RAID Project Bibliography

The following bibliography describes papers about or closely related to the Berkeley Redundant Arrays of Inexpensive Disks (RAID) Project. The RAID project is closely related to Berkeley's XPRS and SPRITE projects and participates in the broader Mammoth project. The abstracts to these papers form an overview of the project.

To obtain copies of any of these papers, contact Bob Miller, 533 Evans Hall, University of California, Berkeley CA 94720, bmiller@sprite.Berkeley.EDU, (415) 642-6037.

[Patterson87] Patterson, David A., Garth A. Gibson, Randy H. Katz, ''A Case for Redundant Arrays of Inexpensive Disks (RAID),'' University of California Technical Report UCB/CSD 87/391, Berkeley CA, December 1987, preprint of [Patterson88].

[Ousterhout88] John K. Ousterhout, Andrew R. Cherenson, Frederick Douglis, Michael N. Nelson, Brent B. Welch, ''The Sprite Network Operating System,'' *Computer,* February 1988.

[Katz88] R. H. Katz, J. K. Ousterhout, D. A. Patterson, M. R. Stonebraker, ''A Project on High Performance I/O Subsystems,'' *IEEE Computer Society Technical Committee on Database Engineering Newsletter,* **11,** 1, March 1988.

[Stonebraker88a] Stonebraker, M. R., R. H. Katz, D. A. Patterson, J. K. Ousterhout, ''The Design of XPRS,'' University of California Technical Report UCB/ERL M88/19, Electronics Research Lab., Berkeley CA, March 1988.

[Patterson88] Patterson, David A., Garth A. Gibson, Randy H. Katz, ''A Case for Redundant Arrays of Inexpensive Disks (RAID),'' *Proceedings of the 1988 ACM SIGMOD Conference on Management of Data,* Chicago IL, June 1988, pp 109-116.

> Increasing performance of CPUs and memories will be squandered if not matched by a similar performance increase in I/O. While the capacity of Single Large Expensive Disks (SLED) has grown rapidly, the performance improvement of SLED has been modest. Redundant Arrays of Inexpensive Disks (RAID), based on the magnetic disk technology developed for personal computers, offers an attractive alternative to SLED, promising improvements of an order of magnitude in performance, reliability, power consumption, and scalability. This paper introduces five levels of RAIDs, giving their relative cost/performance, and compares RAID to an IBM 3380 and a Fujitsu Super Eagle.

[Stonebraker88b] Stonebraker, M. R., R. H. Katz, D. A. Patterson, J. K. Ousterhout, ''The Design of XPRS,'' *Proc. 14th Int. Conf. on Very Large Databases (VLDB),* Los Angeles CA, August 1988.

[Schulze88] Schulze, Martin E., ''Considerations in the Design of a RAID Prototype,'' University of California Technical Report, UCB/CSD 88/448, Berkeley CA, August 1988, Master's Thesis.

> Disk drive arrays are receiving attention from computer researchers looking for higher

performance mass storage. RAID (Redundant Array of Inexpensive Disks) appears to offer benefits over storage systems based on large format disks. This paper investigates disk drive and disk support hardware failures as factors in the data integrity of RAID. Through the use of redundancy in the disk array, support hardware failures can be eliminated as a major factor in the data integrity of RAID. This paper also presents some practical considerations in the design of a RAID prototype and discusses how such a prototype could be constructed at Berkeley in the near future. The prototype will most likely be based on 5.25 inch disk drives and the Small Computer System Interface (SCSI) bus.

[Ousterhout88] Ousterhout, John K., Fred Douglis, ‘‘Beating the I/O Bottleneck: A Case for Log-Structured File Systems,’’ University of California Technical Report UCB/CSD 88/467, Berkeley CA, October 1988, preprint of [Ousterhout89].

[Chen88] Chen, Peter M., Garth A. Gibson, Randy H. Katz, David A. Patterson, Martin E. Schulze, ‘‘Two Papers on RAIDs,’’ University of California Technical Report UCB/CSD 88/479, Berkeley CA, December 1988, preprint of [Patterson 89] and [Schulze89].

A RAID is a Redundant Array of Inexpensive Disk, a new way to organize small format disk devices to drastically increase I/O bandwidth. In this technical report, we describe the RAID concept, the basic RAID levels, a more detailed analysis of RAID performance on reliability, and our initial prototyping plans.

[Gibson88] Gibson, Garth A., Lisa Hellerstein, Richard M. Karp, Randy H. Katz, David A. Patterson, ‘‘Coding Techniques for Handling Failures in Large Disk Arrays’’, University of California Technical Report UCB/CSD 88/477, Berkeley CA, December 1988, preprint of [Gibson89a].

[Ousterhout89] Ousterhout, John K., Fred Douglis, ‘‘Beating the I/O Bottleneck: A Case for Log-Structured File Systems,’’ *Operating Systems Review*, **23**, 1, January 1989, pp 11-28.

CPU speeds are improving at a dramatic rate, while disk speeds are not. This technology shift suggests that many engineering and office applications may become so I/O-limited that they cannot benefit from further CPU improvements. This paper discusses several techniques for improving I/O performance, including caches, battery-backed-up caches, and cache logging. We then examine in particular detail an approach called log-structured file systems, where the file system’s only representation on disk is in the form of an append-only log. Log-structured file systems potentially provide order-of-magnitude improvements in write performance. When log-structured file systems are combined with arrays of small disks (which provide high bandwidth) and large main-memory file caches (which satisfy most read accesses), we believe it will be possible to achieve 1000-fold improvements in I/O performance over today’s systems.

[Patterson89] Patterson, David A., Peter M. Chen, Garth A. Gibson, Randy H. Katz, ‘‘Introduction to Redundant Arrays of Inexpensive Disks (RAID),’’ *Proceedings of the Spring 1989 COMPCON*, San Francisco CA, Spring 1989.

[Schulze89] Schulze, Martin E., Garth A. Gibson, Randy H. Katz, David A. Patterson, ‘‘How Reliable is a RAID?,’’ *Proceedings of the Spring 1989 COMPCON*, San Francisco CA, Spring 1989.

Disk arrays provide the promise of greatly increased transfer bandwidth at low cost. But without additional data redundancy, an array can suffer from significantly degraded reliability. In this paper we more closely examine the reliability of RAID systems and find that although phenomenal reliability disk arrays can not be attained with data redundancy alone, RAID system reliability can be made better than conventional large disks with little extra hardware.

[Douglis89] Douglis, F., J. K. Ousterhout, ''Log-Structured File Systems,'' *Proceedings of the Spring 1989 COMPCON,* San Francisco CA, Spring 1989, pp 124-129.

CPU speeds are improving at a dramatic rate, while disk speeds are not. This technology shift suggests that many engineering and office applications may become so I/O-limited that they cannot benefit from further CPU improvements. This paper discusses several techniques for improving I/O performance, including caches, battery-backed-up caches, and cache logging. We then examine in particular detail an approach called log-structured file systems, where the file system's only representation on disk is in the form of an append-only log. Log-structured file systems potentially provide order-of-magnitude improvements in write performance. When log-structured file systems are combined with arrays of small disks (which provide high bandwidth) and large main-memory file caches (which satisfy most read accesses), we believe it will be possible to achieve 1000-fold improvements in I/O performance over today's systems.

[Katz89a] Katz, R.H., G.A. Gibson, D.A. Patterson, ''Disk system architectures for high performance computing,'' Tech. Report UCB/CSD 89/497, University of California, Berkeley CA, March 1989, preprint of [Katz89d].

[Gibson89a] Gibson, Garth A., Lisa Hellerstein, Richard M. Karp, Randy H. Katz, David A. Patterson, ''Coding Techniques for Handling Failures in Large Disk Arrays,'' *Third Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS III),* Boston MA, April 1989, pp. 123-132.

The ever increasing need for I/O bandwidth will be met with ever larger arrays of disks. These arrays require redundancy to protect against data loss. This paper examines alternative choices for encodings, or codes, that reliably store information in disk arrays. Codes are selected to maximize mean time to data loss or minimize disks containing redundant data, but are all constrained to minimize performance penalties associated with updating information or recovering from catastrophic disk failures. We show codes that give highly reliable data storage with low redundant data overhead for arrays of 1000 information disks.

[Chen89] Chen, Peter M., ''An Evaluation of Redundant Arrays of Disks Using an Amdahl 5890,'' University of California Technical Report UCB/CSD 89/506, Berkeley CA, May 1989, Master's Thesis.

[Katz89b] Katz, Randy H., ''Trends in Disk System Architecture,'' *UNIX Review Magazine*, Invited Paper, September 1989.

[Katz89c] Katz, Randy H., John K. Ousterhout, David A. Patterson, Peter M. Chen, Ann L. Chervenak, Rich Drewes, Garth A. Gibson, Ed K. Lee, Ken C. Lutz, Ethan L. Miller, Mendel Rosenblum, ''A project on high performance I/O subsystems'', *ACM Computer Architecture News*, **17**, 5, Invited paper, September 1989, pp 24-31.

[Gibson89b] Gibson, Garth A., ''Performance and Reliability in Redundant Arrays of Inexpensive Disks,'' University of California Technical Report UCB/CSD 89/535, Berkeley, CA, September 1989, preprint of [Gibson89c].

[Gibson89c] Gibson, Garth A., ''Performance and Reliability in Redundant Arrays of Inexpensive

Disks,'' *Proc. of the 1989 Computer Measurement Group (CMG) Annual Conference,* Reno, Nevada, December 11-15, 1989.

Arrays of magnetic disks, or disk arrays, are becoming more common for a variety of reasons. As overall system processing speeds grow at a rate far ahead of magnetic disk technology, disk arrays can provide much needed increases in data throughput (megabytes per second) and access throughput (accesses per second). Disk arrays capitalize on the smaller form-factor magnetic disks that are rapidly becoming denser and cheaper per megabyte than conventional IBM 3380 class disks. Moreover, a single high quality small disk design can be used as the basis of a range of price and performance disk products when arrays are employed. Disk arrays also offer the opportunity to achieve high data reliability at a lower cost than conventional methods based on complete duplication of data. After an extended review of disk array motivations, organizations, and performance expectations, this paper examines the data reliability of a Redundant Array of Inexpensive Disks (RAID) in comparison to conventional disks. In particular, we include support hardware failures (power, cooling, controllers and cabling), spare pool depletion, and non-exponential disk lifetimes. This paper finds that data redundancy deals with disk failures quite effectively, leaving support hardware failures as the reliability bottleneck; that a relatively small spare pool works quite well; and that although reliability is sensitive to disk lifetime distributions, the effect is only pronounced for small spare pools with slow refill rates.

[Katz89d] Katz, R. H., G. A. Gibson, D. A. Patterson, ''Disk system architectures for high performance computing,'' *Proceedings of the IEEE,* **77**, 12, December 89, pp. 1842-1858.

Because of the mismatch between I/O and CPU speeds, high-performance computers have long been forced to confront the fundamental I/O bottleneck. As processing power and memory size continue to grow rapidly for micro- and minicomputers, they too will become I/O limited. A number of hardware and software approaches such as parallel read-out disks, expanded storage (for example, solid-state disks), and disk striping, have been used to increase I/O bandwidth and thus narrow the CPU-I/O performance gap. In addition, new developments driven by advances in small-diameter (5.25-in and 3.5-in) disk drives, promise very high I/O bandwidth if large numbers of devices can be organized into arrays of disks. In this paper we review the state of the art in disk devices and I/O controllers, and will describe new approaches for very high-performance I/O based on redundant arrays of inexpensive disks (RAIDs).

[Stonebraker89] Stonebraker, M. R., ''Parallelism in XPRS,'' 1989.

[Seltzer90a] Seltzer, Margo I., Peter M. Chen, John K. Ousterhout, ''Disk Scheduling Revisited,'' *Proc. of the Winter '90 USENIX Technical Conference,* Washington, DC, January 22-26, 1990.

Since the invention of the movable head disk, people have improved I/O performance by intelligent scheduling of disk accesses. We have applied these techniques to systems with large memories and potentially long disk queues. By viewing the entire buffer cache as a write buffer, we can improve disk bandwidth utilization by applying some traditional disk scheduling techniques. We have analyzed these techniques, which attempt to optimize head movement and guarantee fairness in response time, in the presence of long disk queues. We then propose two algorithms which take rotational latency into account, achieving disk bandwidth utilizations of nearly four times a simple first come first serve algorithm. One of these two algorithms, a weighted shortest total time first, is particularly applicable to a file server environment because it guarantees that all requests get to disk within a specified time window.

[Stonebraker90] Stonebraker, M. R., G. A. Schloss, ''Distributed RAID - A New Multiple Copy Algorithm,'' *Proc. of the 6th IEEE Int. Conf. on Data Engineering,* April 1990.

[Katz90] Randy H. Katz, ''Very High Performance Disk Subsystems based on Arrays of Disks,'' *Proceedings of U. S. Decus Symposia,* New Orleans, LA, May 1990.

[Chen90a] Chen, Peter M., Garth A. Gibson, Randy H. Katz, David A. Patterson, ''An Evaluation of Redundant Arrays of Disks Using an Amdahl 5890,'' *Proc. of the 1990 ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems,* Boulder CO, May 1990.

> Recently we presented several disk array architectures designed to increase the data rate and I/O rate of supercomputing applications, transaction processing, and file systems [Patterson 88]. In this paper we present a hardware performance measurement of two of these architectures, mirroring and rotated parity. We see how throughput for these two architectures is affected by response time requirements, request sizes, and read to write ratios. We find that for applications with large accesses, such as many supercomputing applications, a rotated parity disk array far outperforms traditional mirroring architecture. For applications dominated by small accesses, such as transaction processing, mirroring architectures have higher performance per disk than rotated parity architectures.

[Chen90b] Chen, Peter M., David A. Patterson, ''Maximizing Performance in a Striped Disk Array,'' *Proc. of the 1990 ACM SIGARCH 17th Ann. Int. Symp. of Computer Architecture,* Seattle WA, May 1990.

> Improvements in disk speeds have not kept up with improvements in processor and memory speeds. One way to correct the resulting speed mismatch is to stripe data across many disks. In this paper, we address how to stripe data to get maximum performance from the disks. Specifically, we examine how to choose the striping unit, i.e. the amount of logically contiguous data on each disk. We synthesize rules for determining the best striping unit for a given range of workloads.
>
> We show how the choice of striping unit depends on only two parameters: 1) the number of outstanding requests in the disk system at any given time, and 2) the average positioning time $\times$ data transfer rate of the disks. We derive an equation for the optimal striping unit as a function of these two parameters; we also show how to choose the striping unit without prior knowledge about the workload.

[Lee90] Lee, Edward K., ''Software and Performance Issues in the Implementation of a RAID Prototype,'' University of California Technical Report UCB/CSD 90/573, Berkeley CA, May 1990, Master's Thesis.

> Due to recent advances in CPU performance, I/O systems are increasingly limiting the performance of modern computer systems. The RAID (Redundant Array of Inexpensive Disks) group at Berkeley is designing storage systems based on disk arrays with an emphasis toward high performance and reliability. This report is the result of knowledge gained in designing and implementing software for RAID-I (RAID the First), a prototype RAID system designed and built at Berkeley. Specifically, this report describes the software organization of RAID-I, and investigates the performance consequences of various parity placement schemes applicable to RAID systems. This paper assumes that the reader is familiar with RAID systems.

[Chervenak90] Chervenak, Ann L., ''Performance Measurements of the First RAID Prototype,'' University of California, Technical Report UCB/CSD 90/574, Berkeley CA, May

1990, Master's Thesis.

This paper examines the performance of RAID the First, a prototype disk array built by the RAID group at U.C. Berkeley. A hierarchy of bottlenecks was discovered in the system that limit overall performance. The most serious is the memory system contention on the Sun4/280 host CPU, which limits array bandwidth to 2.3 MBytes/sec. The array performs more successfully on small random operations, achieving nearly 300 I/Os per second before the Sun4/280 becomes CPU-limited. Other bottlenecks in the system are the VME backplane, bandwidth on the disk controller, and overheads associated with the SCSI protocol. All are examined in detail.

The main conclusion of this report is that to achieve the potential bandwidth of arrays, more powerful CPUs will not suffice. Just as important are adequate host memory bandwidth and support for high bandwidth on disk controllers. Current disk controllers are more often designed to achieve large numbers of small random operations, rather than high bandwidth. Operating systems also need to change to support high bandwidth from disk arrays. In particular, they should transfer data in larger blocks, and should support asynchronous I/O to improve sequential write performance.

[Rosenblum90] Rosenblum, Mendel, John K. Ousterhout, ''The LFS Storage Manager,'' *Proc. of the Summer '90 USENIX Technical Conference*, Anaheim, CA, June 1990.

Advances in computer system technology in the areas of CPUs, disk subsystems, and volatile RAM memory are combining to create performance problems existing file systems are ill-equipped to solve. This paper identifies the problems of using the existing UNIX file systems on 1990's technology and presents an alternative file system design that can use disks an order-of-magnitude more efficiently for typical UNIX workloads. The design, named LFS for log-structured file system, treats the disk as a segmented append-only log. This allows LFS to write many small changes to disk in a single large I/O while still maintaining the fast file reads of existing file systems. In addition, the log-structured approach allows near instantaneous file system crash recovery without coupling CPU and disk performance with synchronous disk writes. This paper describes and justifies the major data structures and algorithms of the LFS design. We compare an implementation of LFS in the Sprite distributed operating system to SunOS's file system running on the same hardware. For tests that create, destroy, or modify files at a high rate, LFS can achieve an order-of-magnitude speedup over SunOS. In spite of its obvious write-optimization, LFS's read performance matches or exceeds the SunOS file system under most common UNIX workloads.

[Seltzer90b] Seltzer, Margo, Michael Stonebraker, ''Transaction Support in Read Optimized and Write Optimized File Systems,'' *Proc. 16th Int. Conf. on Very Large Data Bases, VLDB*, August 1990, pp 174-185.

This paper provides a comparative analysis of five implementations of transaction support. The first of the methods is the traditional approach of implementing transaction processing within a data manager on top of a read optimized file system. The second also assumes a traditional file system but embeds transaction support inside the file system. The third model considers a traditional data manager on top of a write optimized file system. The last two models both embed transaction support inside a write optimized file system, each using a different logging mechanism. Our results show that in a transaction processing environment, a write optimized file system often yields better performance than one optimized for reads. In addition, we show that file system embedded transaction managers can perform as well as data managers when transaction throughput is limited by I/O bandwidth. Finally, even when the CPU is the critical resource, the difference in performance between a data manager and an embedded system is much smaller than previous work has shown.

[Schloss90] Schloss, G. A., M. R. Stonebraker, ‘‘Highly Redundant Management of Distributed Data,’’ to appear *Proc. of the 1st IEEE Workshop on Management of Replicated Data,* November 1990.

[Schloss91] Schloss, G. A., M. R. Stonebraker, ‘‘Highly Redundant Management of Distributed Data - position paper,’’ to appear *Newsletter of IEEE Technical Committee on Operating Systems,* **4,** 3, 1991.

[Lee91] Lee, Edward K., Randy H. Katz, ‘‘Performance Consequences of Parity Placement in Disk Arrays,’’ to appear *Fourth Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS IV),* Palo Alto CA, April 1991.

> Due to recent advances in CPU and memory system performance, IO systems are increasingly limiting the performance of modern computer systems. Redundant Arrays of Inexpensive Disks (RAID) have been proposed by [Patterson88] to meet the impending IO crisis. RAID’s substitute many small inexpensive disks for a few large expensive disks to provide higher performance (both transfer rate and IO 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. RAID’s 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 at relatively large request sizes (a couple of hundred 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.

[Seltzer91] Margo Seltzer, Michael Stonebraker, ‘‘Read Optimized File System Designs: A Performance Evaluation,’’ submitted to *Data Engineering,* 1991.

> This paper presents a performance comparison of several allocation policies for producing read optimized file systems. The file systems are designed to provide high bandwidth between disks and main memory by taking advantage of parallelism in an underlying disk array, catering to large units of transfer, and minimizing the bandwidth dedicated to the transfer of meta data. Our goal is to provide efficient support for large files without sacrificing small file efficiency, either in terms of disk capacity or read/write performance. All of the file systems described use a multiblock allocation strategy. That is, rather than supporting a fixed block size, the file systems support multiple block sizes. In this way, both large and small files may be allocated efficiently.
>
> Simulation results show that such multiblock policies result in systems that are able to utilize a large percentage of the underlying disk bandwidth; more than 90% in sequential cases and 20-80% under application specific workloads. Furthermore, this high throughput is attainable without suffering poor disk utilization due to internal and external disk fragmentation. As general purpose systems, which have traditionally catered to small files and a time sharing environment, are called upon to satisfy larger and more data intensive applications such as databases and supercomputing, these multiblock policies offer an opportunity to provide superior performance to a larger class of users.

[Chervenak91] Chervenak, Ann L., Randy H. Katz, ‘‘Performance of a RAID Prototype,’’ submitted to *1991 ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems,* San Diego CA, May 1991.

This paper examines the performance of a prototype disk array. We hoped that its performance would be limited only by the number of disks in the system and by their capabilities. In reality, bottlenecks existed in each layer of the system hierarchy. This paper describes the combination of SCSI bus traces, Sprite Operating System kernel traces, user programs and deduction that revealed these performance bottlenecks.

It also describes each bottleneck in detail. The most serious one is the memory system on the Sun4/280 host machine, which limits array bandwidth when running the Sprite operating system to 2.3 MBytes/sec for large, sequential operations. The array performs more successfully on small random operations, achieving nearly 300 I/Os per second on 14 disks before the Sun4/280 becomes CPU-limited. Other system bottlenecks described here are the VME backplane, bandwidth on the disk controller, and overheads associated with the SCSI protocol.

Our main conclusion is that to achieve the bandwidth possible from arrays, increasing the number of disks and using more powerful host CPUs will not suffice. Equally important are adequate host memory bandwidth and high bandwidth on disk controllers. Operating systems should transfer data in larger blocks and avoid unnecessary copy and cache flush operations.

[Gibson91] Gibson, Garth A., ‘‘Redundant Disk Arrays: Reliable, Parallel Secondary Storage,’’ University of California Technical Report UCB/CSD 90/615, Berkeley CA, March 1991, PhD Dissertation.

Although processor and memory technology advances have led to rapid increases in computational speeds, secondary storage technology has kept up only in capacity. Arrays of disks leverage parallelism to provide scalable increases in secondary storage performance. Coupled with emerging small disk technology, disk arrays provide the cost and capacity of current disk subsystems, but many times the performance. Unfortunately, arrays of hundreds of small disks may have hundreds of times the failure rate of the single large disk they replace. Redundant Arrays of Inexpensive Disks (RAID) use simple redundancy to provide high data reliability. This dissertation investigates the data encoding, performance, and reliability of redundant disk arrays.

The organization of redundant data in a disk array is a coding problem. Codes as simple as parity effectively correct single, self-identifying disk failures because disk repair durations are orders of magnitude shorter than disk lifetimes. For very large or highly reliable disk arrays, more complex and expensive codes can be used to correct multiple simultaneous failures.

Disk arrays that stripe data across multiple disks substantially reduce response time for large transfers. Striping also distributes independent accesses across disks so that throughput for small accesses is maximized. Performance surrendered to redundant data maintenance can be negligible except for small writes where disk utilization may increase by up to a factor of four and penalize throughput.

Disk failure characteristics are critical to the effectiveness of redundancy in disk arrays. This dissertation presents evidence that disk lifetime can be modeled as a random variable with an exponential distribution. Models for array lifetime, the time until data is lost, are presented and evaluated against event-driven simulation. These models incorporate effects of dependent disk failures (shared support hardware), as well as the effects of a limited pool of on-line, spare disks. For large redundant disk arrays, support hardware and redundancy groups can be organized to protect against many dependent disk failures. Similarly, a relatively small spare pool effectively minimizes failure recovery time so that array lifetime is maximized.

Authors can be reached electronically at the following net addresses
(where x@{a,b}.z means x@a.z or x@b.z):

Peter M. Chen (pmchen@{cs,sprite,ginger,ernie}.berkeley.edu),
Ann L. Chervenak (alc@{sprite,ginger,ernie}.berkeley.edu),
Fred Douglis (douglis@{cs,sprite,ginger,ernie}.berkeley.edu, douglis@cs.vu.nl),
Rich Drewes (drewes@{sprite,ginger}.berkeley.edu),
Garth A. Gibson (gibson@{cs,sprite,ginger,ernie}.berkeley.edu),
Lisa Hellerstein (hstein@ernie.berkeley.edu),
Randy H. Katz (randy@{cs,sprite,ginger,ernie}.berkeley.edu),
Richard M. Karp (karp@{cs,ginger,ernie}.berkeley.edu),
Edward K. Lee (eklee@{cs,sprite,ginger,ernie}.berkeley.edu),
Ken C. Lutz (lutz@{sprite,ginger}.berkeley.edu),
Ethan L. Miller (elm@{cs,sprite,ginger,ernie}.berkeley.edu),
John K. Ousterhout (ouster@{cs,sprite,ginger,ernie}.berkeley.edu),
David A. Patterson (pattrsn@{cs,sprite,ginger,ernie}.berkeley.edu),
Mendel Rosenblum (mendel@{sprite,ginger,ernie}.berkeley.edu),
Gary A. Schloss (schloss@{postgres,sprite}.berkeley.edu),
Martin E. Schulze (schulze%cxcad.DEC@decwrl.dec.com),
Margo I. Seltzer (margo@{cs,sprite,ginger,ernie}.berkeley.edu),
Michael R. Stonebraker (mike@{cs,sprite,ginger,ernie}.berkeley.edu).