A Project on High Performance I/O Subsystems

Randy H. Katz, John K. Ousterhout, David A. Patterson,
Peter Chen, Ann Chervenak, Rich Drewes, Garth Gibson, Ed Lee,
Ken Lutz, Ethan Miller, Mendel Rosenblum

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

1. Introduction and Overview

Computing is seeing an unprecedented improvement in performance; over the last five years there has been an order-of-magnitude improvement in the speeds of workstation CPUs. At least another order of magnitude seems likely in the next five years, to machines with 100 MIPs or more. DARPA has already launched a program to develop even larger, more powerful machines, executing as many as 10^{12} operations per second. Unfortunately, we have seen no comparable break-throughs in I/O performance; the speeds of I/O devices and the hardware and software architectures for managing them have not changed substantially in many years [Katz 89].

What will unbalanced improvements in performance mean? Twenty years ago, Gene Amdahl was asked to comment about the Illiac-IV. He noted that while the vector portion of programs might run much faster, a major portion of the programs would run essentially at the same speed. In what has come to be known as Amdahl's Law, he observed that no matter how much faster one piece of the program would go over traditional computers, overall performance improvement is limited by the part of the program that is not improved. Without major increases in I/O performance and reliability we think that transaction processing systems, supercomputers, and high-performance workstations will be unable to achieve their true potential.

Our research group is pursuing a program of research to develop hardware and software I/O architectures capable of supporting the kinds of internetworked workstations and super compute servers that will appear in the early 1990s. The project has three overall goals:

High Performance. We are developing new I/O architectures and a prototype system that can scale to achieve significant factors of improvement in I/O performance, relative to today's commercially available I/O systems, for the same cost. We believe this speedup can be achieved using a combination of arrays of inexpensive personal computer disks coupled with a file system that can accommodate both striped and partitioned file organizations.

High Reliability. To support the high-performance computing of the mid-1990's, I/O

systems cannot just be fast; they must also have large capacity. The unreliability of current disks requires nightly backups on magnetic tape. This process is expensive and un-manageable even with our current systems, and will not scale to meet the needs of the 1990s. By using a small number of extra disks in our disk array to act as standby spares or error recovery disks, we can increase the MTTDL of the magnetic media to the point where media reliability is no longer an issue. With write-once optical disks for user-requested archiving, we hope to eliminate magnetic tapes from the next generation of I/O systems. Our goal is to permit much larger file systems with increased file reliability and decreased human intervention.

Scalable, Multipurpose System. In the past, designers of mainframes and supercomputers have built special-purpose I/O processors that were useful only for one machine, and would need to be redesigned for each new generation, not to mention for each manufacturer. We are structuring the I/O architecture around very-high-speed networks and high-performance file servers in a way that makes diskless mainframes and supercomputers practical. Architects of new computers will be able to guarantee themselves high-performance I/O merely by providing an efficient network interface; they will not need to implement a large collection of controllers and channels for different I/O devices. As demands for I/O increase, the designer has the option of adding more I/O systems to a network if there is enough bandwidth available, or adding extra network interfaces if it is not. Moreover, since the I/O systems will be built from commercially-available computers, they can act either as I/O processors for supercomputers or as "super" file servers on the network.

The expected physical artifacts that will evolve from this work are the following:

(1) A prototype rack-mounted disk array with array controllers of our own design.

(2) A new file system to support the disk array and additional optical disks. This will be implemented in the Sprite network operating system [Nelson 88].

In addition, we are working with Professor Michael Stonebraker and his students to develop a data management system specifically designed to exploit disk arrays, based on POSTGRES [Stonebraker 86].

2. Hardware Platform for High Performance I/O

2.1. High Performance Via Disk Arrays

The large-scale economics of the personal computer and workstation marketplaces, in conjunction with advances in technology, have yielded high-density hard disk assemblies at very low costs. This allows the bandwidth of a disk system to be increased enormously by replacing a small number of large disks with very large numbers of inexpensive small disks [Patterson 88]. Consider a triple-density IBM 3380

disk drive, which can store 7.5 gigabytes in about 24 cubic feet, using about 10,000 watts at a cost of $100,000. Today, a 200-MByte 3.5" disk cartridge (containing media, controller, buffer memory, and interface), taking up only 0.1 cubic feet, using at most 8 watts, is available for less than $1000. Note that the seek times, rotational delays, and transfer rates do not differ by more than a factor of two between these two kinds of disks. Replacing an IBM 3380 by 50 of the smaller disks will yield a comparably priced system with higher capacity (10 GBytes), reduced volume (5 cubic feet), and lower power (500 watts.) However, the greatest advantage of the array is the enormous increase in bandwidth (to 100 Mbytes/sec. or more) that can be achieved by operating all the disks concurrently.

We see three possible ways to structure a disk array: *independent drives*, *synchronous arrays* [Kim 86], and *asynchronous arrays* [Kim 87]. The simplest alternative is the independent-drive approach, where the drives are plugged into one or more servers using off-the-shelf controllers; it would be up to the operating system or application level software to find ways to operate all the drives in parallel. For application domains with large numbers of small independent requests, such as on-line transaction processing, this may well be the best approach. However, it does not necessarily improve the performance of any single application. For example, a program reading a single large file, or a program reading many small files sequentially, may be limited by the latency and bandwidth of a single drive.

The second approach is to build the array so that all disks spin, seek, and transfer in synchrony, a so-called *synchronous array*. Such an organization assumes that data is interleaved across the disks by bit or by byte. Synchronous arrays do not reduce seek time or rotational latency, but they provide high bandwidth by transferring to/from all disks simultaneously. Thus, for a single large application reading or writing a large file, this approach can provide a substantial speedup. The disadvantage of synchronous arrays is that they are difficult and expensive to build, and may not scale readily to arrays with tens or hundreds of disks.

The third approach, an *asynchronous array*, simulates the behavior of a synchronous array using unmodified off-the-shelf disk drives in conjunction with a new controller architecture. A read request is implemented by independent seeks and transfers from each of the underlying disks to a shared buffer in the controller. The controller then interleaves the data returned by the disk drives. This approach yields most of the performance benefits of a synchronous array with less hardware cost. The performance is slightly degraded due to the increase in expected rotational latencies.

These three array structures are well matched to different I/O domains. A reconfigurable array is desirable, especially since the underlying drives are actually asynchronous anyway, and much of the differentiation can be left to file system software. For example, it may make sense to organize the array into several independent clusters each of which is itself a small asynchronous array. The ability to choose the degree of synchronization at run-time leads to even greater flexibility.

2.2. High Reliability Via Disk Arrays

One of the most interesting challenges and opportunities for disk arrays is to make them fault-tolerant [Park 86]. While one might believe that small format disks are not as rugged as larger disks, the opposite appears to be true. A typical 3.5" drive has a mean time between failure of 3.5 years and comes with a 5 year warranty. (The Fujitsu Eagle drive comes only with a 90 day warranty.) Nevertheless, an array of 100 disks will experience a failure 100 times more frequently than any individual disk (about every 300 hours for the arrays we envision); a straightforward design might interleave files across the array, so that a head crash anywhere in the array could make every file inaccessible.

However, arrays of small disks also have a potential to increase reliability at very low cost. We can organize the array into separate groups of disks for recovery purposes (see Figure 1).

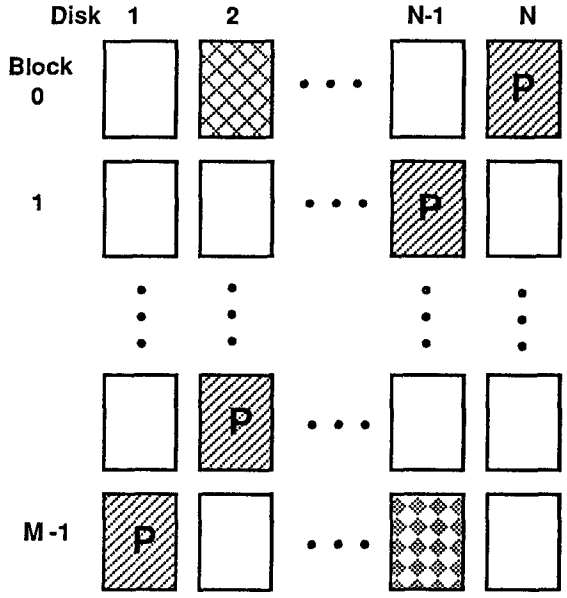

Figure 1: Interleaved Parity Sectors in an N Disk Recovery Group
Alternative interleaved block rows of the array have their parity on different disks within the group. Up to N/2 writes can be serviced at the same time, since one data blocks and one parity block are touched for each logical write, e.g., the write to disk 2, block 0 does not conflict with the write to disk N-1, block M-1.

Soft disk failures are handed by error correcting codes recorded on the disk, combined with the appropriate recovery hardware within the drive electronics. Hard disk failures are actually self-identifying, since disks return status codes that indicate failure. Thus, we can recover information from a failed disk just by adding one extra disk per group to hold the parity calculated horizontally across the group. The cost of the redundancy is just 1/G, where G is the number of disks in a group. If we add a small pool of standby spares to the array, we can almost arbitrarily increase reliability; the question then becomes what are the chances of many reliable components failing within

a very small period.

One potential area of weakness might be the power supply, but much lower power requirements of the disk arrays offers new options. A surge-protected, uninterruptible power supply with battery backup could provide the time to let all disks shut themselves down in the event of a power failure.

Note that such redundancy techniques are only plausible when building a single system from hundreds of components. The only economical solution to large-disk reliability is to double the number of disks. Because of our use of large numbers of inexpensive disks, we expect the parity disks and standby spares to increase the disks in the system by a modest percent.

2.3. High Capacity Via Optical Storage

The supercomputer needs of the 1990's will require not just high-performance file systems, but also file systems with extremely high capacity. We expect the capacity demands within the Computer Science Division at Berkeley to grow from today's 30 Gigabytes to 500 Gigabytes by the year 1995. Although the capacity per dollar of magnetic disks has been improving dramatically, optical storage appears to offer much greater capacity per dollar, particular if jukeboxes are used. We plan to address the issue of how to include optical disks in the storage hierarchy. At the very least, optical disks should provide a less bulky, more efficient, and more convenient archiving mechanism than magnetic tape (e.g., no operator assistance should be needed to retrieve archived data from an optical disk in an on-line jukebox).

3. Operating and File System Issues in High Performance I/O

3.1. Multipurpose Via Networking

A key to separating I/O architecture from CPU architecture is the availability of networks and latencies equivalent to high-performance backplanes. Such networks will be available within a few years. The technical challenge is to provide controller and system architectures that allow very high speed networks to be fully utilized. Our experience with the Sprite operating system is that copying costs are the major obstacle to high protocol bandwidth. For example, when data arrives at a node, the network controller just places it in a buffer; invariably, the operating system ends up copying it from the buffer to its desired destination (e.g., a cache block). Sprite's current protocols can utilize about 2 Mbits/second of burst network bandwidth per MIP of CPU power, but we believe that this can be increased substantially, so a single 100 MIP file server can provide a supercomputer with I/O bandwidth of a Gigabyte/second or more. The most promising approach appears to be a redesign of network controllers to reduce copying. For example, if packets are tagged with very simple type information interpreted by the controller, it could subdivide incoming packets and place various header and data fields

in separate areas indicated by the packet.

3.2. High Performance Via Caching

Our previous work on Sprite has already demonstrated the benefits of caching on a small scale: Sprite eliminates most of the performance impact of remote file access by caching recently-accessed file blocks in the main memories of client and server machines. We are continuing to explore and extend the Sprite caching techniques by measuring the effects of even larger cache sizes and tuning the Sprite caching mechanisms for higher performance machines.

Caching is effective in reducing network and disk accesses for reads, but it still leaves open the issue of writes. A write-through file cache has serious performance implications, but a write-back cache has reliability problems in the face of system crashes. One possible solution is to use a special disk or disk array as a cache backup, and continuously update this disk with the contents of the cache. Information could be written sequentially to the disk in cache order, so that there would be only a single seek per scan of the cache: whereas writing through to files would require at least one seek per file. A disadvantage of this approach is that it might require a backup disk for each cache.

4. Project Status

To date, we have focussed on evaluating the concept of a redundant array of inexpensive disks, to understand its strengths and weaknesses with respect to performance and availability versus alternative disk system organizations. In particular, we have completed: (1) a performance evaluation of software implementations of two alternative redundancy architectures (RAID Level 1 mirrored disks and RAID Level 5 rotated parity) and (2) a more thorough evaluation of the reliability/availability attributes of RAIDs. These are described below.

The performance measurements, described in [Chen 89], were taken on a large configuration Amdahl system. This system consisted of a 15 ns cycle time dual processor connected to 64 channels and IBM 3380 class disk devices. In the original RAID paper, a simple analytical model was proposed to rank the various organizations. The Amdahl experiment was geared towards verifying this simple model in a realistic operational environment. Only the two most promising array organizations were considered: mirroring (RAID Level 1) and rotated parity (RAID Level 5). As predicted, rotated parity provided superior performance for large accesses (as we would expect to find in supercomputer applications), while mirroring did better for small accesses. Some effects were not predicted by our original model. Mirroring benefited from *seek scheduling*, which allows the controller to position the closest of two arms, thus reducing average seek time. Rotated parity suffered from *stripe breakage*: a large write is turned into a series of full stripe writes with partial stripe writes at the beginning and at the end. The latter degrade performance because they are essentially small writes: old data

must be read, the differences computed, the old parity accessed, the differences applied, and the new parity written back. However, the most pronounced effect was the performance enhancement due to large versus small accesses: there was more than an order of magnitude difference in the disk bandwidth achieved. By use of the log structured file system and large file caches, we expect our operating system to be effective in turning small accesses into large accesses for presentation to the disk array.

The availability study, described in [Schulze 88; Schulze 89], demonstrated the importance of understanding support component (i.e., controller hardware, cables, power supplies, fans, etc.) reliability and its contribution to mean time to data loss (MTTDL). While RAID techniques may achieve media availabilities measured in millions of hours, the overall system availability will be substantially less. By using a simple technique which we have called *orthogonal RAID*, basically calculating parity across disk strings rather than within strings, the availability of a multidisk array can be as good as a single large expensive disk. By replicating some of the support components, such as power supplies and fans, the availability can be significantly improved, without introducing the added complexity, both hardware and software, of duplexed disk controllers.

However, to seriously consider constructing disk subsystems out of thousands of disks with acceptable availability, redundancy techniques beyond RAID Level 5 are needed. These techniques must be able to recover from two (or more) failed disks within a recovery group. [Gibson 88; Gibson 89] describes alternative codes that give high data availability while minimizing the capacity overhead associated with redundant data.

We have just completed the implementation of a first hardware prototype, the purpose of which is to serve as a testbed for developing the array controller software (e.g., logical to physical block mappings, parity calculations, disk scheduling, disk reconstruction algorithms). The hardware is a 128 MByte SUN-4, with four Interphase Host Bus Adapters (8 SCSI strings), and thirty two 330 MBytes 5.25" Imprimis Wren-IV disk devices. The Sprite operating system has been ported to the hardware, and we are currently debugging and tuning a "striping" device driver for a rotated parity array. A performance evaluation of the prototype is underway.

We also begun the design of a second prototype, which will incorporate an array controller of our own design rather than off-the-shelf components. The goals of this effort are to produce a single board design that can (1) scale to (at least) a 100 disk device array, (2) provide a generic bus interface to facilitate tracking advances in memory and bus technology, while easing the retargeting of the design to alternative host systems, and (3) utilize 3.5" SCSI/SCSI-2 disk devices. A Unix/Mach device driver will be provided, to which the array will appear as a single large, extremely high performance disk. We also expect to support a superset interface for Sprite's experimental log structured file system, revealing more controller details to higher levels of software.

5. Summary

We have briefly described a coordinated hardware-software attack on the I/O

bottleneck. Relying on new ideas in operating system for file caching and I/O buffering, and exploiting the arrival of low-cost disks, we plan to demonstrate significant factors of improvement in performance and reliability over current commercially-available systems. An important aspect of the project is the development of hardware and software prototypes to demonstrate our claims.

The first RAID prototype and its use for database applications is currently funded by the National Science Foundation under grant # MIP-8715235. The larger scale prototype, targeted towards high performance file service and supercomputer workloads, is being funded by a joint DARPA and NASA contract. In addition, we have received support from the State of California MICRO program, in conjunction with our industrial affiliates: DEC, Hewlett-Packard, Imprimis Technology, Intel Scientific Computers, IBM, Kodak, and Sun Microsystems.

6. References

Chen 89 P. M. Chen, "An Evaluation of Redundant Arrays of Inexpensive Disks using an Amdahl 5890," Technical Report No. UCB/CSD 89/506, (May 1989), 49 pps.

Gibson 88 G. Gibson, L. Hellerstein, R. Karp, R. Katz, D. Patterson, "Coding Techniques for Handling Failures in Large Disk Arrays," Technical Report No. UCB/CSD 88/477, (December 1988), 29 pps.

Gibson 89 G. Gibson, L. Hellerstein, R. Karp, R. Katz, D. Patterson, "Failure Correction Techniques for Large Disk Arrays," Proceedings Third Intr. Conf. on Arch. Supp. for Prog. Lang. and Operating Systems, Boston, MA, (April 1989).

Katz 89 R. Katz, G. Gibson, D. Patterson, "Disk System Architectures for High Performance Computing," *Proceedings of the IEEE*, Special Issue on Supercomputing, (January 1990). Also available as Technical Report No. UCB/CSD 89/497.

Kim 86 M. Y. Kim, "Synchronized Disk Interleaving," I.E.E.E. Transactions on Computers, V C-35, N 11, (November 1986), pp. 978-988.

Kim 87 M. Y. Kim, A. N. Tantawi, "Asynchronous Disk Interleaving," IBM T. J. Watson Research Center TR RC 12497 (#56190), Yorktown Heights, NY, (February 1987).

Nelson 88 M. Nelson, B. Welch, J. Ousterhout, "Caching in the Sprite Network File System," *ACM Transactions on Computers Systems*, V. 6, N. 1, (February 1988), pp. 15-24.

Park 86 A. Park, K. Balasubramanian, "Providing Fault Tolerance In Parallel Secondary Storage Systems," Computer Science TR 057-86, Princeton University, (November 1986).

Patterson 88 D. A. Patterson, G. Gibson, R. H. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)," A.C.M. SIGMOD Conference, Chicago, IL, (May 1988), pp. 109-116.

Schulze 88 M. Schulze, "Considerations in the Design of a RAID Prototype," Technical Report No. UCB/CSD 88/448, 35 pps.

Schulze 89 M. Schulze, G. Gibson, R. Katz, D. Patterson, "How Reliable is a RAID?," Proceedings IEEE Spring Computer Conference, S.F., CA, (February 1989).

Stonebraker 86 M. R. Stonebraker, L. A. Rowe, "The Design of POSTGRES," A.C.M. SIGMOD Conference, Washington, D.C., (May 1986).