Performance Consequences of Parity Placement in Disk Arrays

Edward K. Lee and Randy H. Katz

Computer Science Division
Electrical Engineering and Computer Science
University of California at Berkeley
Berkeley, California 94720

Abstract

Due to recent advances in CPU and memory system performance, I/O systems are increasingly limiting the performance of modern computer systems. Redundant Arrays of Inexpensive Disks (RAID) have been proposed by Patterson et. al. [10] to meet the impending I/O crisis. RAIDs substitute many small inexpensive disks for a few large expensive disks to provide higher performance (both transfer rate and I/O rate), smaller footprints and lower power consumption at a lower cost than the large expensive disks they replace. Unfortunately, with so many small disks, media availability becomes a serious problem. RAIDs provide high availability by using parity encoding of data to survive disk failures. As will be shown by this paper, the way parity is distributed in a RAID has significant consequences for performance. In particular, we show that for relatively large request sizes of hundreds of kilobytes, the choice of parity placement significantly affects performance (up to 20-30 percent for the typical disk array configurations that are common today) and propose properties that are generally desirable of parity placements.

1 Motivation

In recent years, improvements in CPU and memory system performance has greatly outpaced improvements in I/O performance. If the trend continues, future improvements in CPU and memory system performance will be wasted as computer systems become increasingly I/O bound. To overcome the impending I/O crisis, Patterson et. al. have proposed Redundant Arrays of Inexpensive Disks (RAID) [3,9,10].

RAIDs substitute many small inexpensive disks for a few large expensive disks to provide higher performance (both transfer rate and I/O rate), smaller footprints and lower power consumption at a lower cost than the large expensive disks they replace. Unfortunately, with so many small disks, media availability becomes a serious problem. RAIDs provide high availability by using parity encoding of data to survive disk failures. Patterson [10] and Chen [1] define six different RAID organizations:

RAID level 0 Non-redundant disk array. Only data striping is supported.

RAID level 1 Mirrored disk array. Data is duplicated for reliability.

RAID level 2 Hamming-coded disk array.

RAID level 3 Parity-protected disk array with byte-interleaved data. Reads access all disks except the parity disk and writes access all disks. Only one I/O request may be serviced per parity disk at a time.

RAID level 4 Parity-protected disk array with block-interleaved data. Small reads access a single data disk and small writes access a data disk and a parity disk. Several reads and a single write per parity disk may be serviced concurrently.

RAID level 5 Parity-protected disk array with block-interleaved data and distributed parity. Similar to RAID level 4 except that the parity is distributed across all disks. Several reads and writes per parity disk may be serviced concurrently.

This paper will investigate the performance implications of different ways of distributing parity (*parity placements*) in RAID level 5 disk arrays.

Despite the fact that several papers have been written about RAID level 5 disk arrays, none of them has investigated the performance of alternative parity placements. Most of the papers have either been unaware of the performance consequences of parity placements or have assumed that all parity placements result in equivalent performance.[1] The result has been a dissemination of ad-hoc parity placements in the literature, encouraging the adoption of arbitrary parity placements by commercial vendors without performance justifications. This is regrettable given the small difference in complexity of implementing one parity placement over another. Thus, if one initially knew the performance trade-offs involved in selecting parity placements, immediate performance benefits could be achieved at negligible cost. The purpose of this paper is to motivate the importance of parity placements, and to raise the level of "parity placement consciousness" everywhere. Specifically, we will shows that at relatively large request sizes,[2] the choice of parity placement results in significant differences in both read and write I/O performance (20-30 percent for the typical disk array configurations that are common today). Based on our studies, we will also propose properties that are desirable of parity placements in general.

2 Overview

This section introduces concepts and defines terms that will be used throughout the paper.

2.1 Data Mapping

Ignoring for now the placement of parity, the following data mapping entities are considered. Figure 1 illustrates each mapping entity.

The **stripe unit** is the unit of data interleaving, the amount data that is placed on a given disk before placing data on a different disk.

The **parity stripe** is the minimal collection of stripe units over which parity is computed.

The **row** is the minimal collection of disks over which a parity stripe can be placed.

The **column** is the collection of disks formed by taking one disk from each row. Note that a disk may not belong to more than one column.

[1] Dibble [2] has investigated parity placements under the restriction that all writes are small. A large write is, therefore, broken into many independent small writes. We do not assume such a restriction.

[2] Ethan Miller [8], a member of our research group, is currently performing studies of supercomputing I/O traces which indicate that I/O requests of approximately 200KB in size are common.

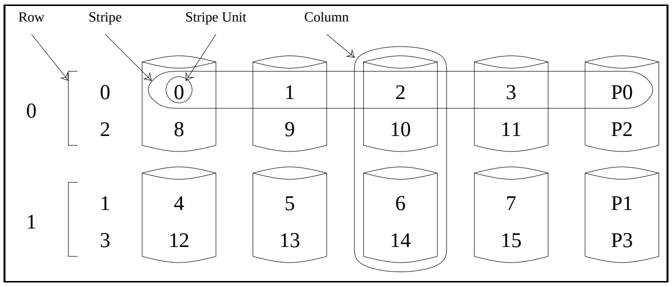

Figure 1: Data Mapping. This figure illustrates the data mapping entities, *stripe unit*, *stripe*, *row* and *column* in the context of a RAID level 4 disk array.

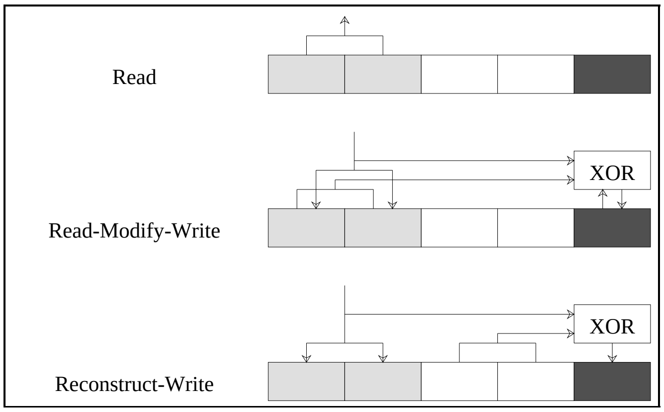

Figure 2: I/O Request Servicing. Each rectangle corresponds to a stripe unit. The lightly shaded regions correspond to data that is to be read or written. The dark regions correspond to parity.

2.2 I/O Request Servicing

Figure 2 illustrates the ways in which I/O requests to RAID level 4 and RAID level 5 disk arrays are serviced when there are no failed disks. A read-modify-write computes the new parity by xoring the new data, old data and old parity. A reconstruct-write computes the new parity by xoring the new data and the data from the part of the parity stripe that is not being written.

3 Definition of Parity Placements

The number of different ways parity can be placed relative to data is astronomical. We will begin our investigation of parity placements by requiring that they satisfy the following properties:

- Stripe units belonging to the same parity stripe should not map to the same column. (In many RAID systems, the disks within a column have

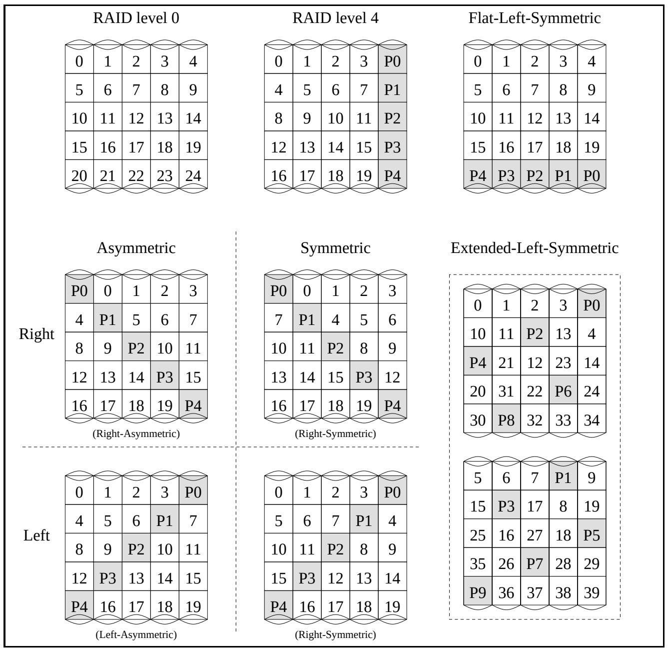

Figure 3: Parity Placements. Each square corresponds to a stripe unit. Each column of squares corresponds to a disk. Each matrix corresponds to a single row of disks. In all placements, $P0$ computes parity over stripe units $0, 1, 2$ and 3; $P1$ computes parity over stripe units $4, 5, 6$ and 7; etc. In all cases except the RAID level 0 and RAID level 4 placements, the minimum repeating pattern is shown.

a common failure mode, the string interface for example.) This is referred to as the "orthogonal RAID" property and guarantees that the failure of a single column does not result in data unavailability [12].

- In a RAID with n stripe units per parity stripe, the ith parity stripe unit should correspond to logical stripe unit j such that $j \operatorname{div} n = i$. This guarantees that the parity for any write request that is aligned on a parity stripe and a parity stripe in size can be computed by using only the data being written without having to read old data.

Figure 3 illustrates eight parity placements which satisfy the above properties. Lee [6] discusses why these are the most interesting parity placements; we simply take the parity placements as given. The following sections give an informal description for each parity placement.

3.1 RAID level 0

The RAID level 0 placement results in the conventional modulo n data striping scheme [4,5,7,11]. The RAID level 0 placement is defined here for comparison purposes only.

3.2 RAID level 4

The RAID level 4 placement is derived from the RAID level 0 placement by adding a parity disk to each row. The read performance of a RAID level 4 placement with n disks per row is identical to the read performance of a RAID level 0 placement with $n-1$ disks per row. Because the RAID level 0 placement is widely used and better understood than the other placements, this can be a desirable property. A disadvantage of the RAID level 4 placement is that the parity disks will become a bottleneck for small writes [10] since every write must update a parity disk of which there is only one per row. Also, only $n-1$ disks per row, instead of n disks per row are available for servicing reads since the nth disks do not contain data.

3.3 Right-Asymmetric

The right-asymmetric placement is derived from the RAID level 0 placement by pushing out data stripe units horizontally as parity stripe units are inserted. For each successive parity stripe, the point at which the parity stripe unit is inserted is rotated one stripe unit towards the right.

3.4 Left-Asymmetric

The left-asymmetric placement is derived from the RAID level 0 placement by pushing data stripe unit out horizontally as parity stripe unit are inserted. For each successive parity stripe, the point at which the parity stripe unit is inserted is rotated one stripe unit towards the left.

3.5 Right-Symmetric

The right-symmetric placement is derived by right rotations of entire parity stripes from the RAID level 4 placement.

3.6 Left-Symmetric

The left-symmetric placement is derived by left rotations of entire parity stripes from the RAID level 4 placement.

3.7 Extended-Left-Symmetric

The extended-left-symmetric placement is derived from the RAID level 0 placement by pushing out data stripe units vertically as parity stripe units are inserted as

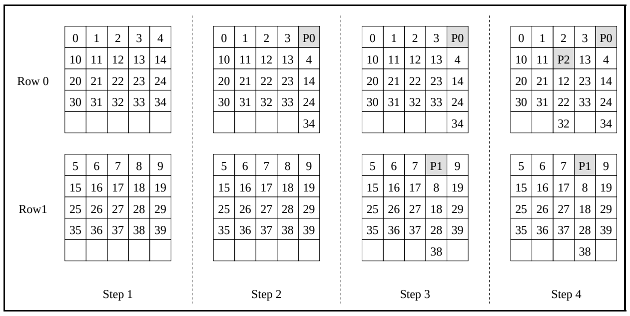

Figure 4: Derivation of the Extended-Left-Symmetric Placement

shown in Figure 4. For each successive parity stripe, the point at which the parity stripe unit is inserted is rotated one stripe unit towards the left. Figure 4 illustrates the conceptual steps in deriving the extended-left-symmetric placement from the corresponding RAID level 0 placement. In arrays with only one row of disks, the extended-left-symmetric placement is identical to the left-symmetric placement. In arrays with multiple rows of disks, traversing the logical blocks of the extended-left-symmetric placement in sequence results in a traversal of all the disks whereas the left-symmetric placement will skip every nth disk. Note that in Figure 3, where each five element column represents a distinct disk, $P1$ is the parity corresponding to stripe units 4, 5, 6 and 7 and *not* 5, 6, 7 and 9. Likewise, $P2$ corresponds to stripe units 8, 9, 10 and 11. This is to ensure that writes that are a parity stripe in size and parity stripe aligned can always be written without having to read additional information from the disks.

3.8 Flat-Left-Symmetric

The flat-left-symmetric placement is derived from the *extended*-left-symmetric placement by grouping all of the parity together and placing them at identical offsets within each disk. When reading large amounts of data, all disk heads within the same row skip over parity at the same time; thus, reducing disk synchronization time. When writing data, however, performance is likely to be worse than the extended-left-symmetric placement since the parity stripe unit is located at a different offset within the disk relative to its corresponding data stripe units. Note that in Figure 3, $P0$ is the parity corresponding to stripe units 0, 1, 2, and 3. Likewise, $P1$ corresponds to 4, 5, 6 and 7.

cylinders per disk	949
tracks per cylinder	14
sectors per track	48
bytes per sector	512
track skew	4
revolution time	13.9 ms
single cylinder seek time	2.0 ms
average seek time	12.5 ms
max stroke seek time	25.0 ms
max sustained transfer rate	1.7 MB/s

Table 1: Disk Parameters

4 Simulation of Parity Placements

This section evaluates the performance of the RAID level 0, RAID level 4, right-asymmetric, left-asymmetric, right-symmetric, left-symmetric, extended-left-symmetric and flat-left-symmetric parity placement schemes (all the placements illustrated in Figure 3) via simulation.

4.1 The RAID Simulator

The simulator consists of a module for computing the various parity placements, a model of disk behavior and a program to generate synthetic I/O requests.[3] The only hardware resources modeled are disks, which are rotationally synchronized. The request types used are random reads and random writes of various sizes. Sequential requests are not used because the sequentiality is much less meaningful when there is more than one process generating requests to the same disk (the disk will thrash between the two sequential request streams). It is also desirable in most systems to convert small sequential requests to large random requests. The load on the system is controlled by specifying the number of concurrent processes which issue requests. The number of processes, request type and request size are fixed for each simulation run.

4.2 The Disk

The disk model is loosely based upon the IBM 0661 3.25 inch SCSI disk drive. Table 1 tabulates the parameters of the simulated disk. Figure 5 plots the seek profile of the simulated disk.

[3] I am indebted to Garth Gibson and Peter Chen for supplying the disk model and program to generate synthetic I/O requests respectively.

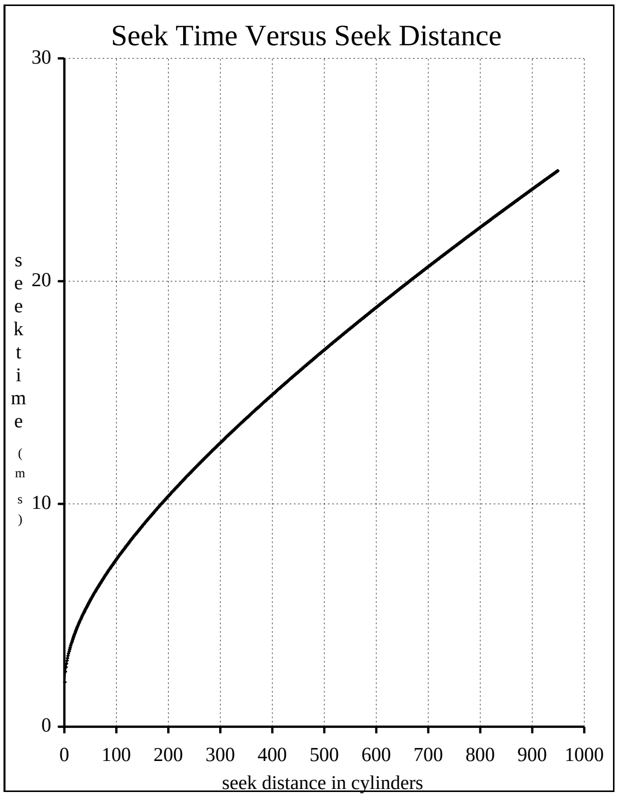

Figure 5: Seek Profile. A plot of the seek time in milliseconds versus the seek distance in cylinders for the simulated disk.

$$seekTime = \begin{cases} 0, & \text{if } x = 0 \\ a(x-1)^{0.5} + b(x-1) + c, & \text{if } x > 0 \end{cases}$$

Where x is the seek distance in cylinders and a, b and c are constants chosen to satisfy the single cylinder, max stroke and average seek times. For the simulated disk, $a = 0.4623, b = 0.0092$ and $c = 2$.

4.3 Simulation Parameters

The following is a list of the input variables used in the simulation and their corresponding ranges.

- Parity placement scheme (one of the eight previously described).
- Number of rows of disks in array (1 or 2).
- Number of disks per row (5 or 9).
- Size of the stripe unit (4KB, 8KB, 16KB, 32KB, or 64KB).
- Request type (read or write).
- The size of the request (2KB to 1MB).
- Request alignment (aligned or unaligned). Aligned requests are aligned on their size boundaries. Unaligned requests are aligned on sector boundaries (512 bytes).
- The degree of concurrency; that is, the number of processes generating requests (1, 2 or 16). When the concurrency equals one, the average response time is proportional to the inverse of throughput. The term *low load* refers to a degree of concurrency of one. The term *high load* refers to a degree of concurrency of sixteen.

The only output variable of interest is the throughput, measured in megabytes per second. Note, in the interests of reducing the parameter space, all workloads are homogeneous, mixtures of reads and writes as well as non-constant distributions of request sizes are not used. Another reason for using homogeneous workloads is that they are more likely to accentuate the differences in performance of the various parity placements. We leave heterogeneous workloads as future work.

The cross product of all input variables was considered too large to simulate and analyze properly. We also did not have reasonable ranges for the input variables when we started. Thus, only specific subsets of the cross product were investigated. The selection of the subsets and the investigation of the results proceeded in an iterative manner to reduce the possibility of missing interesting points in the cross product. In all, approximately 20,000 points in the cross product were investigated.

4.4 Simulation Results

Figures 6, 7, 8 and 9 illustrate the simulated performance of each parity placement under a set of particular inputs and is characteristic of other input combinations which are not illustrated here. Note that at low load, the graphs display a sawtooth pattern with a period approximately equal to 2 $rows \times 5$ $columns \times$ $32KB$ $stripeunits = 320KB$. The first dip occurs when requests become large enough to wrap around the array. At this point, some of the disks must read/write two stripe units while other disks read/write only one stripe unit. Some of the disks end up waiting for the other disks, resulting in inefficient resource utilization. This behavior is repeated each time the request wraps around the array, resulting in a periodic behavior. Note that at low load and relatively large requests sizes, the choice of parity placement results in up to a 20 to 30 percent difference in performance for the array configuration simulated. The following sections examines each of the Figures 6, 7, 8 and 9 in greater detail.

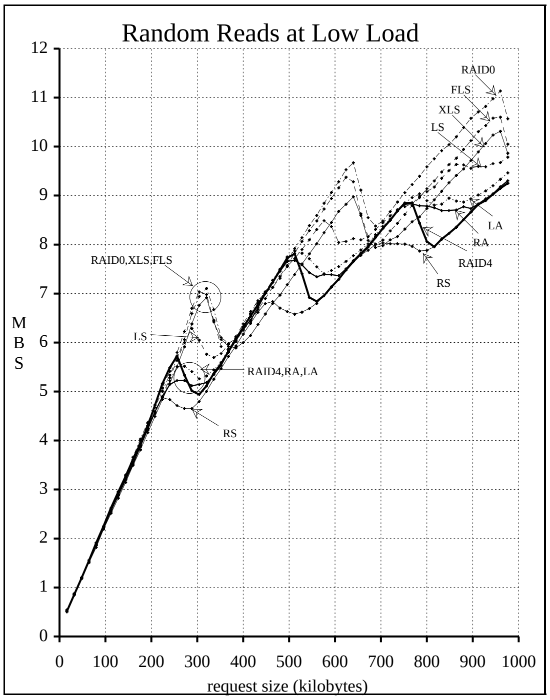

Figure 6: Reads at Low Load

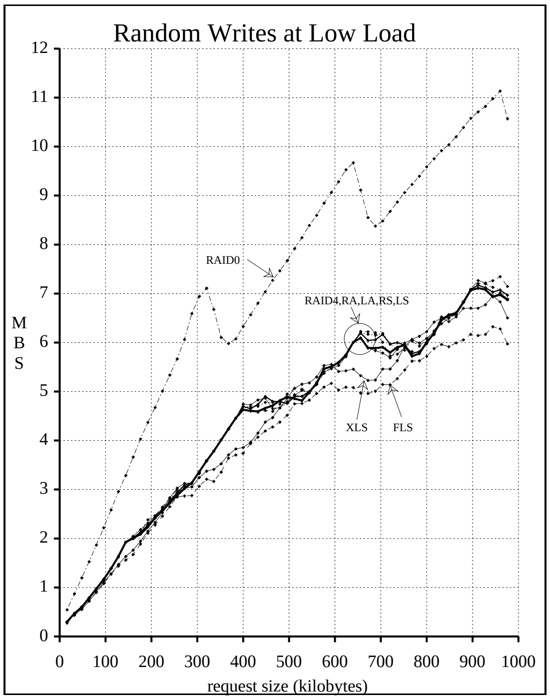

Figure 7: Writes at Low Load

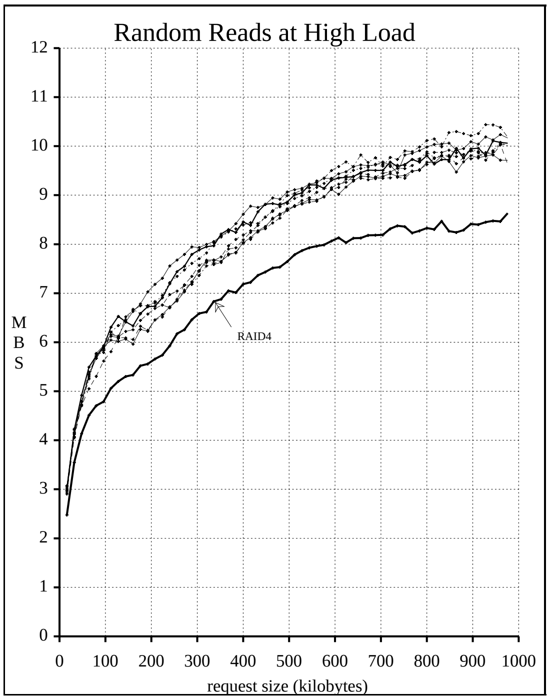

Figure 8: Reads at High Load

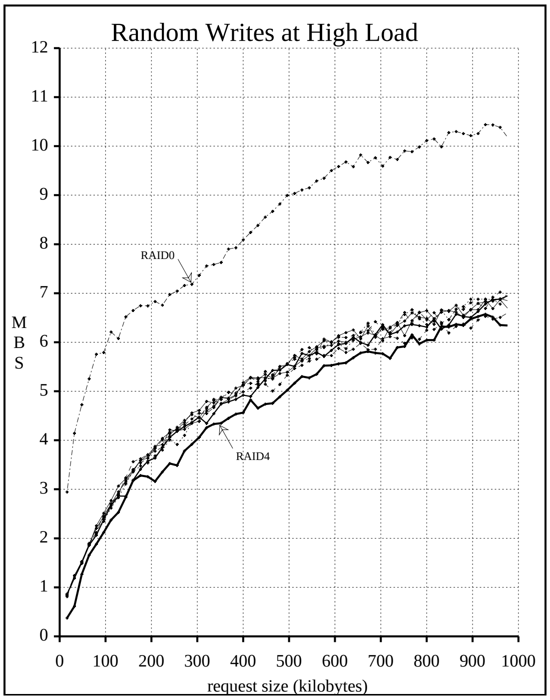

Figure 9: Writes at High Load

The simulation parameters are two rows, five disks per row, 32KB stripe units, and unaligned requests. The degree of concurrency for low load is one and the degree of concurrency for high load is sixteen.

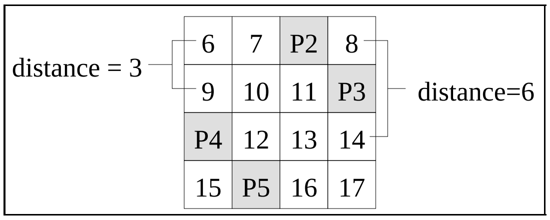

Figure 10: Placement Distance

4.4.1 Reads at Low Load

When performing reads at low load, roughly four groups of placements can be distinguished based on performance. From highest performance to lowest performance they are grouped as follows:

1. RAID level 0, extended-left-symmetric and flat-left-symmetric.

2. Left-symmetric.

3. RAID level 4, right-asymmetric and left-asymmetric.

4. Right-symmetric.

We define the *minimum placement distance* of a placement, which gives a good prediction of read performance at low load, as the minimum arithmetic difference between logical data stripe unit addresses of stripe units that are logically sequential on the same disk. Figure 10 illustrates the concept of distance. Parity stripe units are ignored when determining minimum placement distances. The minimum placement distance gives an indication of the number of disks a parity placement can stripe data over before reusing a disk and hence the amount of parallelism available for servicing a single request. The larger the minimum placement distance, the better the read performance at low load. We will refer to a placement with a minimum placement distance of N as a *distance N placement.* The maximum minimum placement distance for an array with N disks is N. Table 2 tabulates the minimum placement distances for the parity placement so far considered.

Among placements with the same distance, the RAID level 0 placement outperforms the extended-left-symmetric placement because the RAID level 0 placement never has to skip over parity stripe units. The flat-left-symmetric placement outperforms the extended-left-symmetric placement because the flat-left-symmetric placement aligns the parity stripe units together, so that on large reads, all parity stripe units can be skipped simultaneously. The left-asymmetric placement outperforms the RAID level 4 placement and

	minimum placement distance
RAID level 0	mn
Flat-Left-Symmetric	mn
X-Left-Symmetric	mn
Left-Symmetric	$m(n-1)+1$
Left-Asymmetric	$m(n-1)$
RAID level 4	$m(n-1)$
Right-Asymmetric	$m(n-1)-1$
Right-Symmetric	$m(n-1)-1$

Table 2: Parity Placement Distances. The symbol m denotes the number of rows in the array and the symbol n denotes the number of disks per row. Note that the left-symmetric and extended-left-symmetric placements are identical placements when the number of rows is one.

the right-asymmetric placement outperforms the right-symmetric placement because, even though their minimum placement distances are the same, the average placement distances are different. In each of the above two cases, the placement with the larger average placement distance performs better. The minimum placement distance is a safer but less discriminatory indicator of performance than the average placement distance in that it is possible to contrive placements with very good average placement distances but very poor performance.

4.4.2 Writes at Low Load

When performing writes at low load, roughly four groups of placements can be distinguished based on performance. From highest performance to lowest performance they are grouped as follows:

1. RAID level 0.

2. RAID level 4, right-asymmetric, left-asymmetric, right-symmetric and left-symmetric.

3. Extended-left-symmetric.

4. Flat-left-symmetric.

As expected, the RAID level 0 placement – no redundancy – displays the highest performance since it does not maintain parity. The extended-left-symmetric placement performs worse than the placements in the second group because it places the parity of the previous stripe on the same disk as the data of the current stripe. For example, note that in the extended-left-symmetric placement illustrated in Figure 3, $P0$ is on the same disk as stripe unit 4. Thus, if two sequential parity stripes are written, a single disk must service both a parity access and a data access. The extended-left-symmetric placement performs better than the flat-left-symmetric placement because it aligns parity with its corresponding data stripe units, whereas the flat-left-symmetric placement aligns the parity together. Note that this was the same reason the flat-left-symmetric-placement outperformed the extended left-symmetric-placement on reads at low load.

4.4.3 Reads at High Load

The RAID level 4 placement has the worst read performance at high load because it does not distribute parity and data over all disks. Thus, only $n-1$ rather than n disks are available for servicing read requests. The performance of the other placements, which distribute parity and data, are comparable. At high load, the most important criterion for high read performance is uniform disk utilization. Thus, placements which distribute the load uniformly are desirable.

4.4.4 Writes at High Load

As expected, the RAID level 0 placement displays the highest write performance at high load since it does not maintain parity. Note that because the redundant placements must write parity, their write performance asymptotically approachs $(n-1)/n$, $4/5$ in the case of Figure 9, of the maximum RAID level 0 performance at large request sizes. The RAID level 4 placement displays the worst write performance at high load due to contention for the parity disks. For small request sizes ($<$ 32KB) and these particular simulation parameters, the difference in performance between the RAID level 4 placement and the other placements is over a factor of two. With more disks per row and at higher load, we would expect the performance difference to be even larger. Thus, for high write performance at high load, placements which distribute the parity uniformly are desirable.

4.4.5 Performance Summary

In this section, we have examined the performance characteristics of eight parity placements. The RAID level 0 placement is unsuitable since it does not support parity. The left-symmetric placement outperforms the RAID level 4, right-asymmetric, left-asymmetric and right-symmetric placements on reads at low load over almost all request sizes and has comparable or better performance in all of the other cases. Thus, the choice to be made is between the left-symmetric, extended-left-symmetric and flat-left-symmetric placements. Which placement is used for a particular system depends on the importance of read performance at low load versus write performance at low load. For the reader's benefit, the performance of the above three placements at low load is redisplayed in Figure 11 with the other placements omitted. To make the differences easier to evaluate, Table 3 tabulates the absolute throughput and throughput relative to the left-symmetric placement for a few key requests sizes ($2\ rows \times 5\ columns \times 32KB\ stripeunits = 320KB$) from Figure 11.

Note that the performance of the left-symmetric, extended-left-symmetric and flat-left-symmetric placements converge as the request size increases for reads but remains relatively unchanged for writes. Note also that the flat-left-symmetric placement achieves only 90% of the performance of the left-symmetric placement for 16KB request sizes.

Due to space constraints, we have only presented the simulation results for a particular array configuration; however, the preceding is characteristic of other array configurations that were simulated and the following is applicable to RAID level 5 systems in general. We propose the following list of desirable placement properties, roughly in the order of their importance:

1. Stripe units belonging to the same parity stripe should not occupy the same column. (In many RAID systems, the disks within a column have a common failure mode, the string interface for example.) This is referred to as the "orthogonal RAID" property. All of the proposed placements have this property.

2. In a RAID with n disks per row, the ith parity stripe unit should correspond to stripe unit j such that $j \operatorname{div} n = i$. This guarantees that any write request that is stripe aligned and a stripe in size can be written without reading old data. All of the proposed placements have this property.

3. Parity and data should be distributed over all disks. All of the proposed placements except the RAID level 4 placement have this property.

4. The minimum placement distance should be maximized. Only the extended-left-symmetric and flat-left-symmetric placements are maximum distance placements. The left-symmetric placement is a maximum distance placement for a single row of disks but not for multiple rows of disks.

5. If the number of rows is n then stripe units belonging to any n consecutive parity stripes should not occupy the same disk. This avoids the write contention experienced by the extended-left-symmetric and flat-left-symmetric placements which are the only proposed placements which do not satisfy this property.

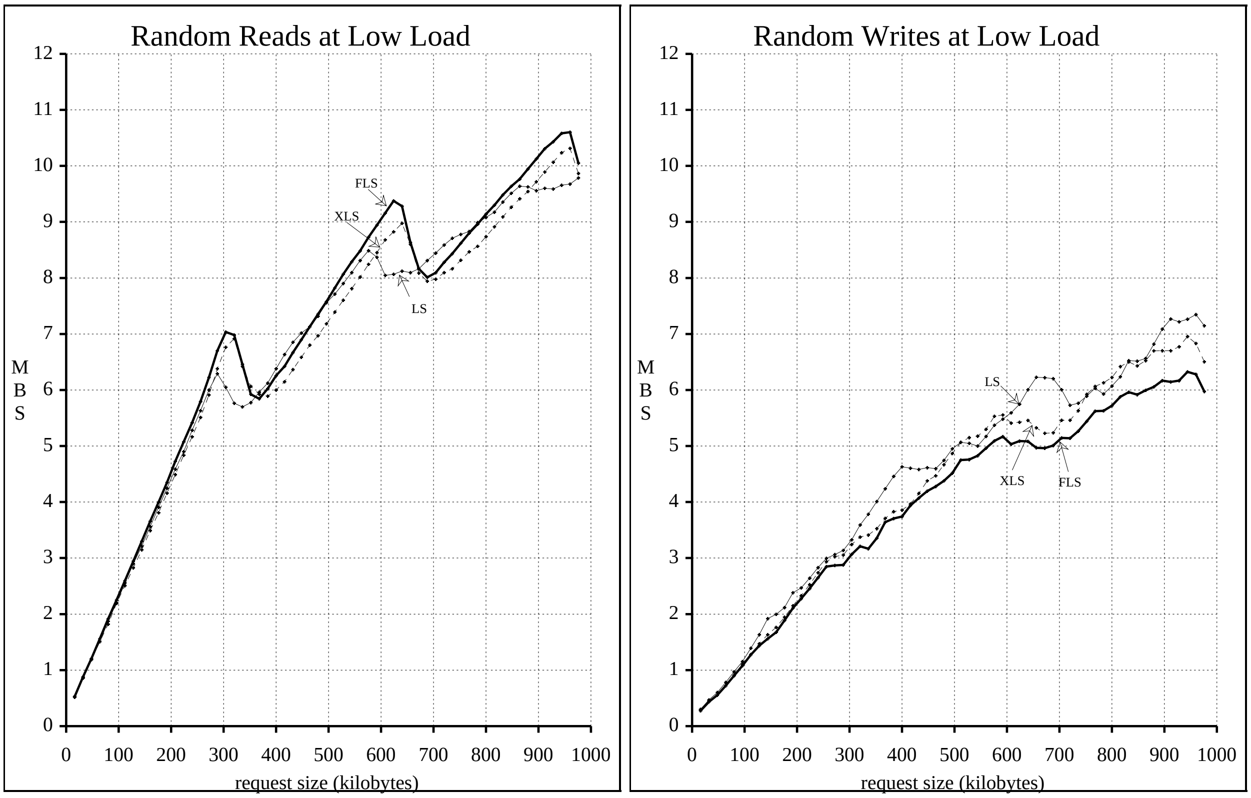

Figure 11: Parity Placement Performance Summary. The simulation parameters are two rows, five disks per row, 32KB stripe units, unaligned requests, and the degree of concurrency for low load is one.

Random Reads at Low Load

	16KB		320KB		640KB		960K	
	MBS	% of LS	MBS	% of LS	MBS	% of LS	MBS	% of LS
Left-Symmetric	0.52	100%	5.77	100%	8.12	100%	9.68	100%
X-Left-Symmetric	0.53	102%	6.92	120%	8.98	111%	10.31	107%
Flat-Left-Symmetric	0.52	100%	6.98	121%	9.28	114%	10.60	110%

Random Writes at Low Load

	16KB		320KB		640KB		960K	
	MBS	% of LS	MBS	% of LS	MBS	% of LS	MBS	% of LS
Left-Symmetric	0.30	100%	3.59	100%	6.01	100%	7.35	100%
X-Left-Symmetric	0.29	97%	3.38	94%	5.46	91%	6.83	93%
Flat-Left-Symmetric	0.27	90%	3.21	89%	5.08	85%	6.28	85%

Table 3: Parity Placement Performance Summary. *MBS* denotes throughput in megabytes per second. *Percentage of LS* denotes throughput relative to the left-symmetric parity placement.

Unfortunately, properties 4 and 5 are at odds with one another. Given an array with N disks, it is not generally possible to devise a placement which is both distance N and satisfies Property 5 for a general m-by-n disk array. With only a single row of disks, the left-symmetric, extended-left-symmetric and flat-left-symmetric placements are optimal with respect to both properties. When selecting a parity placement for a general disk array, one must sacrifice either read performance or write performance.

5 Summary

We have defined and investigated the performance of a variety of parity placement schemes. We have shown that at relatively large request sizes of hundreds of kilobytes, the choice of parity placement significant affects performance (20-30 percent for the typical disk array configurations that are common today). We have shown that the left-symmetric, extended-left-symmetric and flat-left-symmetric are the best RAID level 5 parity placements, and that of these, the placement with the highest read performance, flat-left-symmetric, has the lowest write performance and the placement with the lowest read performance, left-symmetric, has the highest write performance. Finally, we have proposed properties that are desirable of parity placements. One thing that would be interesting to investigate and which we leave as future work is how the various placements would perform on heterogeneous workloads. In particular, how do the three best placements mentioned above perform on workloads that are a mixture of reads and writes?

6 Acknowledgements

We would like to thank members of the RAID group, especially Peter Chen, Ann Chervenak, Garth Gibson and David Patterson, and members of the Sprite group, especially Mendel Rosenblum, Mary Baker and John Ousterhout for the support they have provided in researching and writing this paper. We would also like to thank our government and industrial affiliates, Array Technologies, DARPA/NASA (NAG2-591), DEC, Eastman Kodak, Hewlett-Packard, IBM, Intel Scientific Computers, California MICRO, NSF (MIP 8715235), Seagate, Sun Microsystems and Thinking Machines Corporation for making this work possible.

References

[1] Peter M. Chen. An evaluation of redundant arrays of disks using an amdahl 5890. Technical Report UCB/CSD 89/506, University of California at Berkeley, May 1989.

[2] Peter C. Dibble. A parallel interleaved file system. Technical Report CS TR 334, University of Rochester, March 1990.

[3] R. H. Katz, G. A. Gibson, and D. A. Patterson. Disk system architectures for high performance computing. In *Proc. IEEE*, December 1989.

[4] M. Y. Kim. Synchronized disk interleaving. *IEEE Trans. on Computers*, November 1986.

[5] M. Y. Kim and A. N. Tantawi. Asynchronous disk interleaving. Technical Report RC12497, IBM, January 1987.

[6] Edward K. Lee. Software and performance issues in the implementation of a raid prototype. Technical Report UCB/CSD 90/573, University of California at Berkeley, May 1990.

[7] M. Livny, S. Khoshafian, and H. Boral. Multi-disk management algorithms. In *Proc. SIGMETRICS*, May 1987.

[8] Ethan L. Miller. Input/output behavior of supercomputing applications. Technical report, University of California at Berkeley, 1991. In preparation.

[9] David A. Patterson, Peter M. Chen, Garth Gibson, and Randy H. Katz. Introduction to redundant arrays of inexpensive disks (raid). In *Proc. IEEE COMPCON*, Spring 1989.

[10] David A. Patterson, Garth Gibson, and Randy H. Katz. A case for redundant arrays of inexpensive disks (raid). In *Proc. ACM SIGMOD*, June 1988.

[11] K. Salem and H. Garcia-Molina. Disk striping. In *Proc. IEEE Data Engineering*, February 1986.

[12] Martin Schulze, Garth Gibson, Randy H. Katz, and David A. Patterson. How reliable is a raid? In *Proc. IEEE COMPCON*, Spring 1989.