Introduction to Redundant Arrays of Inexpensive Disks (RAID)

David A. Patterson, Peter Chen, Garth Gibson, and Randy H. Katz

Computer Science Division
Department of Electrical Engineering and Computer Sciences
571 Evans Hall
University of California
Berkeley, CA 94720
(pattrsn@ginger.berkeley.edu)

1. The Pending I/O Crisis

The computer industry has entered a period of unprecedented improvement in CPU performance. Midrange uniprocessors are improving at a rate of 50% to 100% per year, with this uniprocessor rate multiplied by the increasing popularity of multiprocessors.

There is more to a computer system, however, than the processor. Memory and I/O must match these gains to deliver a system that achieves the potential of the processor. Various rules of thumb tie CPU performance to both the capacity and the speed of the memory and I/O subsystems, so we need advances in both dimensions to create faster yet balanced computers.

Memory subsystems are matching the challenge of CPU performance. DRAMs are growing in capacity by about the same rate as CPUs are improving in performance. The performance of DRAMs is improving at a much more modest rate, perhaps doubling ever decade. SRAMs, however, are matching the performance improvement of CPUs. Fortunately there is a long list of architectural innovations--duplicated caches, cache coherency, multilevel caches, prefetching, interleaved memory, pipelined memory, and so on--allowing fast SRAMs and large DRAMs to be combined into memory systems that can match the performance demands of new processors.

I/O systems performance is limited by networks and magnetic disks. There are several efforts to improve networks speeds by factors of 10 to 100, so we see little trouble here provided some of these efforts succeed. The good news about magnetic disks is that improvements in capacity and cost per megabyte are keeping pace with processors. The bad news is that performance gains are modest. Rotation speed is unchanged in more than a decade, and in that same time period seek time has improved by no more than a factor of two.

Without innovation, we see most programs becoming I/O bound. If such an I/O crisis comes to pass, there will be little reason to buy faster processors, since it is economic nonsense to pay more to increase processor idle time.

2. Arrays of Inexpensive Disks

While the magnetic disk industry has made little progress in improving speed of disks, it has significantly reduced the size of disks. The personal computer industry has created a market for 5.25 and 3.5 inch drives, reducing the cost per disk system as well as the traditional lowering of cost per megabyte. Table I below compares the top-of-the-line IBM 3380 model AK4 mainframe disk, Fujitsu M2361A "Super Eagle" minicomputer disk, Impress/CDC Wren-IV workstation disk, and the Conner Peripherals CP 3100 personal computer disk.

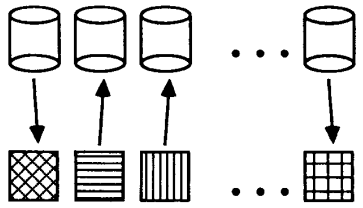
Characteristics	*IBM 3380*	*Fujitsu M2361A*	*CDC Wren-IV*	*Conners CP3100*
Disk diameter (inches)	14	10.5	5.25	3.5
Formatted Data Capacity (MB)	7,500	600	340	100
MTBF rated by manufacturer (hours)	30,000	20,000	40,000	30,000
No. Actuators	4	1	1	1
Maximum I/O's/second/Actuator	50	40	40	30
Maximum I/O's/second/box	200	40	40	30
Transfer Rate (MB/sec)	3	2.5	1.5	1
Power/box (W)	1,650	640	40	10
MB/W	4.5	0.9	8.5	10.0
Volume (cu. ft.)	24	3.4	0.3	.1
MB/cu. ft.	312	176	1133	1000

Table I. *Comparison of IBM 3380 disk model AK4 for mainframe computers, the Fujitsu M2361A "Super Eagle" disk for minicomputers, Impress/CDC Wren-IV disk for workstations, and the Conners Peripherals CP 3100 disk for personal computers. By "Maximum I/O's/second" we mean the maximum number of average seeks and average rotates for a single sector access.*

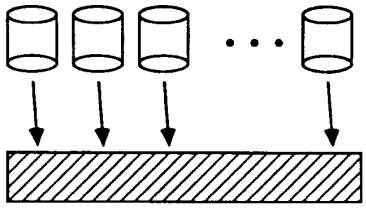
The table shows that the small drives are close to the performance and the reliability of the large drives. The biggest difference is the lower power and smaller volume per megabyte, with the smallest drive about three to five times better. What is not listed is cost per megabyte. The trade-off is the potential lower cost per megabyte of large drives--since they can amortize the cost of the support electronics over a larger number of megabytes--versus the economies of scale provided by the higher sales volume of the smaller drives. It is also an issue of manufacturer's cost versus selling price, with the likelihood that the mainframe drives having larger markup. In our discussions with companies that make several sizes of disks they say the cost per megabyte is a function of the popularity of a given disk at a given time; they suggest assuming the cost per megabyte is independent of disk diameter. We make that assumption.

Given similar performance and cost per megabyte, one way to address higher performance is replace a single large drive by an array of many smaller drives.[Kurzweil 88] Such an array provides many more arms per dollar, meaning higher performance because many small request can be serviced independently *and* large requests can be spread over several disks to transfer in parallel (Figure 1). Moreover, the advantages in volume and power can mean a smaller footprint and lower air conditioning requirements.

CH2686-4/89/0000/0112$01.00 © 1989 IEEE



(a) Arrays of inexpensive disks support several small, independent reads and writes simultaneously.



(b) Arrays of inexpensive disks also support large reads or writes, where all disks trasnsfer in parallel.

Figure 1. *Replacing a single large expensive disk by an array of inexpensive disks improves performance because it can support (a) many small individual accesses simultaneously and (b) large accesses with all disks transferring in parallel ("striping"* [Kim 86] [Salem 86]).

3. Redundant Arrays of Inexpensive Disks

The drawback to replacing a single large disk with, say, 100 small disks is reliability. Basically 100 devices have 1/100th the reliability of a single device, reducing the mean time between failure (MTBF) from over three years to less than two weeks. This is so poor that without a scheme to improve reliability†, arrays containing many disks are unfeasible.

Fortunately redundancy can improve reliability of 100 small disks beyond that of a single large disk. Although failures occur 100 times more frequently with 100 disks, the chances of a second failure before the first is replaced is small enough to tolerate more failures and still be more reliable than a single disk. Thus Redundant Arrays of Inexpensive Disks, or RAID, has the potential advantage of not only higher performance with lower power and smaller footprint, but also higher reliability.

In an earlier paper we presented five different schemes for disk redundancy [Patterson 88], but here we only present the two schemes most likely to be implemented. If we include the rest of the computer system it would seem we would need to duplicate all components to achieve high reliability. A companion paper shows high reliability is possible to achieve with more modest redundancy costs[Schulze 89]. In this paper we assume that the reliability is sufficient. Readers interested in redundancy schemes for much larger disk arrays, file systems or databases for RAID should see [Gibson 89], [Douglis 89], or [Stonebraker 88].

4. Mirrored RAID

The simplest redundancy scheme is to double the number of disks, keeping a redundant copy of each datum. If a disk fails, the system uses the redundant copy until the failed disk is replaced, and then copies a redundant version to the new disk. Data is lost only if the other disk of the pair fails before first is replaced. In normal operation a copy is maintained by making every write update both disks. This scheme is variously called *mirroring*, *shadowing*, or *copying*, and is used by Tandem, DEC, and IBM to improve reliability. (In our original paper we called mirroring a level 1 RAID.)

This scheme has the highest cost: the user must double number of disks for the same amount of data or, conversely, use only half the real storage capacity of the disks. If the arms and spindles of a pair were synchronized then the performance of mirroring versus nonredundant disks would be the same. This is not commonly how the mirroring is implemented, and a write results in independent writes to two disks. They can be overlapped, but in general one will have longer seek and/or rotational delay. On the other hand, the independence of disks can improve performance of reads. The system might look at the pair of disks that have the data; if only one is busy, it chooses the other. If both are idle, it picks the disk that has shortest seek [Bitton 88].

In summary, mirrored RAIDs have the highest cost for a given storage capacity, but performance versus a nonredundant disk array depends on the mix of reads and writes.

5. N+1 RAID

An advantage of disks is that they can detect their own mistakes: either the disk controller will not get a response or the ECC code per sector will be incorrect. By calculating and storing parity of a group of disks on a bit-per-disk basis, any single disk failure can be corrected simply by reading the rest of the disks in the group to determine what bit value on the failed disk would give the proper parity. This N+1 RAID can lose data only if there is a second failure in the group before the failed drive is replaced.

This scheme has much lower cost overhead, with the customer deciding how much overhead he wants to pay by increasing the number of disks in the parity group. Performance depends not only on the mix of reads and writes, but also on the size of the accesses. Since there is ECC information on each sector, read performance is essentially the same as nonredundant disk arrays. For "large" writes--writing at least a sector to every disk in the parity group--the only performance hit is 1/N more writes to write the parity information. Writes to data on a single disk, on the other hand, require four disk accesses:

1) Read the old data;
2) Read the old parity;
3) Write the new data;
4) Write the new parity using this formula:
   new parity = (old data xor new data) xor old parity

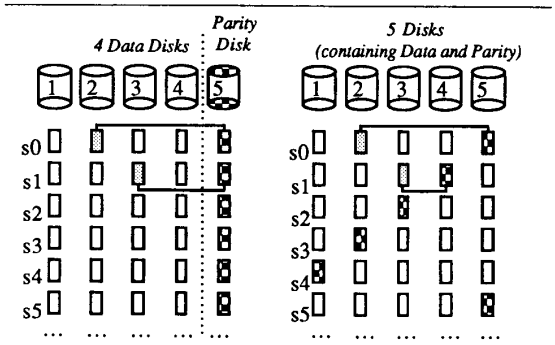
It would seem that an additional performance limit would be the parity disk, since small writes to any disk must also cause a read and a write to the parity information. Such a bottleneck is avoided by spreading the parity over several disks. Figure 2 shows how the straight-forward implementation is altered to avoid parity bottlenecks. (In our original paper we called N+1 a level 5 RAID.)

6. Performance of Mirrored RAID vs. N+1 RAID

Comparing these two RAID organizations is both simple and difficult. Common sense suggests Mirrored RAID, using roughly twice as many disks, is more expensive and has higher performance. If cost is your only concern, you pick N+1, and you pick Mirroring if performance is the only concern. What if you care about both cost and performance? We use the metric of throughput per disk, since a customer can always buy more disks to solve an I/O bottleneck in any scheme. From the discussion above it is clear that comparative performance is then sensitive to whether the accesses are reads or writes and the size of the accesses.

By making several simplifying assumptions we can estimate comparative performance in all these measures. These simplifying assumptions include:

†In this paper we use the term reliability to include availability.



(a) Parity information in straight-forward scheme. The sectors are shown below the disks. (The checked areas indicate the parity information.) Writes to s0 of disk 2 and s1 of disk 3 imply writes to s0 and s1 of disk 5. The parity disk (5) becomes the write bottleneck.

(b) Parity information for rotated scheme. The sectors are shown below the disks, with the parity information and data spread evenly through all the disks. Writes to s0 of disk 2 and s1 of disk 3 still imply 2 writes, but they can be split across 2 disks: to s0 of disk 5 and to s1 of disk 4.

Figure 2. *The performance impact of this small change is large in large parity groups since it allows N+1 RAID to support multiple individual writes per group. For example, suppose we want to write sector 0 of disk 2 and sector 1 of disk 3. As shown on the left writes must be sequential since both sector 0 and sector 1 of disk 5 must be written. However, as shown on the right, writes can proceed in parallel since a write to sector 0 of disk 2 still involves a write to disk 5 but a write to sector 1 of disk 3 involves a write to disk 4.*

- Every access is assumed to take one average seek and one average rotation;
- Every access is the same size;
- Accesses are spread optimally across all disks;
- Disks are never idle waiting for requests;
- Accesses are assumed to be homogeneous, e.g., 100% small reads.
- There is no optimization to schedule reads on mirrored disks;
- Latency is ignored.

Figure 3 is a comparison of the two schemes using these assumptions, with 100% being the performance of a nonredundant disk array for that type of access. Using these assumptions we see that read performance is identical, with N+1 winning on capacity and large writes while Mirroring wins on small writes.

To see if these results would hold in more realistic circumstances, we recently completed an experiment on an Amdahl 5890 using many large Amdahl 6380 devices[Chen 89]. N+1 RAID used 11 6380 devices with a track of 4 KB as the primary block parity block size. This experiment improved the above calculation in the following ways:

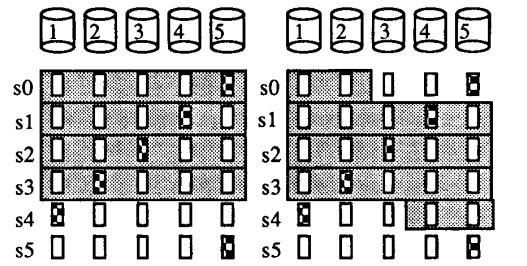
- Real hardware was used, accounting for CPU time and xor calculation time;
- Seek and rotation times were not constant;
- The size of large accesses is not exactly one block per disk in a parity group and a large access is not aligned to fit in the minimum number of parity groups (see Figure 5 below);
- Accesses varied in size around an average. The average size was 6.5 KB for small accesses and 1.5 MBfor large accesses. Several distributions of access sizes were used.
- Accesses were not spread evenly between all disks, so some were hot spots and some were underutilized;
- Latency is considered, with the the load varied until 90% of accesses met a latency threshold;
- For mirrored disks the reads were optimized to minimize seeks, thereby slowing mirrored writes.

Figure 4 compares the measured results versus the estimated results in Figure 3.

While most of the measured results were close to the estimates, a few were not:

- Large reads for N+1 RAID did not achieve 100% because it of the parity information. While parity need not be read, it still takes time for sector containing the parity to spin under the head or for the head to move over the track containing parity during a seek. For the large writes in this experiment about 3 of the 44 tracks would contain parity, and 41/44 ratio of data to total space is the same 93% of total time that we measured.
- The optimization choosing which mirrored disk was best to minimize seek distance of small reads over a nonredundant disk by 14%. This optimization had little effect on large transfers.
- N+1 RAID large writes were 70% vs. 91% of nonredundant disks because the accesses were not "aligned" to exactly one track per disk. Figure 5 shows the model for the estimate and the more realistic model for the measurements. An average large write in this experiment would write 75% of the tracks in a full parity group write with the remaining 25% split across two partial parity group writes.
- Small writes were slightly faster for N+1 RAID and significantly slower than expected for Mirrored RAID. Small writes do not need to seek to write the new values after reading the old in N+1, they just pay a full rotation waiting for the old data to spin under the head again. Mirrored RAID small writes were slower than expected, in part becuase of the higher latency since writes go to twice as many disks.

Using the assumptions and the results from this experiment we see a closing of the performance gap on writes--Mirroring is closer to N+1 on large writes and N+1 is closer to Mirroring on small writes--while Mirroring gains a slight edge on reads, with capacity still on the side on N+1.



(a) In estimate we assumed that a large writes were multiplies of the size of the parity group, and that they were aligned so that there were no partial groups.

(b) The writes to s0 of disks 1 and 2 will require reads in the s0 parity group to calcualte the parity in s0 of disk 5, and similarily extra reads will be needed for parity group s4.

Figure 5. *The estimates used the model on the left (a), while the experiment randomly placed the large files so there would be writes to portions of two parity groups in addition to the writes to full parity groups. Clearly the importance of alignment is dependent on the size of a large write.*

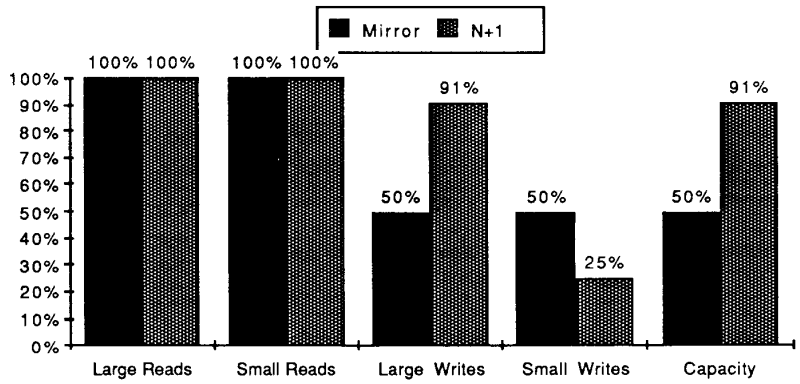


Figure 3. *Estimated performance of Mirrored RAID vs. N+1 RAID for Large Reads, Small Reads, Large Writes, Small Writes, and Useful Storage Capacity. Small accesses mean one block while large accesses mean one block per disk in the parity group. The group size for N+1 is 11 disks. Measures are percentage of nonredundant disk throughput.*

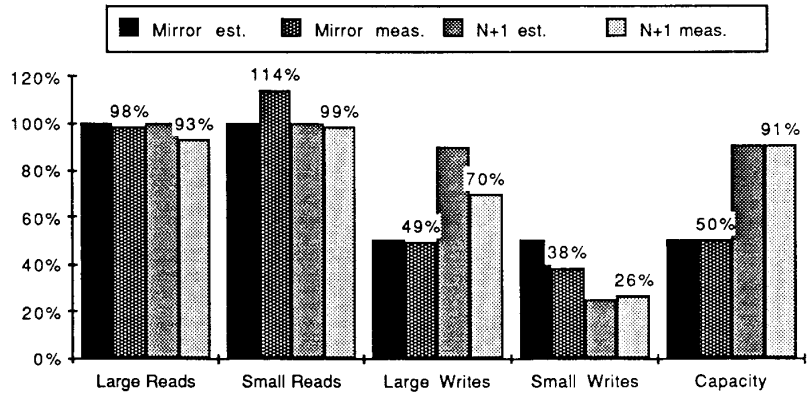


Figure 4. *Estimated and measured performance of Mirrored RAID vs. N+1 RAID for Large Reads, Small Reads, Large Writes, Small Writes, and Useful Storage Capacity. The experiment was run on an Amdahl 5890 CPU using Amdahl 6380 devices. For N+1 the group size was 11. The average size of a small access is 6 KB and the average size of a large access is 1.5 MB. Measures are percentage of nonredundant disk throughput.*

7. RAID-I Prototype

The RAID hardware research is not being done in isolation. The XPRS project--eXperimental Postgres, Raid, and Sprite--includes the development of the Sprite operating system and the Postgres database to take advantage of higher performance I/O systems. XPRS is in turn part of Mammoth project of U.C. Berkeley Computer Science Division that is exploring the advantages of massive storage across many fields of computer science.

To provide a vehicle for architectural experiments and software development for the XPRS and Mammoth projects, we are constructing a prototype we call RAID-I. It consists of:

Sun-4/280 with 128 MB of memory;
7 32-bit VME-SCSI Host/Bus Adapters;
32 CDC Wren-IV 340 MB 5.25" disks (using embedded SCSI controllers);
1 Ethernet interface.

Figure 6 shows drawing of RAID-I.

RAID-I is an off-the-shelf system, to be used for software development and for measurements to determine the design of later RAID systems. Hence RAID-I has no performance goals, with the only goal being stability for software development. We expect that the VME bus, operating system overhead, and SCSI controller overhead will conspire to reduce the available bandwidth to much below that of 32 disks. Our goal is for RAID-I to have enough capacity to make it an attractive resource in our department so that we can get hands on experience with RAIDs with real users. We expect RAID-I to be running Sprite and Postgres in Spring 1989.

8. Discussion

We have discussed the RAID ideas with several groups of people, and some questions come up so commonly that we address them here.

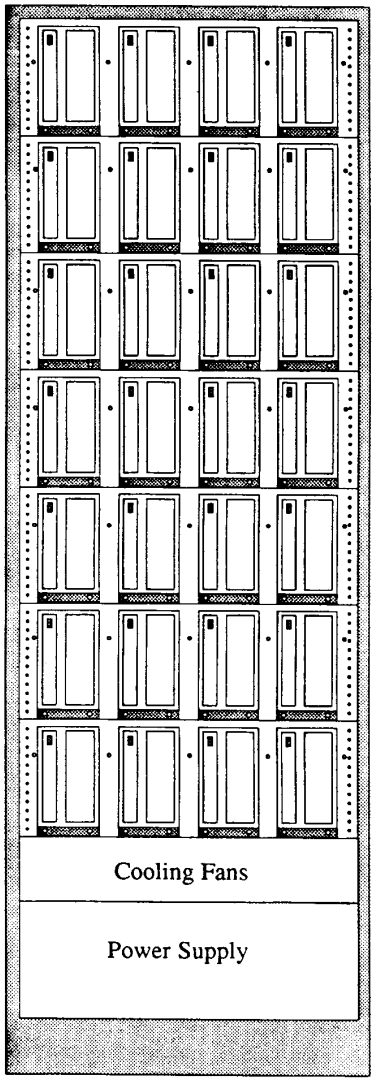


Figure 6. *RAID-I mechanical configuration for magnetic disks. (The CPU is in a separate cabinet.) Note that disks can be placed in the back of the cabinet as well as the front, so the maximum capacity for this organization is 56 5.25" drives per rack.*

1. How realistic are the assumptions of the independence of disk failures and constant failure rates? Is it a triumph of hope over experience?

We wish we knew. There are no published papers showing the real lifetime failure rates of disks. Disk manufacturers estimate MTBF by accelerated life-cycle testing, assuming that failures are independent and the exponential failure model. If the reader knows of a source of such information, please contact the authors. It matters not who the manufacturer is, we just need empirical data on disks to evaluate the viability of alternatives for redundancy.

One problem that several magnetic disk manufacturers have mentioned is what we would call the "Pinto Effect;" a mistake is made in manufacturing process that is so disastrous that the disk manufacturer will recall all affected disks and replace them. The common theme is that the mistake is uncovered after the disks have been in the field for several months and the disks all fail within a short time of one another. One example was a manufacturer who glued together the two halves of an head-disk assembly, with this glue dissolving after the disks had been in the field for 18 months. Another example was that a new bateriacide used in an air filter interacted with the disk surface so that many failures occurred six months later. A common cause of the Pinto Effect is that a supplier will change some component as a cost cutting measure without notifying the disk manufacturer, and disastrous consequences occur due to unforeseen interactions.

Although we desperately need real data on disk failures, we are performing studies of using models to estimate the impact of the Pinto Effect on RAID reliability.

2. What are the implications of having or not having a hot standby spare?

A standby spare is an unused but electrically connected disk that can replace a failed disk in the system without human intervention. The major advantage of standby spare is reducing the mean time to repair (MTTR), with the disadvantage of increasing the cost and complexity of the system. As soon a disk fails, the system can immediately reconstruct the information onto the spare. Depending on the load on the other disks in the system and the capacity of the failed disk, the MTTR could be 10 to 60 minutes with a standby spare.

Without a standby spare, the MTTR will be significantly longer:

- The repair man must be contacted to bring a disk for replacement; this could be anywhere from 1 hour to 12 hours depending on the level of service.
- Disks have a limit as to how fast they can adjust to temperature changes; a typical specification is 5 degrees per hour. If the field engineer stores replacement disks in his car, then he may have to wait for the disk to acclimatize before installing it. This could be 0 to 4 hours. If disks are stored in the computer room to overcome acclimatization delays, this removes the cost advantage of not having standby spares.
- Even if the system is designed to allow hot spare insertion, in practice people responsible for a computer will not want someone to open cabinets and replace equipment on a working computer while many important jobs are running on the system. (A failed disk does not disable the system since the data can be reconstructed.) Thus sociological implications of manual replacement may extend MTTR to 24 to 72 hours.

3. If RAID turns out not to be the answer, what is?

The slow seek and rotation delays of mechanical devices can easily be overcome with solid state memory. The recently introduced "Solid state disks" (SSD) provide more than ample speed to match the growth of CPUs, because mechanical devices are essentially removed from the hierarchy. The problem is that cost is essentially a factor of 10 to 20 times larger for solid state disks. In applications where a few hot spots dominate disk accesses, a more economic solution would be to automatically migrate data between SSD and magnetic disks to achieve higher performance at lower cost.

The cost of SSD over RAM is a battery. The advantage of a separate box with solid state memory over just larger main memory is that the SSD is as reliable as magnetic disks in the presence of bugs in the operating system and database software, while this is not the case for main memory. SSD also have the advantage that they are more easily multiported than main memory, allowing other CPUs to access them in case of a CPU failure.

There is also a way to improve reliability without redundant data storage. Recent work has suggested that disk failures can be predicted. For decades field engineers have had diagnostics programs that they run to exercise a disk during a preventative maintenance cycle. The purpose is to decide whether or not the disk should be replaced even though it hasn't failed yet. This is clearly a failure prediction scheme. Lin and Siewiorek studied messages printed on the system console and were able to see indications of disk failures up to two weeks before the failure [Lin 1986]. DEC has a software product, called VAXSimPLUS, that takes advantage of the predictive nature of some failures. It purports to predict 90% disk failures far enough in advance to copy the data from the suspect disk onto a spare disk preventing any disruption of the data.

To achieve improvements in reliability similar to RAIDs would take much higher accuracy of prediction than 90%. It is also clear that disks and interfaces that could provide an early warning system would be ideal parts for even less expensive RAIDs.

4. *Can a disk manufacturer successfully market a RAID?*

Disk manufacturers must live with the interfaces provided by computer systems; for example, SCSI interfaces, HBA interfaces, and operating systems. No matter how large a file is, UNIX will ask for it as a series of small (8 KB) blocks of data. Systems houses can that change the interfaces and the operating system have a much better opportunity to take advantage of the potential RAID.

5. What are implications of RAID for standard interfaces like SCSI and IPI?

RAIDs will be constructed with hundreds of disks, increasing the physical distance between the computer and the furthest disk and the number of disks that must attached to a common bus. Ideally future standards would allow longer distances for the connections, much higher transfer rates, and more devices per connection.

9. Conclusion

RAIDs offer a cost effective option to meet the challenge of exponential growth in the processor and memory speeds. We believe the size reduction of personal computer disks is the key to the success of disk arrays, just as Gordon Bell argues that the size reduction of microprocessors is a key to the success in multiprocessors [Bell 85]. In both cases the smaller size simplifies the interconnection of the many components as well as packaging and cabling. While large arrays of mainframe processors are possible, it is certainly easier to construct an array from the same number of microprocessors (or PC drives). Just as Bell coined the term "multi" to distinguish a multiprocessor made from microprocessors, we use the term "RAID" to identify a redundant disk array made from personal computer disks.

With advantages in cost-performance, reliability, power consumption, and floor space, we expect RAIDs to replace large drives in future I/O systems. There are, however, several open issues that may bare on the practicality of RAIDs:

- *What will be the real lifetime of a RAID vs. MTBF calculated using the independent, exponential failure model?*
- *Will disk controller design limit RAID performance?*
- *How should 100 to 1000 disks be constructed and physically connected to the processor?*

Acknowledgements

This work was supported by the National Science Foundation under grant # MIP-8715235 and the California MICRO program. We would like to thank the support of Sun Microsystems Incoporated, Impress/CDC, and our other industrial partners for the additional support of this research. Peter Chen was supported in part by an ONR fellowship and Garth Gibson was supported in part by both an IBM fellowship and a Computer Measurement Group award.

References

[Adaptec 87] AIC-6250, *IC Product Guide*, Adaptec, stock # DB0003-00 rev. B, 1987, p. 46.

[Bell 85] Bell, C.G., "Multis: a new class of multiprocessor computers," *Science*, vol. 228 (April 26, 1985) 462-467.

[Bitton 88] D. Bitton and J. Gray, "Disk Shadowing," *in press*, 1988.

[Chen 89] P. Chen, "An Evaluation of Redundant Arrays of Disks using an Amdahl 5890," M.S. Report, 1989 (in preparation).

[Douglis 89] F. Douglis and J. Ousterhout, "A log structured file system," Spring COMPCON 89, March 1, 1989, San Francisco, CA, *(in this proceedings).*

[Fujitsu 87] "M2361A Mini-Disk Drive Engineering Specifications," (revised) Feb., 1987, B03P-4825-0001A.

[Gibson 89] G. Gibson, L. Hellerstein, R. Karp, R. Katz, and D. Patterson, "Error Correction in Large Disk Arrays," ASPLOS III, April 1989, Boston, MA.

[Kim 86] M.Y. Kim, "Synchronized disk interleaving," *IEEE Trans. on Computers*, vol. C-35, no. 11, Nov. 1986.

[Kurzweil 88] F. Kurzweil, "Small Disk Arrays - The Emerging Approach to High Performance," presentation at Spring COMPCON 88, March 1, 1988, San Francisco, CA.

[Lin 86] T-T.Y. Lin and D.P. Siewiorek, "Architectural Issues for On Line Diagnostics in a Distributed Environment," *International Conference on Computer Design*, IEEE Computer Society, Rye Town, NY, October 1986.

[Livny 87] M. Livny, S. Khoshafian, H. Boral, "Multi-disk management algorithms," *Proc. of ACM SIGMETRICS*, May 1987.

[Park 86] A. Park and K. Balasubramanian, "Providing Fault Tolerance in Parallel Secondary Storage Systems," Department of Computer Science, Princeton University, CS-TR-057-86, Nov. 7, 1986.

[Patterson 88] D. Patterson, G. Gibson, and R. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)," ACM SIGMOD conference proceedings, Chicago, IL., June 1-3, 1988, pp. 109-116. (Also appeared as Technical Report UCB/CSD 87/391, December 1987.)

[Salem 86] K. Salem and Garcia-Molina, H., "Disk Striping," *IEEE 1986 Int. Conf. on Data Engineering*, 1986.

[Schulze 89] M. Schulze, G. Gibson, R. Katz, and D. Patterson, "How Reliable is a RAID?," Spring COMPCON 89, March 1, 1988, San Francisco, CA, *(next paper in this proceedings).*

[Stonebraker 88] M. Stonrebraker, R. Katz, D. Patterson, and J. Ousterhout, "The Design of XPRS", *Very Large Data Base Conference Proceedings*, August 1988, Long Beach, CA., pp. 318-330.