Proceedings of the 1995 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems

Striping in a RAID Level 5 Disk Array

Peter M. Chen
Computer Science Division
Electrical Engineering and Computer Science Department
University of Michigan
pmchen@eecs.umich.edu

Edward K. Lee
Systems Research Center
Digital Equipment Corporation
eklee@src.dec.com

Abstract: Redundant disk arrays are an increasingly popular way to improve I/O system performance. Past research has studied how to stripe data in non-redundant (RAID Level 0) disk arrays, but none has yet been done on how to stripe data in redundant disk arrays such as RAID Level 5, or on how the choice of striping unit varies with the number of disks. Using synthetic workloads, we derive simple design rules for striping data in RAID Level 5 disk arrays given varying amounts of workload information. We then validate the synthetically derived design rules using real workload traces to show that the design rules apply well to real systems.

We find no difference in the optimal striping units for RAID Level 0 and 5 for read-intensive workloads. For write-intensive workloads, in contrast, the overhead of maintaining parity causes full-stripe writes (writes that span the entire error-correction group) to be more efficient than read-modify writes or reconstruct writes. This additional factor causes the optimal striping unit for RAID Level 5 to be four times smaller for write-intensive workloads than for read-intensive workloads.

We next investigate how the optimal striping unit varies with the number of disks in an array. We find that the optimal striping unit for reads in a RAID Level 5 varies *inversely* to the number of disks, but that the optimal striping unit for writes varies *with* the number of disks. Overall, we find that the optimal striping unit for workloads with an unspecified mix of reads and writes is *independent* of the number of disks.

Together, these trends lead us to recommend (in the absence of specific workload information) that the striping unit over a wide range of RAID Level 5 disk array sizes be equal to 1/2 * average positioning time * disk transfer rate.

1 Introduction

For the past several decades, input/output (I/O) performance has not kept up with advances in processing speeds. Disk arrays have been proposed and used to increase aggregate disk performance by ganging together multiple disks in parallel [Patterson88]. Redundant disk arrays add error-correcting codes to improve reliability. One popular type of redundant disk array interleaves data in block-sized units known as striping units and uses parity to protect against failures. This type of disk array is known as a RAID Level 5 (Figure 1) [Patterson88].

One of the most crucial parameters in designing a disk array is the size of the striping unit. The striping unit determines how a *logical* request from a user is broken up into *physical* disk requests to the disks. This distribution profoundly affects a disk array's performance—[Chen90] shows that performance of a non-redundant disk array (RAID Level 0) can vary by an order of magnitude depending on the striping unit.

Several papers address how to choose the striping unit in non-redundant disk arrays [Chen90, Lee93a, Scheuermann91], but none have yet examined how to choose it in a RAID Level 5[1]. The overhead of maintaining parity in a RAID Level 5 leads to additional factors in choosing the best striping unit. The goals of this paper are to

- quantify the effect these additional factors have in choosing the striping unit,
- create simple design rules for determining the best striping unit for a range of workloads, and
- quantify how the number of disks in the array affects the optimal striping unit.

Our key results, found by using synthetic workloads and verified by using real-world traces, are:

- Concurrency is the single most important feature of a workload. Knowing concurrency allows one to choose a single striping unit that achieves near-optimal performance for a range of request sizes. That striping unit is S * average positioning time * disk transfer rate * (concurrency - 1) + 1 sector, where S (with 17 disks) is approximately 1/4 for reads and 1/20 for writes.
- Even without knowing concurrency, one can still choose a striping unit that guarantees at least 50% of optimal performance for any concurrency and request size. That

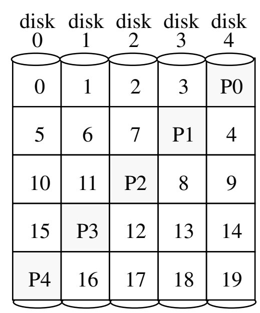


Figure 1: Data and Parity Layout in a RAID Level 5 Disk Array. This figures illustrates how data is distributed in a round-robin manner across disks in a RAID Level 5 disk array. Each number represents one data block; the size of each data block is called the striping unit. Parity blocks have the same size and are rotated between the disks. Each parity block contains the parity for the data blocks in its row. For example, P0 contains the parity computed over data blocks 0-3. The specific data distribution shown here (and used throughout the paper) is called the left-symmetric distribution [Lee93b].

1. [Gray90] describes a scheme called parity striping that has similar performance to a RAID Level 5 with infinitely large striping unit. [Gray90] does not investigate different striping units for RAID Level 5 disk arrays, however.

striping unit is Z * average positioning time * disk transfer rate, where Z (with 17 disks) is approximately 2/3 for reads and 1/6 for writes.

- As the number of disks in the array varies, the best striping unit for reads increases, but the best striping unit for writes decreases. These two trends effectively cancel and the best striping unit over workloads with any mix of reads and writes, any number of disks, and any concurrency is 1/2 * average positioning time * disk transfer rate.

The paper is organized as follows: Section 2 reviews previous research on disk array striping and describes qualitatively how RAID Level 5 disk arrays change the optimal striping unit. Section 3 describes the experimental setup that we use in this paper. In Sections 4 and 5, we use synthetic workloads to derive simple design rules for striping data in RAID Level 5 disk arrays given varying amounts of workload information and disks. We then validate the synthetically derived design rules in Section 6 using real workload traces to show that the design rules apply well to real systems.

2 Background and Previous Work

2.1 Non-Redundant Disk Arrays (RAID Level 0)

We define the *striping unit* as the amount of logically contiguous data stored on a single disk (Figure 1). A large striping unit will tend to keep a file clustered together on a few disks (possibly one); a small striping unit tends to spread each file across many disks.

Parallelism describes the number of disks that service a user's request for data. A higher degree of parallelism increases the transfer rate that each request sees. However, as more disks cooperate in servicing each request, more disks waste time positioning (seeking and rotating) and the efficiency of the array decreases. We define the degree of *concurrency* of a workload as the average number of outstanding user requests in the system at one time. A small striping unit causes higher parallelism but supports less concurrency in the workload; a large striping unit causes little parallelism but supports more concurrency in the workload.

Fundamentally, disk striping affects the amount of data that each disk transfers before re-positioning (seeking and rotating to the next request). This amount of data has a drastic impact on disk throughput. For a Seagate Elite3 disk, if a disk transfers one sector per random request, throughput will be 0.03 MB/s; if it transfers one track per request, throughput will be 1.79 MB/s; if it transfers one cylinder per request, throughput will be 4.16 MB/s. The choice of striping unit should maximize the amount of useful data each disk transfers per request and still make use of all disks. Large striping units maximize the amount of data a disk transfers per access but require higher concurrency in the workload to make use of all disks. Small striping units can make use of all disks even with low workload concurrency, but they cause the disks to transfer less data per access.

Chen and Patterson investigate these tradeoffs in a RAID Level 0 with 16 disks [Chen90]. They find that the choice of striping unit depends heavily on the workload concurrency and only minimally on the workload's average request size. If concurrency is known, they find that it is possible to choose a striping unit that guarantees good performance to workloads with that concurrency over a wide range of sizes. Good performance in this context means getting over 95% of the throughput at the optimal striping unit. Their choice of striping unit is expressed as S * (average disk positioning time) * (disk transfer rate) * (workload concurrency - 1) + 1 sector, where S is called the concurrency-slope coefficient and is found to be approximately 1/4 for a 16-disk disk array.

If concurrency is unspecified, a good choice for striping unit can still be made, although the performance guarantees are weaker than when concurrency is specified. Chen and Patterson find a good choice of striping unit to be Z * (average disk positioning time) * (disk transfer rate), where Z is called the zero-knowledge coefficient, since it applies when no workload information is given, and is found to be approximately 2/3 for a 16-disk disk array

Note that the above two equations can be re-written to depend on only one composite disk parameter: (average disk positioning time) * (disk transfer rate). [Chen90] calls this term the *disk performance product*.

In this paper, we extend the work of [Chen90] to RAID Level 5 disk arrays. Section 2.2 describes how the choice of striping unit in a RAID Level 5 differs from that in a RAID Level 0 due to the effect of maintaining parity when performing writes in a RAID Level 5.

Lee and Katz [Lee91, Lee93a] develop an analytic model of non-redundant disk arrays and derive an equation for the optimal striping unit to be

$$\sqrt{\frac{\text{Disk Performance Product} \times (\text{Concurrency} - 1) \times \text{Request Size}}{\text{Number Of Disks}}}$$

Taken together, [Chen90] and [Lee91, Lee93a] conclude that the striping unit depends on the disk performance product and concurrency. [Lee91, Lee93a] conclude that the optimal striping unit depends on request size, but [Chen90] shows that this dependence can be ignored with less than a 5% performance penalty.

Scheuermann, Weikum, and Zabback show moderate performance improvements by allowing different striping unit for different files [Scheuermann91, Weikum92]. This performance improvement comes at the cost of additional complexity and assumes a priori knowledge of the average request size and peak throughput of applications accessing each file.

2.2 Block-Interleaved, Parity Disk Arrays (RAID Level 5)

Like RAID Level 0, RAID Level 5 disk arrays interleave data across multiple disks in blocks called striping units (Figure 1). To protect against single-disk failures, RAID Level 5 adds a parity block for each row of data blocks. These parity blocks are distributed over all disks to prevent any single disk from becoming a bottleneck. The group of data blocks over which a parity block is computed is called a *parity group* or *stripe*. In Figure 1, data blocks 0-3 and parity block P0 compose a stripe.

Note that the distribution, or rotation, of parity blocks among the disks in the array is independent of the striping unit. The amount of contiguous parity on a disk is called the parity striping unit. All current disk arrays set the parity striping unit equal to the striping unit.

Reads in a RAID Level 5 are very similar to accesses (both reads and writes) in a RAID Level 0. Writes in a RAID Level 5, however, are quite different because they must maintain correct parity. There are three types of writes in a RAID Level 5, depending on how the new parity is computed.

- *Full-stripe writes*: writes that update all the stripe units in a parity group. The new parity value for the associated parity block is computed across all new blocks. For instance, in Figure 1, writing blocks 0-3 would be a full-stripe write. No additional read or write operations are required to compute parity, and hence full-stripe writes are the most efficient type of writes.
- *Reconstruct writes*: writes that compute parity by reading in the data from the stripe that are *not* to be updated. Parity is then computed over this data and the new data. For instance, writing data blocks 0-2 in Figure 1 could lead to a reconstruct write. The disk array software would read the values of data block 3, then compute parity over the new data (blocks 0-2) and data block 3. Reconstruct writes are less efficient than full-stripe writes because they perform more I/Os per byte written.
- *Read-modify writes*: writes that compute the new parity value by 1) reading the old data blocks from the disks to be updated, 2) reading the old parity blocks for the stripe, 3) calculating how the new data is different from the old data, and 4) changing the old parity to reflect these differences. For instance, writing block 0 in Figure 1 could be done as a read-modify write. To compute the new parity, the disk array software reads the old value of block 0 and the old value of parity block P0. It would then compute parity by seeing how the old value of block 0 was different from the new value and applying these differences to the old parity block to generate the new value of parity block P0. Read-modify writes are less efficient than full-stripe writes because they must read the old values of the data and parity blocks. Note that read-modify writes could be performed as reconstruct writes. We assume the system chooses the method that minimizes the total number of I/Os.

The method used to compute parity depends on the size of a request and its starting location. In general, *writes that span a larger fraction of the stripe are more efficient than writes that span a smaller fraction*. This factor tends to make the choice of striping unit for writes in a RAID Level 5 smaller than for reads. The paper uses simulation to study the effects of the above factors and determine guidelines for selecting the best striping unit.

3 Experimental Design

3.1 The Disk Model

We model several types of disks in our study. The default disks are IBM Lightning 3.5” disks [IBM89]. In order to maintain the same disk array capacity, we only use the first 300 MB of each disk. Thus the effective average seek time for the Seagate Elite3 disks is 3 times faster than if we had spread requests over the entire disk. All disks in the array are rotationally synchronized, that is, sector 0 passes under the read/write head at the same time for all disks. Table 1 summarizes the relevant parameters for the disks we model in this study.

3.2 The Workload

The primary workload we use in this paper is stochastic and characterized by three parameters: degree of concurrency, request size, and read/write mix. In Section 6 we verify our results using traces. We chose to primarily use a stochastic workload for the following reasons:

	IBM Lightning	**Seagate Elite3**	**Seagate Wren4**
Bytes Per Sector	512	512	512
Sectors Per Track	48	99	53
Tracks Per Cylinder	14	21	9
Cylinders Per Disk	949	2627	1549
Disk Capacity	311 MB	2667 MB	361 MB
Revolution Time	13.9 ms	11.1 ms	16.7 ms
Single Cylinder Seek Time	2.0 ms	1.7 ms	5.5 ms
Average Seek Time	12.5 ms	11.0 ms	16.5 ms
Max Stroke Seek Time	25.0 ms	22.5 ms	35.0 ms
Sustained Transfer Rate	1.8 MB/s	4.6 MB/s	1.6 MB/s
Effective (Simulated) Capacity	300 MB	300 MB	300 MB
Effective Average Seek Time (first 300 MB of disk)	12.2 ms	4.1 ms	14.9 ms
Effective Average Positioning Time	19.2 ms	9.7 ms	23.2 ms
Effective Disk Performance Product	33 KB	43 KB	37 KB

Table 1: Disk Model Parameters. Average seek time is the average time needed to seek between two equally randomly selected cylinders. Sustained transfer rate is a function of bytes per sector, sectors per track and revolution time. IBM Lightning is the IBM 0661 3.5” SCSI disk drive; Seagate Elite3 is the Seagate ST-43400N 3.5” SCSI disk drive; and Seagate Wren4 is the Seagate ST-4350N 5.25” SCSI disk drive. The seek profile for each disk is computed as seekTime(x) = a * sqrt(x-1) + b * (x-1) + c, where x is the seek distance in cylinders, and a, b, and c are chosen to satisfy the single-cylinder seek time, average seek time, and max-stroke seek time. The square root term in the above formula models the constant acceleration/deceleration period of the disk head and the linear term models the period after maximum disk head velocity is reached. We have compared the model to the seek profile of the Amdahl 6380A [Thisquen88] and IBM Lightning and have found the model to closely approximate the seek profile of the actual disk. We did not model zone-bit recording; instead, we assumed an average transfer rate over all cylinders.

- Our goal of synthesizing design rules as a function of workload required an extremely flexible workload. In particular, we needed to be able to independently vary each of three workload parameters: concurrency, request size, and read/write mix. A stochastic workload allowed us to easily explore many different workloads. Traces, in contrast, can provide a realistic snapshot of a system and workload, but their fixed characteristics would have unduly limited our study.
- One goal of this paper was to compare our choice of striping units for RAID Level 5 with the choice of striping units for RAID Level 0 from previous studies. Previous studies were done using stochastic workloads, so our use of a stochastic workload enabled a direct comparison.

Concurrency controls the load on the system by specifying the number of independent processes that are simultaneously issuing requests. The number of processes is fixed during each simulation run and each process repeatedly issues an I/O request, waits for its completion, and immediately issues the next request. We simulate concurrencies between 1 and 20. Unless the striping unit is much smaller than the average request size, concurrency 1 leaves most disks idle. On the opposite extreme, workloads with concurrency 20 keep all the disks busy.

Request size controls the amount of data for each logical access. We use the following five distributions of request size:

- exp4KB: An exponential distribution with a mean of 4 KB
- exp16KB: An exponential distribution with a mean of 16 KB
- norm100KB: A normal distribution with a mean and standard deviation of 100 KB
- norm400KB: A normal distribution with a mean and standard deviation of 400 KB
- norm1500KB: A normal distribution with a mean and standard deviation of 1500 KB

The exponential distributions model requests generated by time-sharing and transaction-processing applications and the normal distributions model requests generated by scientific applications. For both types of distributions, the request sizes generated are always a multiple of a kilobyte and are always at least a kilobyte in size; that is, zero and negative request sizes are not allowed. The means of the target distributions are adjusted so that the actual mean request size seen by the array is 4 KB, 16 KB, 100KB, 400 KB, or 1500 KB. In all cases, requests are aligned on kilobyte boundaries and are uniformly distributed throughout the disk array.

Other workload parameters are possible, such as disk skew, sequentiality, and burstiness. Skew between the disks of a round-robin interleaved array is likely to largely absorbed by higher-level caches if those caches can hold multiple striping units worth of data. The underlying sequentiality of a workload is included in the request size parameter—sequential disk accesses often are simply single, large logical requests that have been broken down (and can easily be recombined by a disk array controller). Burstiness can be thought of as a workload whose concurrency varies with time. During a burst, concurrency is high; between bursts, concurrency is low. We synthesize rules for choosing a single striping unit that achieves good performance for a mix of workloads of both high and low concurrency.

3.3 The RAID Simulator

Our study is conducted using a detailed simulator of the disk system. We opted for a simulation-driven study rather than a queueing-model study because read-modify writes and reconstruct writes in a RAID Level 5 create synchronization constraints that are very difficult to model analytically.

The RAID simulator, *raidSim*, is an event-driven simulator developed at Berkeley for modeling non-redundant and redundant disk arrays. It consists of a module for implementing a variety of RAID levels, a module for modeling disk behavior, and a module for generating synthetic I/O requests. The only resources modeled are disks. In particular, resources such as the CPU, disk controllers and I/O busses are ignored. This allows us to focus on the intrinsic performance effects of using multiple disks. *RaidSim* simulates each workload until the mean response time is within 5% of the true mean at a 95% confidence level. This typically requires simulating several thousand requests. The disks support request merging (as is provided by an on-disk cache [Ruemmler94]). That is, if a disk receives two consecutive requests two adjacent disk blocks, it services these requests as though they had been one large requests.

3.4 Metrics

Common disk system performance metrics are throughput and response time. With a fixed level of concurrency and a specific request size, higher throughput always leads to faster response time.[1] In this paper, we use throughput as the main performance metric. Most throughput values will be given as a percentage of the maximum throughput over all striping units. For example, given a particular workload, if the maximum throughput over all striping units is 10 MB/s, and a striping unit S yields a throughput of 3 MB/s, then the throughput for striping unit S will be given as 30% of maximum throughput.

4 Choosing the Striping Unit

Figure 2 and Figure 3 show the throughput versus the striping unit for reads and writes over a range of sizes and concurrencies. We vary the striping unit from 1 KB to 1 MB and make the following observations:

- At any fixed striping unit, the throughput increases with larger request sizes and higher degrees of concurrency. Increasing request sizes allow each disk to access more data per request; higher degrees of concurrency are able to make use of more disks.
- Reads perform uniformly better than writes due to the extra overhead of maintaining parity when writing data.

1. Little's Law implies that Throughput = Average_Request_Size * Average_Concurrency/ Response_Time, so response time can be easily calculated for each workload from its throughput [Denning78].

In order to compare trends from different workloads more easily, we scale the throughput of each workload, expressing it as a percentage of the maximum throughput (Figure 4 and Figure 5).

If one knows the parameters of a workload, that is, the request size distribution, concurrency, and read/write mix, one can use Figure 4 and Figure 5 to choose the striping unit which maximizes throughput. However, in most systems, the exact workload is not known. One of, or possibly both, the request size and the concurrency will be unspecified. Thus, it is desirable to be able to choose a good striping unit with as little knowledge about the workload as possible. In this paper, we will strive to maximize the *minimum percentage throughput over a range of workloads*. In other words, we wish to guarantee the highest percentage of maximum throughput to all workloads in consideration. For example, if the concurrency is known to be 4 but the request size is unknown, we can use Figure 4b to choose a striping unit. In Figure 4b, over the range of request sizes, the striping unit that maximizes the minimum percentage throughput is 40 KB. At that striping unit, all workloads considered yield at least 98% of their maximum possible throughput. As in [Chen90], we find that knowing the workload concurrency is more important than knowing request size. That is, it is possible to guarantee a higher percentage of maximum throughput if we only know the workload concurrency than if we only know request size.

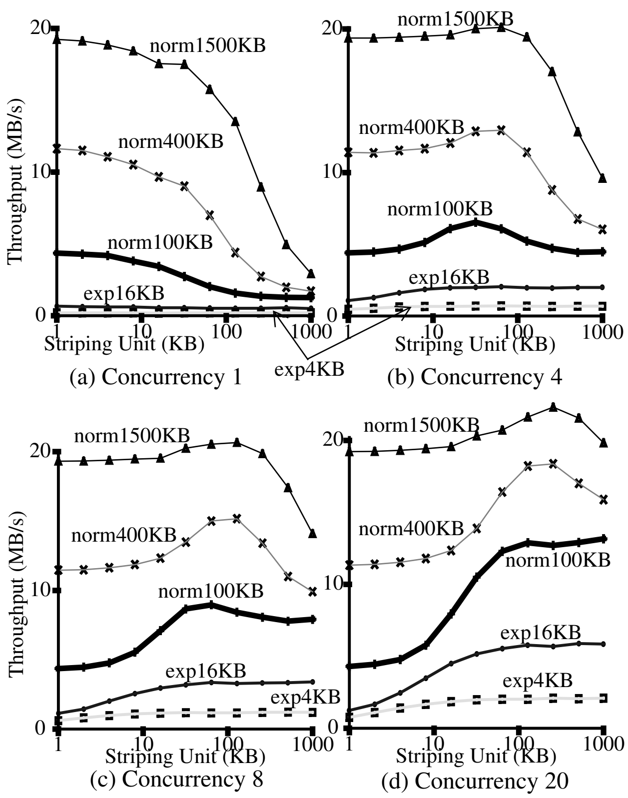


Figure 2: Read Throughput for a Range of Sizes and Concurrencies. Throughput is shown as a function of striping unit. The throughput increases as request sizes or concurrencies increase, as expected. These results are for 17 IBM Lightning disks.

4.1 Read Workloads

To verify previous results we start by focusing on performance of reads in a RAID Level 5. As in [Chen90], if we know the concurrency of a read workload, it is possible to choose a striping unit that guarantees over 95% of the maximum throughput for any request sizes. In fact, there is a *range* of striping units that guarantee 95% of maximum throughput to workloads with any request size. Figure 6a shows the range of these striping units for read workloads as vertical lines delineated by + signs. Note that higher concurrencies lead to larger striping units. This is expected from the discussion in Section 2.2—higher concurrencies inherently use all disks, so we can then use larger striping units to maximize the amount of data each disk transfers per logical request.

We can express our choice of striping unit as a linear function of workload concurrency by 1) fixing the striping unit choice for a concurrency of one at 1 sector (0.5 KB), and 2) measuring the minimum slope of any striping unit vs. concurrency line that lies entirely in the displayed range.[1] Our choice of striping unit in Figure 6 can then be expressed

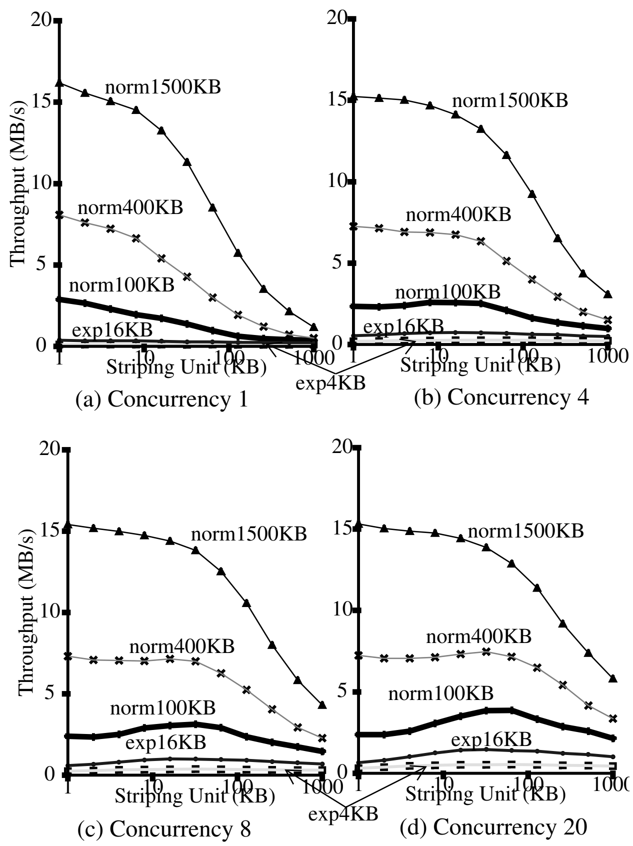


Figure 3: Write Throughput for a Range of Sizes and Concurrencies. Throughput is shown as a function of striping unit. The throughput increases as request sizes or concurrencies increase, as expected. These results are for 17 IBM Lightning disks. Note that write throughput is uniformly lower than read throughput (Figure 2), due to the extra overhead of maintaining parity in a RAID Level 5.

as

$$\text{Striping Unit} = \text{Slope} \times (\text{Concurrency} - 1) + 1 \text{ sector}$$

The Slope term can be expressed as Sr * (disk performance product), where S_r is called the *concurrency-slope coefficient* and can be shown to be independent of disk technology (Table 2). For 17 IBM Lightning disks, the slope of this line is 7.3 KB and hence S_r is 0.22. This matches with results in [Chen90].

Next let us consider the case where we know the workload is 100% reads but have no information about the concurrency or request size. In this case we can choose a good compromise striping unit by superimposing all the graphs in Figure 4 and using the striping unit that guarantees the highest percentage of maximum performance to all workloads. As in [Chen90] we express this choice as

$$\text{Striping Unit} = Z_r \times \text{Disk Performance Product}$$

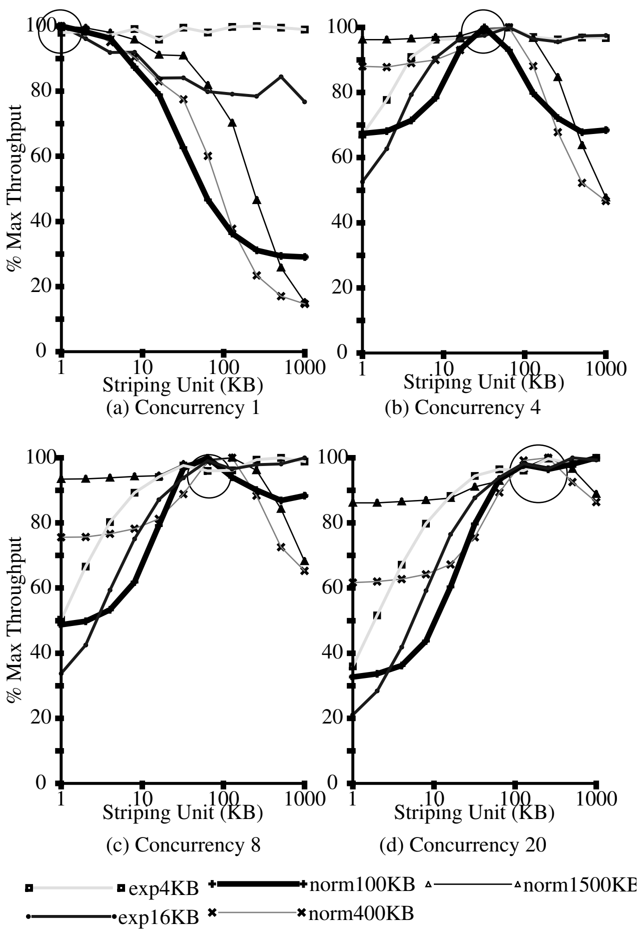


Figure 4: Read Performance for a Range of Sizes and Concurrencies, Shown as Percentage of Maximum Throughput. Percentage of maximum throughput is shown as a function of the striping unit. The circled point on each graph indicates the striping unit that guarantees the highest percentage of maximum throughput to all workloads shown on that graph. Note that increasing concurrency leads to larger optimal striping units.

1. The minimum slope was chosen rather than the mean slope to enhance load balancing among disks.

For 17 IBM Lightning disks, the compromise striping unit for reads is 24 KB and hence Z_r is 0.73. This striping unit guarantees at least 70% of the maximum performance to workloads with any request size or concurrency. Again, this matches closely to previous results. Table 2 lists results for different disk types and shows that Z_r and S_r are independent of disk technology.

4.2 Write Workloads

Now that we have verified that reads in a RAID Level 5 using the left-symmetric parity placement [Lee93b] lead to identical striping unit choices as I/Os in a non-redundant disk array, we turn our attention to writes in a RAID Level 5. From the discussion in Section 2.2, we expect that the best striping unit for write-intensive workloads will be smaller than those for read-intensive workloads.

Figure 6 shows the range of optimal striping units for writes as vertical bars delineated by • signs. For write-inten-

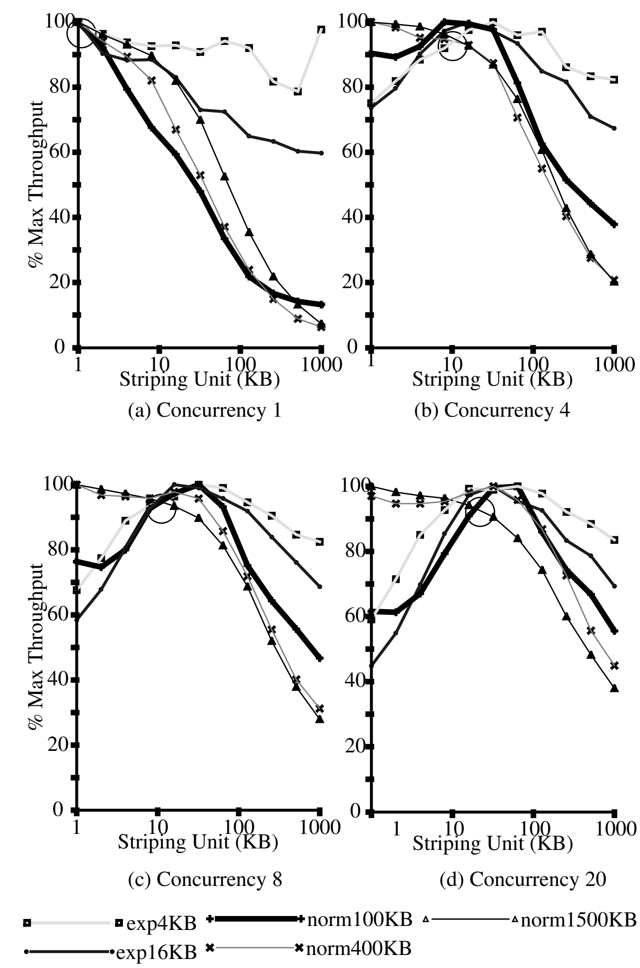
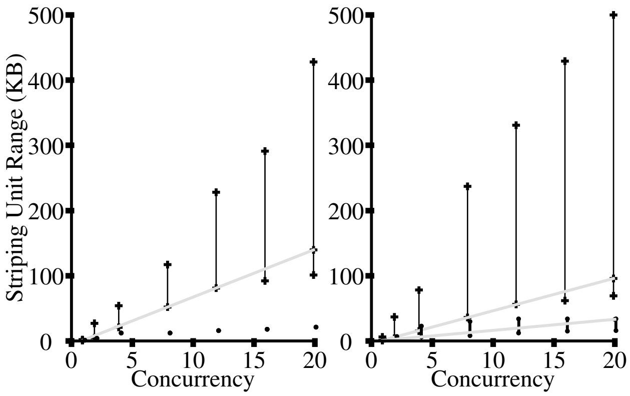


Figure 5: Write Performance for a Range of Sizes and Concurrencies, Shown as Percentage of Maximum Throughput. Percentage of maximum throughput is shown as a function of the striping unit. The circled point on each graph indicates the striping unit that guarantees the highest percentage of maximum throughput to all workloads shown on that graph. Note that the optimal striping unit for write-intensive workloads is smaller than for the read-intensive workloads in Figure 4.

sive workloads, it is not always possible to choose a striping unit for a specific concurrency that yields 95% performance for all request sizes. In Figure 6a, we show the striping unit choice that yields the best performance for all striping units for writes by • signs. These striping unit choices are almost as good as for read-intensive workloads—they always yield performance of at least 93%.

Note that the range of optimal striping units in Figure 6 is uniformly lower for writes than for reads, as expected. Writes perform better when striped across more disks, since



(a) 95% Performance Criterion (b) 90% Performance Criterion

Figure 6: Striping Unit Chosen Versus Concurrency—17 IBM Lightning Disks. Shown here is the range of striping units which yield at least 95% (Figure a) or 90% (Figure b) of the maximum throughput for all request sizes (exp4KB, exp16KB, norm100KB, norm400KB, norm1500KB) for reads and writes. The range for a workload with 100% reads is wider and marked by + signs. The range for workloads with 100% writes is much narrower and marked by • signs. Write workloads are not always able to attain 95% performance, so Figure a shows the striping unit that yields the highest performance for all workloads (which is always at least 93%). The dashed lines are the lines with smallest slope that lie entirely in the striping unit range.

	Read Workloads		Write Workloads		Read and Write Workloads
Disk Type	S_r	Z_r	S_w	Z_w	Z_{rw}
17 Lightnings	.22	.73	.051	.18	.48
17 Elite3	.30	.67	.066	.19	.58
17 Wren4	.22	.68	.051	.17	.46
Average	.25	.69	.056	.18	.51

Table 2: Values of Concurrency-Slope and Zero-Knowledge Coefficients for 3 Disk Types. This table shows the values of the various coefficients useful in choosing the striping unit. Over three very different disks, the values of each coefficient are relatively independent of disk type.

they then require less accesses to compute the new parity. Smaller striping units thus lead to more efficient writes; this makes the range of optimal striping units lower for write-intensive workloads than the range for read-intensive workloads. Also note that even very high concurrency workloads do not lead to very large optimal striping units. This ceiling on the optimal striping unit arises because writes yield better performance when they utilize all the disks by issuing full-stripe writes rather than by utilizing the workload concurrency. Of course, this must be balanced by the fact that the disk array is more efficient when a disk transfers a larger amount of data for a logical request.

By lowering our criterion of good performance to 90% (Figure 6b), we can again use a simple linear equation to describe our choice of striping units at each concurrency. The lower, dashed line in Figure 6b describes the following equation for striping unit:

$$S_w \times \text{Disk Performance Product} \times (\text{Concurrency} - 1) + 1 \text{ sector}$$

where S_w is the concurrency-slope coefficient for writes and is equal to 0.051. Thus the slope for writes is approximately four times shallower than the slope for reads. Table 2 shows that the values of S_w for different disks is relatively constant.

If no information is given about concurrency or request size, we can choose a good compromise striping unit for write-intensive workloads as we did for reads. We superimpose all graphs in Figure 5 and choose the striping unit that guarantees the highest percentage of maximum performance to all workloads. For write-intensive workloads on 17 IBM Lightning disks, this compromise striping unit is 6 KB, so Z_w, which is the optimal striping unit divided by the disk performance product for IBM Lightning disks, is 0.18. A 6 KB striping unit guarantees at least 73% of maximum performance to all workloads tested. We tabulate the values of Z_w using other disk technologies and find that Z_w is independent of disk type.

Without knowledge of the read/write mix of the workload, it is still possible to choose a striping unit that performs reasonably well for all workloads. By superimposing the graphs in Figure 4 and Figure 5, we can find the striping unit that guarantees the best performance to all read or write workloads of arbitrary concurrency and request size. This striping unit is 16 KB and guarantees 60% of maximum throughput to workloads with any request size, concurrency, and mix of reads and writes. We express this choice in terms of the disk-performance product as

$$\text{Striping Unit} = Z_{rw} \times \text{Disk Performance Product}$$

where Z_{rw} is 0.48. Thus, for a 17-disk disk array and in the absence of any workload information, we recommend a striping unit of approximately half the product of the average positioning time and the single-disk transfer rate.

5 Varying the Number of Disks

We have expressed our recommendations for striping unit for a 17-disk disk array in terms of coefficients S and Z and the disk performance product. Disk arrays with more or less than 17 disks will have different values for these coefficients, because the number of disks affects the tradeoffs discussed in Section 2.2. With more disks, smaller striping units are needed to ensure that all disks are used. Conversely, with fewer disks, each disk is more precious, leading to larger striping units to avoid tying up too many disks. Thus we expect that the coefficients will vary inversely to the number of disks in the array. Figure 7 shows that Z_r does indeed vary inversely to the number of disks. [Lee91, Lee93a] predict that the striping unit will vary inversely to the square root of the number of disks. We find empirically that the optimal striping unit for read-intensive workloads varies inversely to the cube root of the number of disks.

Surprisingly, Z_w *increases* as the number of disks increases. This trend means that as the number of disks in the array decreases, the optimal striping unit also decreases and each write request is best striped over *more* disks. To understand this, recall that limiting a write request to a few disks causes more read-modify writes. These read-modify writes cause extra read operations to the parity and data blocks, which increases disk utilization by a factor of approximately 2. In a disk array with less disks, this increased disk utilization ties up a higher fraction of the disks and severely hinders performance. Thus, the optimal striping unit is smaller with less disks to cause more full-stripe writes and hence decrease the number of read-modify writes.

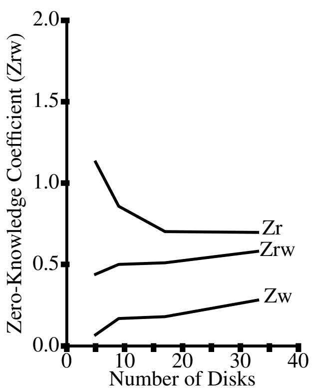


Figure 7: How Striping Units Vary with Number of Disks. This figure shows that the zero-knowledge coefficients Z_r, Z_w, and Z_{rw} vary differently with the number of disks. Since the striping unit is directly proportional to Z, the best striping unit varies in the same way as the various Z coefficients. For read-intensive workloads, Z_r decreases as the number of disks increases because smaller striping units are needed to ensure that all disks are used. For write-intensive workloads, Z_w decreases as the number of disks decreases because the extra overhead of read-modify writes becomes prohibitive when there are less disks. These two trends effectively cancel each other out, and so the overall choice of striping unit without knowing the read/write mix is approximately 0.5 and is independent of the number of disks. The optimal striping units using the values of Z_{rw} shown here guarantees approximately 60% of maximum throughput to all workloads. For read-intensive workloads, a striping unit derived from Z_r guarantees approximately 70% of maximum throughput for all workloads. For write-intensive workloads, a striping unit derived from Z_w guarantees approximately 70% of maximum throughput.

The combination of these trends leads to a fairly constant value for Z_{rw} at 0.5. Thus, without any workload knowledge about request size, concurrency, or mix of reads and writes, one should choose a striping unit equal to half the disk performance product. For our three disk types (using the true seek time, not the shortened seek time of the first 300 MB of each disk), this leads to striping units of 18 KB for the IBM Lightning disks, 38 KB for the Seagate Elite3 disks, and 20 KB for the Seagate Wren4 disks.

6 Verification Using Traces

In this section, we use I/O traces from real systems to validate the design rules derived using synthetic workloads, verifying that our recommended striping unit performs well under specific workloads. The traces we use come from Miller and Katz's supercomputing workload study and capture the I/Os from a range of climate modeling, aerodynamic, turbulence, and structural dynamics applications [Miller91]. The traces are at the file system level and did not capture the disk location of each access, so we processed the traces using a simple, extent-based file allocation. Each file is allocated as a single, contiguous extent in the disk array's logical address space. Concurrencies greater than one are simulated by running multiple instantiations of the same trace at the same time, using disjoint file spaces and starting each trace after a random delay.

We first simulated each trace separately on an array of 17 Seagate Elite3 disks and found that they fell into one of two main categories. Traces in the first category, represented by *venus* (simulation of venus' atmosphere), perform large (approximately 400 KB) I/Os. Traces in the second category, represented by *ccm* (community climate model), perform relatively small (16-32 KB), highly sequential I/Os. The vertical lines in Figure 8-Figure 10 show the striping unit we recommend if no workload information is given (21 KB for the Seagate Elite3 disks with the 300 MB capacity limit).

Figure 8 shows that workloads with all large I/Os perform best with very small striping units over all concurren-

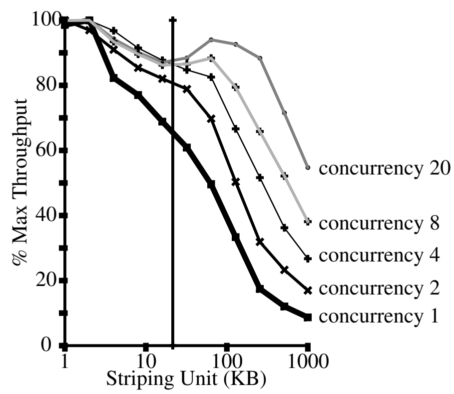


Figure 8: Performance of venus Trace. For a trace with predominantly large I/Os, small striping units perform best over a range of concurrencies. The vertical shows the striping unit we recommend without any advance workload information. Though it guarantees reasonable performance to all concurrencies (66%), it can certainly be improved upon with more specific workload knowledge.

cies. This agrees with the workloads with large requests in Figure 4 and Figure 5. Our recommendation of 21 KB performs reasonably well, at least 66% of the maximum possible performance for each concurrency, though knowing in advance that the request size is large would certainly make it possible to achieve higher performance.

Figure 9a shows that no single striping unit performs well for workloads with relatively small, sequential I/Os. Our recommendation yields close to the best possible performance, but this only guarantees about 30% of the maximum possible performance to each concurrency. This low performance guarantee is due to the workload with concurrency one. Because most of the I/Os in the trace were sequential, a workload with only one process can achieve much better performance by using a small striping unit. With a small striping unit, all disks always seek and transfer in parallel. Because the workload is sequential, using small striping units does not cause extra seeks. In contrast, when one process runs on a disk array with large striping units, a single disk seeks and transfers the data, then another disk seeks and transfers the next request, etc. In essence, having a small striping unit allows the disk array to overlap the seeks for all 17 disks, even with only one process issuing I/Os. This phenomenon does not occur at concurrencies higher than one, since multiple, interleaved processes disrupt the workload sequentiality and prevent the disks from servicing more than one request after each seek.

There are several ways to prevent this sharp performance drop-off for workloads with concurrency one.[1] The best way, though often impractical, is to modify the application or the operating system to group multiple I/Os together into a single larger I/O (a form of prefetching). With larger requests, disk arrays with large striping units can achieve the same disk parallelism that arrays with small striping units can. An easier solution that does not require modifying the application is to have all idle disks seek to the same position as the active disk. This overlaps the seeks, but not the transfers, of multiple disks. Figure 9b shows the performance resulting from grouping multiple sequential requests into a single, large one for one process. We see that our recommended striping unit is able to perform much better here, guaranteeing 80% performance to all concurrencies.

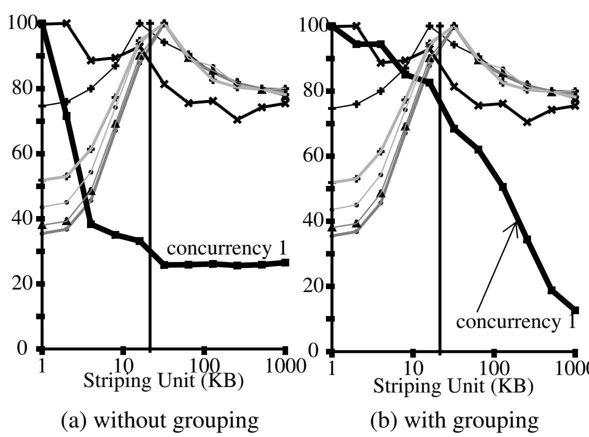


Figure 9: Performance of ccm Trace. Figure a shows the performance of the ccm trace, comprised mostly of relatively small (32 KB), sequential requests. The workload with concurrency one achieves much higher performance with small striping units because they allow the disks to service multiple, sequential requests without seeking. Figure b shows the same workload after grouping is used to combine multiple, sequential requests into a few large ones. In Figure a, no single striping unit achieves good performance for all workloads, though our recommended striping unit (shown as a vertical line) does about as well as any other. In Figure b, our recommended striping unit achieves almost 80% of the best possible performance for each individual workload.

Finally, we merged all the traces into a single composite trace and explored how the composite trace performed under different concurrencies. Figure 10 shows that, similar to Figure 9a, no single striping unit performs well for all concurrencies. Again, concurrency one performs differently than other concurrencies, but for a different reason. Mixing the traces removes all sequentiality between requests, but small striping units still perform significantly better than large ones for concurrency one. The reason is that performing only full striped I/Os allows the disk arms to seek in unison, even when not performing sequential requests. This is faster than synchronizing 17 randomly positioned disk arms and improves performance. No workload transformation can change this phenomenon—it is simply a benefit of very small striping units (RAID Level 3).

Figure 10 shows that if you place high value on workloads with concurrency one, choose a small striping unit. However, if you value workloads with both high and low concurrencies, then our recommendation of half the disk performance product performs as well as any other striping unit. And if you do not value concurrency one workloads, this paper's recommended striping unit provides excellent

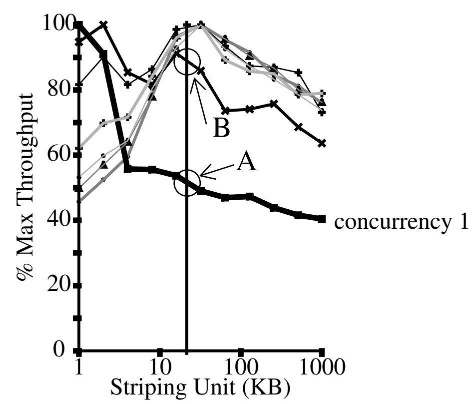


Figure 10: Performance of Composite Trace. As for the ccm trace, no single striping unit guarantees good performance for all concurrencies, though our recommended striping unit (21 KB) performs as well as any other (circle A). For concurrency one, the best striping unit is small. For all others, our recommended striping unit guarantees at least 85% of the best performance for each workload (circle B).

1. Note that workloads that leave many disks idle are poor candidates to run on disk arrays.

performance to all workloads, guaranteeing at least 85% of the best possible performance. These results agree with the major results of Section 4: 1) knowing concurrency enables one to choose a near-optimal striping unit, and 2) without knowing workload concurrency, the best striping unit is approximately 1/2 * average positioning time * disk transfer rate (21 KB for the 300 MB capacity Seagate Elite3 disks).

7 Conclusions

We have investigated how to stripe data in a RAID Level 5 disk array for a variety of workloads, both synthetic workloads and application traces. We found that reads in a RAID Level 5 behaved similarly to reads and writes in a non-redundant disk array. The resulting optimal striping unit for reads traded off two factors: 1) making use of all the disks in the array and 2) having each disk transfer as much as possible before repositioning. Write-intensive workloads, on the other hand, achieved optimal performance at smaller striping units (1/4 the size for a 17-disk disk array) on RAID Level 5 than on RAID Level 0. This was because full-stripe writes performed better than read-modify or reconstruct writes. Smaller striping units led to more full-stripe writes and correspondingly better performance.

For both reads and writes in a RAID Level 5, concurrency was the most important workload factor in choosing an optimal striping unit. The optimal striping unit increased with concurrency for both reads and writes, but the rate of increase was approximately four times more gradual for writes than for reads. Again, this was due to full-stripe writes performing better than read-modify or reconstruct writes. We also noted that even infinite levels of concurrency did not lead to arbitrarily large striping units for writes. This ceiling on the striping unit for writes occurred because writes “prefer” to utilize all the disks by issuing full-stripe writes rather than by depending on the workload’s concurrency.

Next, we investigated the effect of varying the number of disks in a RAID Level 5. We found that the striping unit for read-intensive workloads decreased as the number of disks increased, but that the striping unit for write-intensive workloads increased as the number of disks increased. Both these trends sought to lower the number of disks each request occupied as the number of disks in the array decreased. The combined effect of these opposing trends was that, without any workload knowledge of the read/write mix, the optimal striping unit stayed fairly constant at half the disk performance product. Thus, in the absence of any workload knowledge, we recommend a striping unit of 1/2 * average positioning time * disk transfer rate for RAID Level 5 disk arrays with any number of disks.

Last, we validated our main design rule by using I/O traces gathered from a range of scientific workloads. As found using stochastic workloads, specific workloads can achieve their best performance by using a carefully tailored striping unit. This was especially true for workloads with very large requests or concurrency one. For systems that run a range of workloads, the traces verified that our simple design rule accurately selects a near-optimal striping unit.

8 References

[Chen90] Peter M. Chen and David A. Patterson. Maximizing Performance in a Striped Disk Array. In *Proceedings of the 1990 International Symposium on Computer Architecture*, pages 322–331, May 1990.

[Denning78] Peter J. Denning and Jeffrey P. Buzen. The Operational Analysis of Queueing Network Models. *ACM Computing Surveys*, 10(3), September 1978.

[Gray90] Jim Gray, Bob Horst, and Mark Walker. Parity Striping of Disc Arrays: Low-Cost Reliable Storage with Acceptable Throughput. In *Proceedings of the 16th Very Large Database Conference*, pages 148–160, 1990. VLDB XVI.

[IBM89] IBM 0661 Disk Drive Product Description–Model 371. Technical report, IBM, July 1989.

[Lee91] Edward K. Lee and Randy H. Katz. An Analytic Performance Model of Disk Arrays and its Applications. Technical Report UCB/CSD 91/660, University of California at Berkeley, 1991.

[Lee93a] Edward K. Lee and Randy H. Katz. An Analytic Performance Model of Disk Arrays. In *Proceedings of the 1993 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 98–109, May 1993.

[Lee93b] Edward K. Lee and Randy H. Katz. The Performance of Parity Placement in Disk Arrays. *IEEE Transactions on Computers*, 42(6), June 1993.

[Miller91] Ethan L. Miller and Randy H. Katz. Input/Output Behavior of Supercomputing Applications. In *Proceedings of Supercomputing 1991*, pages 567–576, November 1991.

[Patterson88] David A. Patterson, Garth Gibson, and Randy H. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *International Conference on Management of Data (SIGMOD)*, pages 109–116, June 1988.

[Ruemmler94] Chris Ruemmler and John Wilkes. An Introduction to Disk Drive Modeling. *IEEE Computer*, pages 17–28, March 1994.

[Scheuermann91] Peter Scheuermann, Gerhard Weikum, and Peter Zabback. Automatic Tuning of Data Placement and Load Balancing in Disk Arrays. *Database Systems for Next-Generation Applications: Principles and Practice*, 1991. DBS-92-91.

[Thisquen88] Jean Thisquen. Seek Time Measurements. Technical report, Amdahl Peripheral Products Division, May 1988.

[Weikum92] Gerhard Weikum and Peter Zabback. Tuning of Striping Units in Disk-Array-Based File Systems. In *Proceedings of the 2nd International Workshop on Research Issues on Data Engineering: Transaction and Query Processing*, pages 80–87, 1992.