technology, the internal data rate is *the* constant. A corollary is that the linear velocity must be kept constant as we vary the rotational speed. In other words, a reduction of diameter can be matched by an increase of the same factor in rotational speed without exceeding the allowable data rate.

The second possible constraint is power consumption. Interestingly, power is proportional to the 4.6th power of diameter but only the 2.8th power of rotational speed [10]. Consequently, this is a lesser constraint than the internal data rate. In other words, if we increase the rotational speed while maintaining the same data rate, for a smaller disk the power consumption of the new drive will be less.

The Second Square Root Rule

Suppose we decrease the platter capacity by a factor of C. Given the same areal density, the platter area also decreases by a factor C. As a result, the diameter decreases by a factor of $\sqrt{C}$. This reduces maximum seek distance by a factor of $\sqrt{C}$. At the same time, the internal data rate constraint discussed above allows us to increase the rotational speed by a factor of $\sqrt{C}$. We summarize this in our second rule of thumb.

Rule 2 *By reducing the platter capacity by a factor of C, we can improve the overhead-independent part of random read latency by a factor of $\sqrt{C}$.*

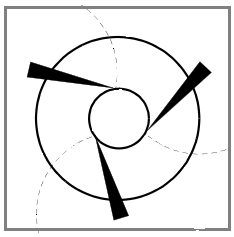

Figure 4: Increasing the number of heads per surface.

Unlike the use of extra disks, however, reducing diameter does not allow us to flexibly devote more resources to one of the seek and rotational dimensions: each dimension always gains an equal improvement of $\sqrt{C}$ given a C-fold reduction of capacity.

Although it is impossible to *dynamically* adjust the amount of improvement that each dimension receives, it is possible to do so *statically* by adjusting the inner and outer radius of the platters while maintaining constant area. A larger inner radius results in fewer tracks and better seek characteristics at the expense of slower rotational speed, while a smaller inner radius has the opposite effect. The optimal radius under the power and data rate constraints is a subject of our current research.

2.2.2 Increasing the Number of Heads per Surface

The techniques that we have examined so far all involve increasing the number of spindles for the same amount of usable capacity. We now turn to the most direct and potentially most cost-effective way of building a more "balanced" drive: increasing the number of heads per surface. Figure 4 shows this approach. Each head is mounted on its own independent arm. The arms are spaced in such a way that they cannot collide. The size of the inner radius of the platter limits the number of heads that we can place on a single surface.

Although such a drive can amortize the cost of the components such as the spindle and the power supply over a larger number of heads, due to the data rate constraint discussed earlier, each arm in such a drive is likely to need its own data channel.

To provide reliability, we can complement these more expensive drives with cheaper conventional drives that are write-only *secondary drives*. Using write-optimized techniques such as those that perform writes to free sectors that are closest to the head position [4, 29], we can easily achieve very low write latency even on a conventional drive.

The Third Square Root Rule

Increasing the number of heads per surface is very much like employing extra disks. In fact, we can show that a D-way mirroring system with synchronized spindles can emulate the behavior of a disk with D heads per surface by carefully choosing the rotational positions of the replicas. We summarize this in our third rule of thumb.

Rule 3 *By placing H heads on each surface, we can improve the overhead-independent part of random read latency by a factor of $\sqrt{H}$.*

This approach also affords us the largest degree of freedom in terms of choosing which one of the seek and rotational dimensions we would like to devote more resources to. Recall that when we use extra disks to reduce latency, we adjust the amount of resource we devote to each dimension by changing the number and placement of replicas. Also recall that when we reduce platter capacity to reduce latency, we can only make such adjustments statically by changing the platter inner radius. When we employ multiple heads per surface, all we need to do for such adjustments is to change the positioning of the heads.

2.3 Summary and Discussion of Latency Reduction Techniques

We have examined a number of latency reduction techniques. The first set of techniques use extra disks to accomplish this goal. The second set of techniques shift the balance between capacity and latency by altering disk geometry. These techniques form a spectrum in terms of their ease of adoption and cost-effectiveness.

2.3.1 Cost-effectiveness

The technique of using extra disks is the least cost-effective because replication and discarding space are both wasteful. Maintaining replicas also introduces complexity. Reducing platter diameter is less wasteful because no bits are wasted; but this technique results in an increase of both the number of heads and spindles for the same amount of capacity, so it is more costly than the third technique of just adding heads. We note that there does not exist a single "perfect" drive that has the "right" diameter and "right" number of heads per surface. Instead, there may exist one "right" drive for every cost/performance specification.

An alternative to these techniques is to simply buy more memory to address low memory cache hit rates. This strategy is clearly viable if we have enough money to buy extra memory to significantly raise the cache hit rate. However, when operating under regimes where we cannot afford the large sum of money required to buy extra memory to raise the cache hit rate, but we can still significantly increase the number of the disk heads, our techniques are better.

This phenomenon can be explained by Amdahl's Law, which states that the performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used. Although memory is much faster, the large cost gap between disk and memory dictates that only a small fraction of the accesses can enjoy this speedup. In contrast, when spent on extra disk heads, the same amount of money allows one to speed up a larger fraction of the accesses, achieving a larger overall performance gain. Of course, more disk heads also result in other benefits such as better write throughput.

2.3.2 Ease of Adoption

The easiest to implement is reducing seek distance using splitting or replicated splitting. If we can accurately predict the position of a disk head at any instant, the second practical technique that we can incorporate is even rotational replication. In Section 4.1, we show how this can be done without hardware support in the drive. The third in line in terms of feasibility is reducing disk diameter. There is considerable resistance to any change to established form factors, although once in a while, we do see diameter reductions, albeit in an ad hoc way. For example, we have recently seen the emergence of 2.5-inch drives that spin at 12,000 RPM. The last technique of increasing number of heads per surface is likely to face the most resistance. Large drives previously employed this technique to devote separate heads to different cylinder groups to reduce seek time. Their popularity has died down, partly due to the cost sensitive nature of the desktop market. As drive capacity continues its rapid increase, and as we develop new software techniques that can take advantage of the extra heads (such as using them for reducing rotational delay and fast write techniques [4, 29]), we believe that this approach is becoming more attractive.

3 Simulations

In this section, we perform simulations to accomplish two goals. One is to validate the Square Root

Diameter	3.5 inch
Capacity	4.55 GB
Average sectors per track	212
Cylinders	6962
Heads	6
Spindle speed	10,025 RPM
Interface	Ultra2 SCSI
Single track seek	0.8 - 1.1 ms
Maximum seek	12.2 - 13.2 ms

Table 1: Parameters of the Seagate ST34502LW (Cheetah 9LP).

Rules that we have intuitively motivated for random reads in the last section. The second is to study the effectiveness of our latency reduction techniques on real world disk access traces. We present results from micro-benchmarks on a prototype implementation in Section 4.

3.1 Simulators

As a simulation platform, we use a modified version of DiskSim, a disk simulation environment built at the University of Michigan [8]. We modified the simulator to evaluate our various latency reduction techniques as well as to identify the software enhancements required by a disk controller module. For techniques that replicate data, the simulator identifies the various replicated blocks and chooses the closest copy. The primary source of complications is the use of variable density disk regions that are commonly referred to as zones or bands. Our simulation modules use a variety of book-keeping data structures to help us identify the various physical copies of a given logical block without having to maintain explicit space-inefficient disk maps. The choice of the closest copy is made using seek profiles obtained from actual measurements so as to closely mimic the access latencies of disks. As validation, our simulator accurately predicts the measurements from our prototype implementations for random work loads.

The disks that we simulate are the Seagate ST34502 (Cheetah 9LP). We define *overhead* as the part of the latency that cannot be improved by the use of extra disks. This includes host processing, SCSI command processing, bus arbitration, transfer times, and the mechanical delay incurred in order to accelerate and decelerate the arm for long distance seeks[2]. We have experimentally determined that this overhead is about 2.7 ms for a single sector read on the Cheetah 9LP, shown by the bottom line labeled "0" in Figure 5. Of this 2.7 ms, about 1.5 ms is due to the acceleration and deceleration of the arm, which is not likely to dramatically improve in the future, while the remainder is more amenable to optimization by disk manufacturers.

3.2 Techniques Simulated

We simulate five latency reduction techniques. "Unsynchronized mirroring" uses mirrored disks whose spindles are not synchronized. The tracks at the same relative position within different disks share the same content but the rotational positions of the replicas are random with respect to each other.

"Synchronized mirroring" uses mirrored disks whose spindles are synchronized. The replicas on different disks share the same track number and are evenly spaced from each other rotationally. In both mirroring schemes, we greedily choose a disk head that is closest to a replica of the target sector.

The "intra-disk" approach uses a combination of replicated splitting and rotational replication. We perform splitting at cylinder granularity and evenly distribute the disks to the seek and rotational dimensions.

We simulate two additional techniques that alter disk geometry: reducing platter size and increasing heads per surface. Other than these two parameters, we retain all other characteristics of the Cheetah 9LP. For the technique of reducing platter size, retaining the seek profile of the Cheetah 9LP is a conservative assumption since smaller drives usually have smaller and more rigid arms that have better seek profiles.

3.3 Random Read Latency

Figure 5 shows the latency of the random read benchmark. The curve labeled "4" shows the latency predicted by the Square Root Rule ($2.7 + 6.2/\sqrt{D}$ ms). All techniques are remarkably close to each other in their latency and the Square Root Rule is a very accurate prediction. Although splitting is a more effective way of reducing seek distance than mirroring, the mirroring latency is close to that of the intra-disk approach because an additional disk in a mirrored system can improve both

[2] According to the seek model developed by Ruemmler and Wilkes [25], a seek profile can be roughly approximated by two parts. Below a small threshold distance, the seek delay is linear with respect to the square root of the seek distance; beyond that threshold, the seek delay is linear with respect to the seek distance. The y-intercept of the second part is largely due to acceleration and deceleration of the arm.

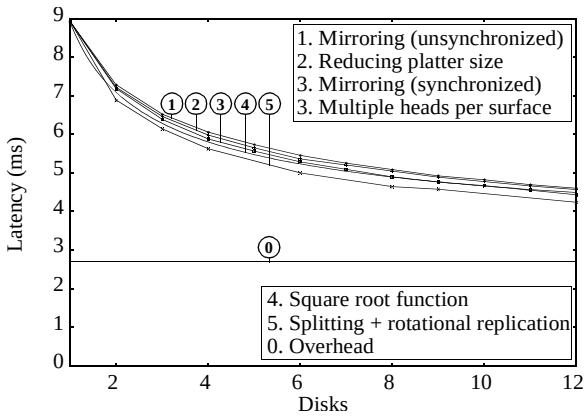

Figure 5: Latency of random reads. When reducing platter size, the x-axis denotes the number of disks required to obtain the same capacity as the 4.55 GB Cheetah 9LP. When increasing heads per surface, the x-axis denotes the number of heads per surface.

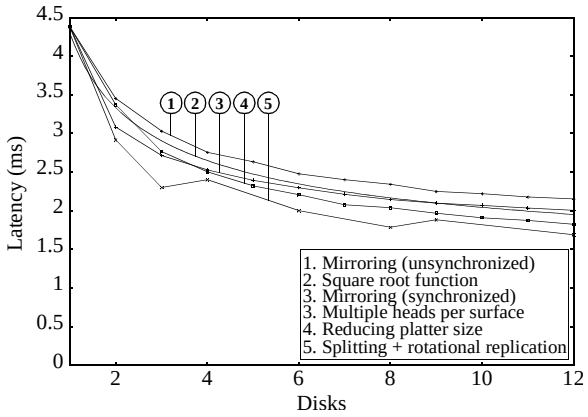

Figure 6: Latency of file system trace "hplajw".

seek and rotation while the intra-disk approach requires separate disks devoted to each dimension. Although the intra-disk approach has the best latency, the performance difference is small enough that its attractiveness will only become evident in later sections. Similarly, although the techniques that alter disk geometry do not perform better than other approaches with the same number of disk heads, their main attractiveness is their simplicity and cost effectiveness.

3.4 File System Latency

We next study the effectiveness of the latency reduction techniques on a file system workload, which should contain a fair amount of locality. We use the "hplajw" trace [3] collected at HP Labs [24]. The week-long trace records all disk accesses on a single user workstation. The danger of relying on this disk level trace is that such traces are sensitive to the particular file system implementation and technology limitations at the time when the trace was taken (such as cache size and disk capacity). Nonetheless, we believe that the locality pattern demonstrated in this trace is still largely relevant. We ignore the writes in the trace and only consider the read latency, assuming that the replicas, if there are any, are propagated in the background.

The "hplajw" trace contains activity on two disks, but the accesses to one is insignificant so we focus our attention on the dominant disk. The disk traced is small and we only use about 1/6 of the cylinders on the Cheetah disk. This has the effect of performing a 6-way split before we apply any other latency reduction techniques. Still, the data spans enough cylinders to make seek reduction potentially worthwhile, although the rotational delay is a more important consideration.

Figure 6 shows the read latency from this trace. As a result of the 6-way split and locality in the reference, the one disk latency is only half of that of random reads. The latency improvement of all techniques, however, still closely follows the Square Root Rule, shown by the curve labeled "2" ($1 + 3.3/\sqrt{D} ms$). The overhead (as defined in Section 3.1) of roughly 1 ms is significantly lower that the 2.7 ms seen in the random read case because the disk head never seeks very far in this case. The anomaly for curve "5" at three disks occurs because we cannot evenly distribute three disks along the seek and rotational dimension. Although using all disks for rotational reduction does not always result in the best performance, it does with three disks.

3.5 Transaction Processing Latency

The third case we study is a TPCC disk I/O trace, also from HP Labs[4]. The trace contains 4.25 million records, about half of which are reads. The trace contains activities on 38 disks, most of which share the load evenly. Similar to the disks used in the "hplajw" trace, these disks are much smaller than the Cheetah 9LP. In our simulations, however, we still devote an entire Cheetah disk to each disk that appears in the trace.

Figure 7 shows the latency results. (We defer the throughput study to Section 4.3.) Although we conjecture that the TPCC benchmark has less locality than the file system trace, the simulation results are similar. This is because the approach of

[3] We have used the week-long version on the HPL web site. We are obtaining results from the longer version and the rest of the trace set.

[4] This is an unaudited run.

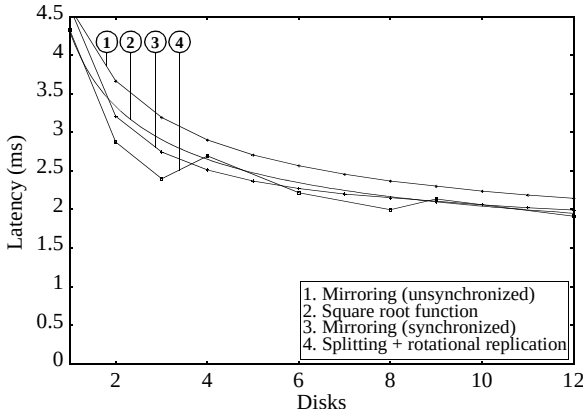

Figure 7: Latency of the transaction processing benchmark TPCC.

substituting the small disks with the large Cheetah disks (and using only a small portion of each disk) has drastically reduced the average seek distance. Had we filled the Cheetah disks, we would expect the result to be closer to Figure 5. The result of the TPCC run also largely obeys the Square Root Rule.

4 Implementation and Experimental Results

In this section, we describe the prototype MimdRAID implementation and report the experimental results. The main goals of these experiments are: 1) to validate the First Square Root Rule, 2) to evaluate the various latency reduction techniques that utilize extra disks, and 3) to explore the feasibility of accurate tracking of the disk head position for the purpose of rotational delay reduction.

For the techniques that require propagation of replicas in the background, we have not implemented such propagations. We expect the cost of such propagations to be masked by a combination of idle time between accesses and extra resources (in the forms of extra disks, heads, and/or interconnect bandwidth) that are devoted to propagating replicas so that it does not severely interfere with user operations. These issues are subjects of our current research.

The experiments are performed on a Pentium II 300 that runs Windows NT 4.0 service pack 5. Attached to the host, using an Adaptec 2940U2W SCSI card, are four Seagate ST34502 (Cheetah 9LP) disks, whose key parameters are shown in Table 1.

Currently, our implementation consists of a set of user level libraries that are not yet integrated with the operating system or any file system. As a result, we are only able to evaluate the different latency reduction techniques with micro-benchmarks. We are also in the process of implementing fast write-anywhere techniques [29] using the disk head tracking mechanism described in the next section. The write implementation is not yet complete.

4.1 Tracking the Disk Head Position

With the exception of disk splitting, all other latency reduction techniques that use extra disks rely on the ability to accurately predict the disk head location at any time. With this knowledge, we can choose the closest alternative target sector among several possible choices. Without this knowledge, for mirroring, instead of choosing the closest replica to read, the controller can only issue a number of requests for the same data to a number of disks in parallel and return the requested data to the host as soon as the first response from the disks arrives. Unfortunately, by wasting the work done by the remaining disks, this approach greatly reduces available throughput. Therefore, being able to predict the precise locations of the disk heads is crucial for both intra-disk and inter-disk approaches.

In order to predict the disk head position at arbitrary times, we need three pieces of information. The first is the current rotational position of the head. The second is the current track number of the head position. The third is the amount of time required for executing various operations such as track switches, seeks, reads, and writes.

Obtaining the rotational position of the disk head is the most difficult. Previous proposals that depend on the knowledge of head positions have relied on hardware support [4, 29]. Unfortunately, this level of support is rarely available from commodity drives. We have developed a software-based calibration method.

Our head tracking algorithm correlates the disk head position with the time stamp taken immediately after the completion of a read request that is delivered to the disk surface. In order to minimize operating system involvement, we directly issue SCSI commands using either a user level SCSI interface or a loadable kernel module. To ensure that the request goes to the disk surface instead of being absorbed by the cache, we use the direct-media access feature supported by most SCSI disks.

Suppose a track has B sectors, a full rotational delay is R, and the SCSI command overhead is O_s. We pick an arbitrary sector as the *reference sector*. Our algorithm consists of the following steps:

- Measure the rotational delay R by repeatedly reading the reference sector.

- Read the reference sector and record the time stamp t_1 upon completion of the read. We can calculate the time stamp t_0 immediately before the head is about to perform the read as:

$$t_0 = t_1 - O_s - \frac{R}{B}$$

- At any given time t, the rotational position of the head is given by:

$$360^\circ \cdot \frac{(t - t_0) - R \cdot \lfloor \frac{t - t_0}{R} \rfloor}{R}$$

  where "$\lfloor\ \rfloor$" is the floor function.
- Periodically, we re-calibrate the head position by reading the reference sector and re-taking the time stamps t_1 and t_0. We also recompute the rotation speed by noting the difference between the actual time stamp t_1 and its predicted value t_1' and adjusting the rotational speed accordingly.

Our experiments show that periodic re-calibration at an interval of two minutes yields a maximum error of only 1% (3.6 degrees) for the 10,000-RPM Seagate Cheetah 9LP. This is encouraging, not only because of the high degree of accuracy, but also because of the negligible overhead involved in reading one reference sector every two minutes. A variation of this approach is to keep track of the location and timing of the actual disk requests and use this information to calibrate the head location. We have not found the need to try this method due to the success that we have had with our current approach.

Once we have developed an algorithm for tracking the rotational position of the disk head, the remaining two tasks are relatively simple. Finding out which track the disk head is on reduces to the problem of obtaining the logical to physical block mapping. We need to obtain per-zone information including the number of cylinders in the zone, number of tracks per cylinder, number of sectors per track, and track skew. This information is obtained by a combination of issuing the SCSI "MODE SENSE" command and empirical measurements that are similar to those proposed by Worthington [32]. The process is further simplified on the Seagate disks by the fact that bad sectors do not appear to disturb the global mapping.

The last piece of information that we need for accurate prediction of head positions is the timing of track switch, seek, read, and write operations. Our experiments on the Cheetah 9LP suggest that our predictions are within a range of $\pm$ 100 μs of the expected value for short seeks and $\pm$ 200 μs for long seeks. Considering that a full rotational delay is 6 ms on these disks, this is satisfactory for the degree of rotational replication that we are interested in.

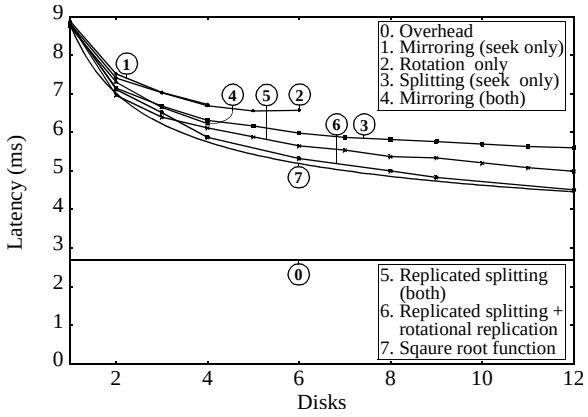

Figure 8: Random read latency.

4.2 Latency Reduction Techniques Implemented and Latency Results

We first describe the latency reduction techniques that we have implemented and we examine the latency of a random read benchmark. The average latency of randomly reading a sector on a Cheetah 9LP disk is 8.8 ms (shown in the upper left hand corner of Figure 8). The bottom line labeled "0" shows the overhead as defined and measured in Section 3.1.

The first strategy that we implement is to read the replica with the shortest seek distance in a mirrored system. If the disk head position cannot be tracked, this strategy obtains the best performance out of a mirrored system and serves as the baseline for comparison. The latency is shown by the top curve (labeled "1"), which closely follows Equation (2) of Section 2.1.1. The curve stops at four disks because this was the number of Cheetahs that we had at the time of the experiment.

The second strategy is even rotational replication on adjacent tracks (without any seek distance reduction). For a D-way rotational replication, each disk contains $1/D$ of the original content. The latency is illustrated by the curve labeled "2". This is an intra-disk replication strategy so we could simply use just one disk to study the effect of multiple disks. However, there is a limit on how accurately we can predict the precise head location, especially after a long distance seek. This in turn limits the number of rotational replicas that we can effectively take advantage of. In our implementations, we stop at six disks.

The third strategy is disk splitting, shown by the

curve labeled "3", which closely follows Equation (3) of Section 2.1.1. No replication is involved and we simply use only $1/D$ of the disk capacity for a D-way split (as shown in Figure 1(b)). This strategy has no effect on rotational delay. By concentrating on only one of the seek and rotation dimensions, the overall performance gain of any one of these first three strategies is limited because they are limited by the dimension that they ignore. We next turn our attention to techniques that address both of the seek and rotation dimensions.

Under the fourth strategy, we choose the replica that is closest to the current disk head position in a mirrored system by considering both the seek and rotational distance. Indeed, the seek and rotational delay may partially overlap. The latency of this implementation is illustrated by the curve labeled "4", which also stops at four disks, the number of disks that we had at the time of the experiment. In our implementation, the disk spindles are not synchronized and the replicas are made at random locations with respect to each other, an approach that is consistent with our strategy of performing writes to free sectors closest to the head position. We expect to achieve some additional performance gain on reads if we perform writes to evenly spaced rotational positions while keeping the spindles synchronized.

The fifth strategy is replicated splitting. Compared to simple splitting, the difference is that this strategy replicates the data (from the outer tracks) in the "discarded" space in the middle (as shown in Figure 1(c)). We choose the copy that is closest to the disk head. The resulting latency is shown by the curve labeled "5". The performance is better than that of simple splitting due to both seek and rotational reductions. Detailed measurements show that the benefit derived from rotational reduction is more significant.

The attraction of replicated splitting is that it can take advantage of the space that is "discarded" by simple splitting. However, although the replicas are at different rotational positions, they are necessarily separated by a long seek distance. Partially because of this reason, our experiments suggest that making more than two replicas does not significantly improve performance. At a high degree of splitting, the latency is again dominated by a large rotational delay.

This leads to the sixth (and final) strategy. On the Cheetah 9LP, as soon as we reduce delay in one of the seek and rotational dimensions, the other becomes more dominant. Therefore, we attempt to evenly distribute the disks to these two dimensions. For example, with four disks, we perform a two-way split and a two-way even rotational replication. We denote this as a 2×2 configuration. With six disks, we use a 3×2 configuration. The curve labeled "6" shows the resulting latency. As a comparison, the curve labeled "7" shows the latency predicted by the First Square Root Rule ($2.7 + 6.1/\sqrt{D}\ ms$). The latency points that exhibit slightly larger deviations from the square root curve are those that we cannot evenly distribute the disks along the two dimensions. For example, when given three disks, we have to settle for a 3×1 configuration that has a disproportionally large rotational delay. Overall, we are pleased to see that our implementation has achieved a close approximation to the latency result predicted by the Square Root Rule.

4.3 Throughput

Traditionally, in a well designed RAID, as we increase the number of disks, the latency experienced by each request remains constant; so the overall throughput increases linearly. In contrast, as we increase the number of disks in MimdRAID, the latency experienced per request decreases according to the Square Root Rule; so the overall throughput should achieve super-linear speedup. In this section, we study these effects.

4.3.1 Disk Head Scheduling

Three factors determine the throughput. One is the organization of data. The second is the disk head scheduling policy. The third is the length of the scheduling queue that contains disk requests that can be reordered. We examine three strategies while varying the length of the reorder queue for each.

We label the first strategy as "mirroring". Under this strategy, we store data in a D-way mirror. The RAID controller evenly distributes k (random) incoming requests to each of the disks at a time. We maintain a queue of length k for each disk on the host and schedule these requests using an elevator algorithm.

We label the second strategy as "splitting". Under this strategy, we store $1/D$ of the data on the outer tracks of each of the D disks. Because there is no redundancy, the RAID controller simply distributes a read request to the disk that is solely responsible for that data. We employ the same elevator algorithm for each disk's queue.

We label the third strategy as "intra-disk". Under this strategy, we use a combination of splitting and inter-track rotational replication. For example, 12 disks form a "4×3 configuration": four disks devoted to the seek dimension using splitting, and

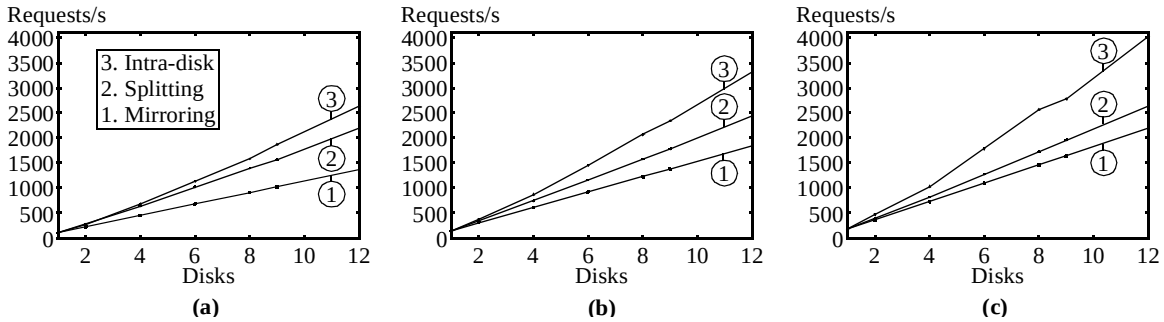

Figure 9: Throughput of random reads. The reads are queued and potentially scheduled out of order. (a) The per drive reorder queue contains 4 requests. (b) The reorder queue contains 16 requests. (c) The reorder queue contains 64 requests.

three disks devoted to the rotational dimension using rotational replication. The RAID controller distributes a read request to the only disk that is responsible for the data. We use a variant of the elevator algorithm for each disk's queue: the disk services the read requests in the same order as it would under the conventional elevator algorithm but chooses the closest rotational replica for each request. Note that "splitting" is a special case of "intra-disk" if we set the number of rotational replicas to one. In our experiments, we have chosen integer solutions for the two dimensions that are close to $\sqrt{D}$.

4.3.2 Throughput Results

We subject the three strategies to the single sector random read benchmark. We show the throughput comparisons with various queue length conditions in Figure 9. The line labeled "1" shows the linear increase in throughput expected of mirroring as we increase the number of disks while keeping the per request latency a constant. The throughput also increases as we increase the scheduling queue length. This occurs because we are able to serve more requests with each seek stroke as the queue lengthens.

The line labeled "2" shows the super-linear rise of throughput as we increase the number of disks using the splitting strategy. Because of the restriction on head movement and the way data is partitioned, during each elevator stroke, each head only passes over a subset of the entire data set. In contrast, in order to serve these same requests on a single elevator stroke, each head under the mirroring strategy would have had to pass over the entire data set, incurring a longer overall seek distance. The throughput of splitting also increases with a longer queue length for the same reason that it improves under mirroring. The throughput difference between splitting and mirroring, however, narrows under heavier queueing. This is because mirroring can better amortize the long seek distance under such conditions and the effects of other latency components such as rotational delay become more prominent in both approaches.

It is worth noting that it is possible to emulate the behavior of splitting in a mirrored system by partitioning data and restricting head movement in a way that is similar to splitting; by carefully directing requests to maximize locality, mirroring effectively becomes splitting.

The line labeled "3" shows the super-linear throughput scale-up of the intra-disk strategy. This strategy has the best performance: it achieves a 21× speedup with 12 disks, which is approximately 80% to 95% better than mirroring under various queueing conditions. Unlike the two previous strategies, which become dominated by rotational delays at a large number of disks, the intra-disk approach evenly distributes the disks along the seek and rotational dimensions. As a result, the performance gap between this strategy and the others widens as the number of disks increases.

It is also worth noting that the intra-disk scheduling algorithm can be emulated on a mirrored system with synchronized spindles. In terms of data layout, there is a natural one-to-one correspondence between a replica in the intra-disk approach and a replica in the synchronized mirroring approach. In terms of disk head placement, we can emulate an "$x \times y$" intra-disk configuration on a mirrored system by restricting head movement to a small number of cylinders. As a result, the aggregate of the local head schedules under an intra-disk system can be emulated with an equivalent global schedule. As a side-effect of this study we have developed an effective head scheduling algorithm for a mirrored system to improve latency and throughput. Maintaining and scheduling the global queue, however, is more complex than queueing and scheduling locally on each drive.

5 Related Work

In Section 1.2, we have briefly described the HP AutoRAID [30] as an example system that systematically trades off capacity for performance. As with RAID-1 systems, however, its primary focus is solving the small write problem of RAID-5. Although mirrored systems can improve read performance by dynamically balancing the load, they have not aggressively sought low read latency. In order to achieve the low read latency goal, we have taken the mirrored approach further in a number of directions: exploring alternatives other than inter-disk mirroring, exploring alternatives that can take advantage of more than doubling the amount of resources, exploring the implication of altering disk geometry, and quantifying the difference among all these alternatives.

The HP Ivy project [16] is a simulation study of how a high degree of replication can improve read performance. Our study differs from Ivy in several significant ways. First, Ivy only explores reducing seek distance and leaves rotational delay unresolved. Second, Ivy only examined mirroring. Section 2.1.1 discusses the advantages of disk splitting compared to mirroring. Indeed, the Ivy study shows that propagating the replicas in order to maintain a high degree of replication may result in significant increase of disk queue length, thus undoing some of the benefits of mirroring. The third difference is a feature of Ivy that we intend to incorporate into MimdRAID in the future: Ivy dynamically chooses the candidate and the degree of replication by observing access patterns. We are currently researching a wide range of access patterns that can be used to dynamically tune MimdRAID behavior. These include the read/write ratio to guide the placement of writes using different logging techniques, the relative contribution of seek and rotation to the total latency to guide which of these two dimensions should receive more resources, and the frequency and nature of accesses to guide the number and placement of replicas.

Ng examines intra-track replication as a means of reducing rotational delay [20]. Section 2.1.2 has explained that a disadvantage of this approach compared to inter-track replication is that it degrades large I/O bandwidth due to more frequent track switches.

The importance of reducing rotational delay has long been recognized. Seltzer and Jacobson independently examined a number of disk scheduling algorithms that take rotational position into consideration [12, 26]. Although these techniques are useful for the local scheduler on the individual drives, it is less clear how these techniques can be extended for replicated splitting, rotational replication, and mirrored systems. These are subjects of our current research.

The history of multi-headed disks dates back to drums (or fixed-head disks), a technology that is no longer viable due to its high cost. Multi-headed disks, however, survived long after the demise of drums. Each head is responsible for a subset of the cylinders to reduce seek distance. However, we are not aware of approaches that used multiple heads to improve rotational delay.

One of our goals of studying the impact of altering disk geometry is to understand how to configure a storage system given certain cost/performance specifications. The "attribute-managed storage" project [9] at HP shares this goal, although its focus is at the disk array level as opposed to individual drive level.

6 Conclusion

The vast performance and cost gap between DRAM and disk presents an opportunity for examining cost effective storage alternatives to fill this gap. In this paper, we have demonstrated how we can construct a low latency secondary storage system by systematically increasing the ratio between disk heads and usable capacity. The techniques include using extra disks, reducing platter size, and increasing the number of heads per surface. We show that these techniques share a Square Root Rule, which states that the overhead-independent part of the latency improves by a factor of the square root of the amount of extra resources.

We have developed a number of latency reduction techniques that takes advantage of extra capacity. Our latency reduction techniques include splitting, which reduces seek distance without maintaining replicas, replicated splitting, a variant of splitting that reduces latency even further by using the space wasted by splitting, and inter-track rotational replication to effectively reduce rotational delay. A D-way intra-disk system that combines splitting and rotational replication can provide slightly better latency than a D-way mirror with only $\sqrt{D}$ replicas. Furthermore, as a result of reducing latency per request, an intra-disk system can achieve super-linear scale-up in throughput as the number of disks increases, significantly outperforming a conventional mirrored system.

References

[1] Baker, M., Hartman, J., Kupfer, M., Shirriff, K., and Ousterhout, J. Measurements of a Distributed File System. In *Proc. of the 13th Symposium on Operating Systems Principles* (Oct. 1991), pp. 198–212.

[2] Brant, W. A., and Nielson, M. E. Disk based cache interfacing system and method. U.S. Patent 5805787 issued to EMC Corporation, March 1998.

[3] Chang, F., and Gibson, G. A. Automatic I/O Hint Generation Through Speculative Execution. In *Proc. of the Third Symposium on Operating Systems Design and Implementation* (February 1999).

[4] Chao, C., English, R., Jacobson, D., Stepanov, A., and Wilkes, J. Mime: a High Performance Parallel Storage Device with Strong Recovery Guarantees. Tech. Rep. HPL-CSP-92-9 rev 1, Hewlett-Packard Company, Palo Alto, CA, March 1992.

[5] Chen, P., Lee, E., Gibson, G., Katz, R., and Patterson, D. RAID: High-Performance, Reliable Secondary Storage. *ACM Computing Surveys 26*, 2 (June 1994), 145–188.

[6] Dahlin, M., Wang, R., Anderson, T., and Patterson, D. Cooperative Caching: Using Remote Client Memory to Improve File System Performance. In *Proc. of the First Symposium on Operating Systems Design and Implementation* (November 1994), pp. 267–280.

[7] Feeley, M. J., Morgan, W. E., Pighin, F. P., Karlin, A. R., Levy, H. M., and Thekkath, C. A. Implementing Global Memory Management in a Workstation Cluster. In *Proc. of the 15th ACM Symposium on Operating Systems Principles* (December 1995), pp. 201–212.

[8] Ganger, G. R., Worthington, B. L., and Patt, Y. N. The DiskSim Simulation Environment Version 1.0 Reference Manual. Tech. Rep. CSE-TR-358-98, Department of Electrical Engineering and Computer Science, February 1998.

[9] Golding, R., Shriver, E., Sullivan, T., and Wilkes, J. Attribute-managed Storage. In *Workshop on Modeling and Specification of I/O* (October 1995).

[10] Grochowski, E. Personal Communication, IBM Almaden Research Center, March 1999.

[11] Growchowski, E. Emerging Trends in Data Storage on Magnetic Hard Disk Drives. *Datatech* (Fall 1998), 11–16.

[12] Jacobson, D. M., and Wilkes, J. Disk Scheduling Algorithms Based on Rotational Position. Tech. Rep. HPL-CSP-91-7rev1, Hewlett-Packard Company, Palo Alto, CA, February 1991.

[13] Kimbrel, T., Tomkins, A., Patterson, R. H., Bershad, B., Cao, P., Felten, E. W., Gibson, G., Karlin, A. R., and Li, K. A Trace-Driven Comparison of Algorithms for Parallel Prefetching and Caching. In *Proc. of the Second Symposium on Operating Systems Design and Implementation* (October 1996), pp. 19–34.

[14] Lamb, C., Landis, G., Orenstein, J., and Weinreb, D. The ObjectStore Database System. *Communications of the ACM 34*, 10 (October 1991), 50–63.

[15] Lee, E. K., and Thekkath, C. E. Petal: Distributed Virtual Disks. In *Seventh International Conference on Architectural Support for Programming Languages and Operating Systems* (October 1996), pp. 84–92.

[16] Lo, S.-L. Ivy: A Study on Replicating Data for Performance Improvement. Tech. Rep. HPL-CSP-90-48, Hewlett-Packard Company, Palo Alto, CA, December 1990.

[17] Mashey, J. R. Big Data and the Next Wave of InfraStress. Computer Science Division Seminar, University of California, Berkeley, October 1997.

[18] Matthews, J. N., Roselli, D. S., Costello, A. M., Wang, R. Y., and Anderson, T. E. Improving the Performance of Log-Structured File Systems with Adaptive Methods. In *Proc. of the 16th ACM Symposium on Operating Systems Principles* (October 1997), pp. 238–251.

[19] McKusick, M., Joy, W., Leffler, S., and Fabry, R. A fast file system for UNIX. *ACM Transactions on Computer Systems 2*, 3 (Aug. 1984), 181–197.

[20] Ng, S. W. Improving disk performance via latency reduction. *IEEE Transactions on Computers 40*, 1 (January 1991), 22–30.

[21] Patterson, D., Gibson, G., and Katz, R. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *International Conference on Management of Data* (June 1988), pp. 109–116.

[22] Patterson, R. H., Gibson, G. A., Ginting, E., Stodolsky, D., and Zelenka, J. Informed Prefetching and Caching. In *Proceedings of the ACM Fifteenth Symposium on Operating Systems Principles* (December 1995).

[23] Perl, S. E., and Sites, R. L. Studies of Windows NT Performance Using Dynamic Execution Traces. In *Proc. of the Second Symposium on Operating Systems Design and Implementation* (October 1996), pp. 169–184.

[24] Ruemmler, C., and Wilkes, J. UNIX Disk Access Patterns. In *Proc. of the Winter 1993 USENIX* (Jan. 1993), pp. 405–420.

[25] Ruemmler, C., and Wilkes, J. An Introduction to Disk Drive Modeling. *IEEE Computer 27*, 3 (March 1994), 17–28.

[26] Seltzer, M., Chen, P., and Ousterhout, J. Disk Scheduling Revisited. In *Proc. of the 1990 Winter USENIX* (Jan. 1990), pp. 313–323.

[27] Transaction Processing Performance Council. *TPC Benchmark B Standard Specification*. Waterside Associates, Fremont, CA, Aug. 1990.

[28] Transaction Processing Performance Council. *TPC Benchmark C Standard Specification*. Waterside Associates, Fremont, CA, August 1996.

[29] Wang, R. Y., Anderson, T. E., and Patterson, D. A. Virtual Log Based File Systems for a Programmable Disk. In *Proc. of the Third Symposium on Operating Systems Design and Implementation* (February 1999).

[30] Wilkes, J., Golding, R., Staelin, C., and Sullivan, T. The HP AutoRAID Hierarchical Storage System. In *Proc. of the 15th ACM Symposium on Operating Systems Principles* (December 1995), pp. 96–108.

[31] Wood, D. A., and Hill, M. D. Cost-effective Parallel Computing. *IEEE Computer 28*, 2 (February 1995), 69–72.

[32] Worthington, B. L., Ganger, G. R., Patt, Y. N., and Wilkes, J. On-Line Extraction of SCSI Disk Drive Parameters. In *Proc. of the 1995 SIGMETRICS* (May 1995).