

A Project Summary

The role played by storage systems is becoming increasingly important as computing systems accelerate their transformation from “number-crunching” devices to “information appliances.” If there is a single number that can best summarize the technology motivation for my research, it is the tremendous growth of disk areal density. At a phenomenal annual rate of 60%, this growth is fueling almost all the progress that is being made today in the magnetic storage industry, which is providing one of the most important infrastructures of the information revolution.

Meanwhile, this density growth has reached such a critical stage that it is outpacing our existing ways of understanding, architecting, and using secondary storage. I highlight two important implications of the areal density growth: larger capacity and lower cost. As disk capacity rapidly increases, ironically, our ability to extract bits out of the storage system is in fact deteriorating because more bits are forced through the mechanical funnel. If nothing is done about the trend, all our computing systems will approach the asymptotic mechanical speeds. The second implication is that as disks become cheaper and as our society continues its analog-to-digital transformation, information that is not traditionally associated with computing is being increasingly stored on inexpensive and ubiquitous disks. Just to name a few, these “invisible disks” may include TVs, cameras, telephones, etc. How to manage this emerging chaos of “invisible bits” is a serious challenge.

Facing these challenges, the state-of-art is woefully inadequate. I enumerate three high level weaknesses. The first is a lack of innovation in storage architectures: as processor architectures have gone through several generations of revolution, the basic mechanism of a disk has remained unchanged for decades. Rigid form factors only make things worse. The second is a lack of coordination of the parties involved in the construction of storage systems: operating systems are being designed for a mythical generic disk and disks are being built without regard to what/how file systems and applications will eventually use them. The third is a failure to recognize the aforementioned “invisible bits” problem as a network storage issue. As a result, ad hoc application-specific approaches are being thrown together for the plethora of emerging devices and they all require all manners of inconvenient manual involvement. Together, these weaknesses constitute a set of “blind spots” of I/O research.

I spent most of my graduate student years designing and building several distributed file systems at Berkeley, which had a strong tradition in file system innovations. I also worked on several high-performance networking projects, which provided insights that were directly applicable to storage systems as well. It was from these experiences that the blind spots described above became painfully apparent. I plan to focus my research on two areas: very low latency I/O and ubiquitous storage. While RAIDs can bring high bandwidth, low latency remains an elusive goal. Much of the computer systems work today is based on the assumption that disk accesses are intrinsically limited by long latency. By fundamentally changing this assumption, the low latency I/O work may fundamentally change the way we program secondary storage and can lead to new applications. Ubiquitous storage refers to providing a common distributed storage abstraction to all devices that have to rely on ad hoc local storage solutions today. By providing a common mechanism for managing migration, coherence, and backup, we may be able to conquer the tyranny of the chaos of “invisible bits.” To accomplish these research objectives, I will address each of the blind spots, by conducting the science of studying how to architect a “better” disk that is more latency friendly, by employing end-to-end vertical integration to build high-performance “information appliances”, and by studying how to apply and extend the decentralized network file system experience to a federation of loosely connected ad hoc devices.

File systems are not only the most visible parts of operating systems, they also intimately interact with many parts of the rest of the systems such as networking and virtual memory. The principles involved in building a high-performance, scalable, and reliable distributed file system are directly applicable to a larger context. Therefore, I expect my research thrust of distributed storage systems will work well within Princeton’s recent effort of revitalizing operating systems education. More specifically, my education plan addresses the following four issues. First, I believe systems education today emphasizes more on details than on methodology (doing the “science”). The result is frequently a “hacking mentality” that does not withstand the test of complexity and time. I plan to strengthen the role of analytical models and formal methods. Princeton is particularly well suited for this effort due to its traditional strength in theory. Second, as part of the department’s concerted effort of vitalizing our graduate curriculum, I am in the process of designing new courses, one of which will be an advanced operating systems course that will cover the important systems developments ranging from classic literatures to state-of-art research. Another class is on advanced topics in distributed storage systems. The third issue is to consciously take advantage of “extracurricular channels” to complement our educational efforts—with the help of colleagues, I am organizing an orientation seminar series, lunch time gatherings, and research retreats that involve industry participation (a concept transplanted from Berkeley). These informal gatherings are valuable for both technical discussions and honing students’ non-technical skills. Princeton’s small size typically blurs the distinction between graduates and undergraduates so these efforts should benefit both groups. The fourth issue concerns infrastructure—I expect the resulting systems of our research to directly contribute to the departmental infrastructure. I have made progress on both the research and education fronts during my first semester and am looking forward to many more productive years spent on the pursuit of knowledge with talented students.

Contents

| | |
|--|-----------|
| A Project Summary | i |
| C Project Description | 1 |
| C.1 Introduction | 1 |
| C.1.1 Inadequacies of State-of-art | 1 |
| C.1.2 Overview and Implications of Previous Research | 1 |
| C.1.3 Outline of Proposed Research | 2 |
| C.2 Low Latency Storage Architectures | 2 |
| C.2.1 Motivations | 3 |
| C.2.2 Approaches | 4 |
| C.2.3 Related Work | 9 |
| C.3 Software Systems for Low Latency I/O | 9 |
| C.3.1 Motivations | 9 |
| C.3.2 Approaches | 10 |
| C.3.3 Related Work | 12 |
| C.4 Ubiquitous Storage | 12 |
| C.4.1 Motivations | 12 |
| C.4.2 Approaches | 13 |
| C.4.3 Related Work | 14 |
| C.5 Education Activities | 14 |
| C.5.1 Prior Activities | 14 |
| C.5.2 Planned Activities | 14 |
| C.6 Conclusion | 15 |
| C.7 Department Endorsement | 16 |
| D Bibliography | 17 |

C Project Description

C.1 Introduction

Magnetic disks play a very important role in virtually every information processing system. In 1998, approximately 150 million drives with a total capacity of 600 petabytes (a petabyte is 10^{15} bytes) and a total value of over 40 billion dollars were manufactured [41]! If the trends of the past decade continue, as they are expected to, these statistics are likely to become even far more impressive, as per byte cost drops by over 50% per year, as capacity per drive increases by about 60% per year, and as the amount of digital information explodes. It is crucial that we understand the science underlying this backbone of our information infrastructure.

C.1.1 Inadequacies of State-of-art

The technology has advanced so far that the inadequacies of the state-of-art is becoming increasingly apparent. The first of these inadequacies is the lack of understanding of the science behind the architecting of the device itself. There are many simple questions that we do not have answers for today. For example, given a certain budget, what “kind” of disks should we be making, in terms of diameter, the number of platters per drive, and the number of heads per surface? When one architects other parts of the computing system, such as the processor, one must take a quantitative approach and carefully choose an optimal mix of the number of the functional units to achieve a “balanced” system. When it comes to building a disk, however, the notion of a “balanced” disk is absent. The conventional wisdom answers these configuration questions with two words: form factors. In other words, we make disks the way we do today because that is how we have been making them! As we will see later in this proposal, conventional wisdom can be far from optimal.

The second inadequacy of the state-of-art is the lack of coordination of the various levels of hardware and software in a storage system. Systems and applications programmers have long internalized that disk accesses are necessarily long-latency operations. To avoid such operations, they typically bend themselves backwards to aggregate small operations into large ones to amortize the overhead. Thanks to the tremendous growth of disk bandwidth (at an annual rate of 60%), successive generations of disks can cater to the needs of these (contorted) codes without much improvement in latency (at an annual rate of only less than 10%). This arrangement encourages a compartmentalization: systems and applications programmers work hard to group small I/Os into large ones without demanding radical architectural changes of the devices, while device makers concentrate on making these “typical” applications run fast. This temporarily convenient compartmentalization works only to a certain a point: the latency \times bandwidth product of disks has risen more than 30 times in the past decade, and this rise demands increasingly larger I/O sizes in order for the amortization to be effective. But there is a limit on how many small I/Os that we are able to aggregate into large ones. In short, the quantitative evolution in technology is reaching a point where qualitative rethinking of the old compartmentalization is becoming necessary.

The third inadequacy of the state-of-art is that the file system designers and builders have not yet recognized the emergence of a new class of consumer storage devices as one of the network storage research issues. As disks get cheaper and smaller, and as traditionally analog information is increasingly being turned into digital representation, consumer devices of all shapes and forms are all starting to store some type of stable state. We call this type of state the “invisible bits.” Potentially as a sign of things to come, IBM recently announced a new miniature disk, dubbed the “Microdrive” (shown in Figure 1). The intended target hosts of these disks are devices such as PDAs and digital cameras. It is not far-fetched to envision a day in the near future when each of us will be surrounded by a cloud of these invisible bits. It is even conceivable that the aggregate of such ad hoc storage can subsume or even replace conventional “machine-room disks” as the dominant form of bits if decentralization is carried out to its logical next step. Today, each of these devices requires its own ad hoc manual management. It is not hard to imagine how this can become a management nightmare. Fortunately, a companion development to the emergence of these devices is the gradual progress that is being made towards ubiquitous network connectivity of various forms. We believe that time is right for us to start exploring how we can marry these two emerging technologies and apply the principles of network storage to conquer the tyranny of invisible bits.

C.1.2 Overview and Implications of Previous Research

My prior research have been in two areas: network file systems and high-performance networking. The work has motivated both the problems and the solutions addressed in this proposal. I categorize the prior work into three categories and discuss them within the context of the proposed research.

The first group of my prior work centers around the design and implementation of a network file system for a cluster of workstations that are connected by a fast switched network. The result is a prototype “serverless file system” called xFS [2, 7, 12]. The gist of the approach underlying xFS is to distribute the many responsibilities of a traditional

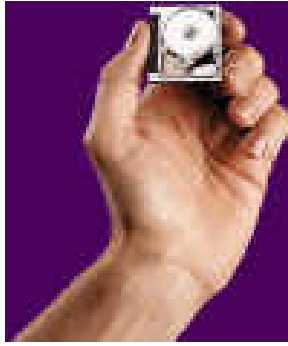


Figure 1: The “Microdrive” recently introduced by IBM. It is only slightly larger than an American quarter coin and holds 340 MB of data.

centralized file server over a number of commodity workstation. The result is better performance, scalability, reliability, and lower cost.

I have learned three main lessons from this experience that have a direct impact on the proposed research. The first lesson is the power of decentralization embodied in a “peer-to-peer” architecture that has no centralized bottleneck anywhere in any dimension. I believe that this architecture is a natural fit for managing a network of ad hoc storage devices as well. The second lesson is the power of formal methods, which is a necessity for managing the complexity of a distributed system that is as complex as xFS. I believe approaches like this will play an increasingly important role in the future. I intend to extend and emphasize this approach in planning the operating systems education. The third lesson is a main weakness of xFS, which is that it offers virtually no improvement in terms of I/O latency. This weakness not only limits the usefulness of xFS for important applications, it also makes xFS itself significantly more complex in terms of issues such as managing crash recovery. This lesson leads directly to my latest focus on low latency I/O.

The second group of my prior work concerns the roles played by network interfaces in optimizing communication latency [27, 52]. Interestingly, I have found that a number of important principles discovered while studying network interfaces should be equally applicable to I/O processors such as a RAID controller or an embedded disk processors, which are subjects of the proposed research. Examples of such synergy include a pipeline latency theory, and the principle of exploiting the intimate knowledge of the underlying media (the network or the disk mechanism) by the I/O processor. Low latency communication also provides a crucial enabling technology in the quest for low latency I/O.

The last and most recent group of my prior work is in fact also a first step of the proposed low latency I/O research. These projects examine how to perform low-latency writes using a programmable disk [51] and how to lower read latency using intelligent reorganization [32]. The low-latency write work demonstrates an order of magnitude improvement in write latency and promises the feasibility of microsecond write latency with further optimizations in the future. However, these are promising but, frankly, timid first steps—they are essentially attempting to answer the question of how to get the most out of *existing* disks. But what happens if we get to make our own disks? How do we extend these ideas to a RAID? How would we write file systems and applications differently given these new disks? These are some of the questions that I would like to answer in the proposed research.

C.1.3 Outline of Proposed Research

So far I have outlined the inadequacies of the state-of-art of storage systems and file systems and how my previous work may provide a springboard for the next steps. I now briefly outline these next steps that will address the inadequacies. The two areas of my proposed research are: 1) low latency I/O, and 2) ubiquitous storage.

To achieve low latency I/O, I will examine a number of vertical layers including 1) the architecture of the disks themselves, 2) how file systems interact with these “new” disks, and 3) how important applications, such as database applications and graphics applications, interact with the new storage systems. To manage ubiquitous storage, I will examine how to leverage the emerging ubiquitous connectivity and a peer-to-peer architecture to reduce or eliminate the unpleasant chores such as managing migration, coherence, and backup manually.

C.2 Low Latency Storage Architectures

Disk arrays can provide scalable bandwidth with increasing number of disks, but they do not address latency. In this part of the proposal, we explore how to improve secondary storage latency by systematically increasing the ratio between disk heads and usable capacity. The most direct technique is to build “faster” disks by altering disk geometry. In addition, as an alternative to building such disks, we also describe how we can emulate these faster disks using multiple conventional disks.

We have developed initial analytical models and have performed measurements in the months leading to this proposal to demonstrate the soundness of the proposed ideas. We show that the proposed techniques are governed by a common principle that the overhead-independent part of the latency improves by a factor of the square root of the amount of extra resources. Initial experiments show very promising results.

C.2.1 Motivations

Although considerable effort in disk array research [9, 38, 53] has resulted in vast bandwidth improvement, low latency remains an elusive goal. Latency impacts the performance of important applications such as persistent object stores [28] and database applications [48, 49].

The gist of our technique is systematically trading off capacity for lower latency. Several technology trends have simultaneously enabled and necessitated this approach. The key trend is the phenomenal growth of disk areal density. At an annual rate of 60% [16], this growth has resulted in a dramatic improvement in disk capacity, bandwidth, and cost.

This trend has several implications. Since disk latency has been improving at only 10% per year [31], disks are becoming increasingly unbalanced in terms of the relationship between capacity and latency. A similar phenomena is happening to the memory subsystem [40]. To address this imbalance, Wood and Hill proposed the so called *Converse Amdahl's Dictum*: each megabyte of memory should be accompanied by one MIPS of processing power [54]. In other words, we can achieve better cost/performance by increasing the processing to memory capacity ratio. We believe that a similar insight applies to secondary storage systems: we can achieve a more balanced system by increasing the disk head to capacity ratio.

The original Amdahl's Dictum stated that each MIPS of processing power should be accompanied by 1 MB of memory, but the exact ratio is open to debate. Similar questions about ratios in a storage system beckon answers. In fact, a disk has a number of parameters such as diameter of platters, rotation speed, number of heads, and number of platters. How should we configure our storage systems so that these ratios are more balanced based on the Converse Amdahl's Dictum?

Unfortunately, the most common answer from disk industry today is “form factors”—these parameters are largely a result of historical reasons. This lack of understanding has led to surprises. For example, customers of some data centers have discovered that when they purchase lower-end disks, they mysteriously achieve better performance! The reason, again, lies in the Converse Amdahl's Dictum: by purchasing lower-end disks that have less capacity per spindle, they have accidentally stumbled upon a more favorable capacity to spindle ratio. One cannot help wondering whether even “lower-end” disks can deliver even better performance. In fact, evidences suggest that as disk density increases rapidly, by blindly conforming to ad hoc form factors, the new disks are becoming increasingly unbalanced as the capacity to spindle ratio continues to climb. Time is ripe for us to systematically investigate what the balanced ratios should be.

Database vendors today have already recognized the importance of building a balanced secondary storage system. For example, in order to achieve high performance on TPCC [49], vendors configure systems based on the number of disk heads instead of capacity. To achieve $D\times$ the bandwidth, the heads form a D -way mirror, a D -way stripe, or a combination of the two (as was done in Petal [29]). What is not well understood is how we can systematically translate the excess capacity into improved latency, which can in turn lead to improved throughput, reducing the number of heads required.

So far, we have only asked questions about hardware configurations. These, of course, cannot be divorced from the software. Unfortunately, no file system today consciously takes advantage of such a balanced system even if one existed.

Another important technology trend is the relationship between memory and disks. The areal density of memory has been improving at a steady rate of 40% per year [16], a rate that is far behind that of the growth in disk density (60%), which has benefited significantly from recent innovations in disk sensing and recording technology. This disparity also manifests itself in terms of cost per megabyte: the gap between memory and disk is roughly two orders of magnitude today and growing.

A naive answer to the disk read latency problem is that it will be addressed in time by the growing memory caches. However, file system implementors are consistently surprised by the low cache hit rate despite the increasing amounts of memory [3]. We believe that the widening gap between memory and disk costs can only make this problem worse. The upside of this growing disparity is that it presents an opportunity for the development of a range of storage alternatives that can fill the gap between memory and disk in terms of cost/performance.

There is encouraging evidence that the industry is beginning to recognize the importance of low latency storage architectures. A number of recent RAID systems are designed with the intent to study the tradeoff between capacity and performance. We highlight two such systems that have inspired the proposed research, while deferring the rest of the related work to a later section.

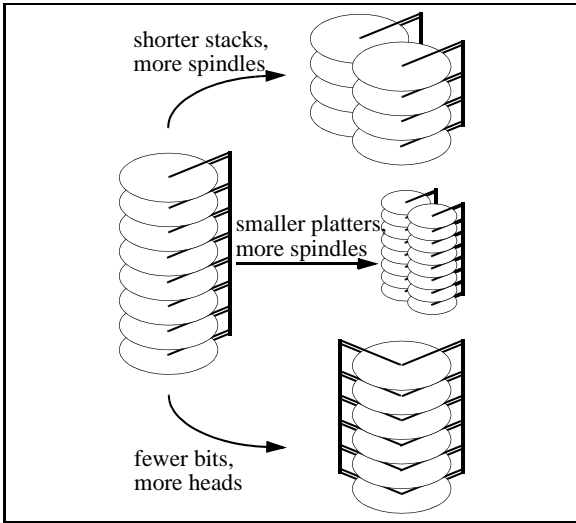


Figure 2: Options of increasing access to capacity ratio.

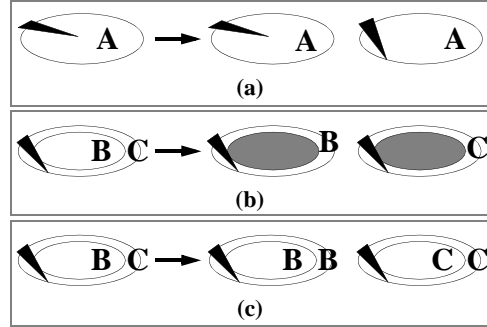


Figure 3: Techniques for reducing seek distance. Capital letters represent a portion of the data. To the left of the arrows, we show how data is (logically) stored on a single disk. To the right, we show different ways that data on the single disk can be distributed on two disks: (a) mirroring, (b) splitting, and (c) replicated splitting.

Petal is a network RAID that exports virtual disk interfaces to network clients [29]. To provide better small write performance, better load balance, and easier crash recovery, Petal chooses mirroring instead of a RAID-5 organization. The HP AutoRAID system makes the tradeoff between capacity and performance more explicit by incorporating both mirroring and RAID-5 into a two-level hierarchy [53]. The mirrored upper level provides advantages similar to those of Petal at the expense of consuming more storage, while the RAID-5 lower level is more frugal in its use of disk space.

We believe that these systems represent steps in the right direction in their explicit recognition of the tradeoff between capacity and performance. However, a large number of questions remain unanswered. The primary advantage of mirroring exploited by these systems is its low *write* latency; how can we extract low *read* latency by exploiting this tradeoff? Are there techniques other than mirroring that can accomplish this goal? Why do we have to stop at two copies? How can we build better disks that are more latency-friendly? What is the relationship amongst these different latency reduction techniques? In short, we believe Petal and AutoRAID represent two interesting design points in a vast design space that beckons a systematic exploration.

C.2.2 Approaches

There are many different ways one can balance the increased capacity with more parallel access. Figure 2 shows some possibilities:

- We can replace a tall spindle of many platters with multiple spindles of fewer platters.
- We can replace a small number of large platters with a larger number of small platters (which can also spin faster and may or may not share a single spindle).
- We can increase the head to surface ratio.

Given a particular workload mix and a fixed budget, we speculate that a balanced disk will employ a combination of these techniques to increase the degree of parallelism in the storage system. Instead of leaving it to ad hoc trials that can lead to surprises, we propose to systematically model the impact of manipulating the ratios of storage system parameters. We would like to understand, for example, how much improvement one can expect if we double the number of disks, halve the diameter of disk platters, or double the number of heads per surface. We will use simple analytical models to study how to best take advantage of the extra resources and evaluate the different alternatives. Today, a large gap exists between disk and memory in terms of cost and performance. By systematically manipulating these ratios based on the Converse Amdahl’s Dictum, we may be able to provide a spectrum of technology alternatives to fill this gap.

MimdRAID

Disks are cheap. The simplest form of introducing extra resources into the system is to add more disks. The challenge is how to effectively turn extra disks into low latency. One might conclude that we can simply leverage the RAID concept [38]. Indeed, one can view the work proposed in this section as an extension of RAID. By reading from or

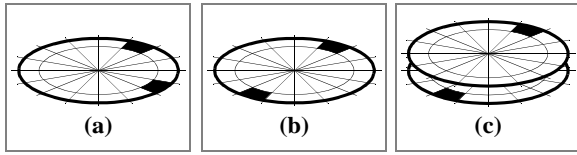


Figure 4: Techniques for reducing rotational delay. (a) Randomly placed replicas. (b) Evenly spaced replicas. (c) Replicas placed on different tracks.

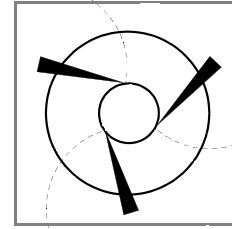


Figure 5: Increasing the number of heads per surface.

writing to a number of disks in parallel, a traditional RAID system delivers large bandwidth for large I/O's. This is analogous to a Single Instruction Multiple Data (SIMD) computing model where a number of different processing elements perform the same operation on a number of different data streams simultaneously in lock step. Although the SIMD model has large bandwidth potential, it does not help improve latency of small I/O operations. In contrast to the SIMD model, a Multiple Instruction Multiple Data (MIMD) computing model allows different processing elements to execute different instructions for different data streams. We propose to extend the traditional SIMD RAID to a MIMD RAID. In a "MimdRAID", the different disk heads not only can operate in lock step to deliver large I/O bandwidth, they can also be decoupled when necessary to perform different duties. This extra degree of flexibility allows a MimdRAID to improve small I/O latency by exploiting the parallelism in the storage system.

Here we consider how we may use multiple disks to reduce both seek and rotational delay. We will explore the other features of a MimdRAID in a later section. Figure 3 shows the three techniques of reducing seek distance. The first is traditional mirroring. Mirroring can reduce seek distance because we can choose the disk head that is closest to the target sector in terms of seek distance (shown in (a)). The second technique is *splitting* (shown in (b)). Under this technique, we stripe data across multiple disks and discard the space in the middle of the disks. As a result, the disk heads are restricted within a smaller region. The third technique is *replicated splitting* (shown in (c)). Under this technique, we replicate the data *within* each disk a number of times by taking advantage of the space discarded by simple splitting. Intuitively, replicated splitting can further reduce seek distance beyond splitting because for most of the time, there is an identical copy of the target sector on each side of the disk head so that the head can choose the closer one. One question is which one of these techniques is the best in terms of seek distance reduction.

Suppose the maximum seek distance on a single disk is S , the total amount of data in question fits on a single disk, but we are given D disks. Suppose we perform random seeks. It can be shown that the average seek distance is $S/3$ using a single disk, $S/(2D + 1)$ under mirroring, less than $S/(3D)$ under splitting, and between $S/(4D)$ and $S/(3D)$ under replicated mirroring.

Reducing seek distance alone is not sufficient. We now consider how a MimdRAID may reduce rotational delay. The general approach is to replicate data at different rotational positions. By choosing a replica that is rotationally closer than others, this approach has the effect of emulating a disk that spins faster. In the following discussions, we assume that the full rotational delay on a single disk is R and we replicate data D times. As a base case, the average rotational delay on a single disk is simply half of a full rotation: $R/2$.

A naive approach is to place replicas at random rotational positions. There are several scenarios under which this arrangement can occur. One is a mirrored system whose spindles are not synchronized. Another is when we schedule the writes of different replicas at different times at "convenient" but random rotational positions either on a single disk (shown in Figure 4(a)) or on different disks. Using D rotational replicas, we can prove that the average rotational delay to the first such replica is: $R/(D + 1)$. A better approach is *even rotational replication*. We place the D replicas $360/D$ degrees apart from each other. This can be done on a mirrored system whose spindles are synchronized or on a single disk when we carefully schedule the replication (shown in Figure 4(b)). We can prove that the average rotational delay under this approach is: $R/(2D)$. Figures 4(a) and (b) have illustrated the concept of rotational replication by making copies within the same track. Unfortunately, this decreases the bandwidth of large I/O. It may also negatively impact read latency as a higher degree of rotational replication effectively reduces the length of a track. To address this weakness, we place the replicas on different tracks, within the same cylinder (shown in Figure 4(c)), elsewhere on the same disk, or on different disks whose spindles are synchronized.

So far, we have considered techniques for reducing seek distance and rotational delay in isolation. Now we consider the combined effect. There exists a tension between reducing rotational delay and reducing seek distance. For example, if we raise the degree of replication within a cylinder to reduce rotational delay, we push data belonging to different cylinders farther apart from each other, raising seek distance. In order to improve the overall latency by a certain factor, we must at least improve each of the seek and rotational dimensions by the same factor. This leads to our first rule of thumb.

Rule 1 *By using D disks, we can improve the overhead-independent part of random read latency by a factor of \sqrt{D} .*

Among all the alternatives that we have discussed, one way to categorize them is whether the replicas are made within a disk (the *intra-disk* approach) or across different disks (the *inter-disk* approach). The intra-disk approach inherits the advantages of disk splitting compared to mirroring in terms of fewer replicas. In order to achieve a factor of \sqrt{D} improvement, the intra-disk approach requires only \sqrt{D} rotational replicas while the inter-disk case requires D replicas. Consequently, propagating replicas is less costly under the intra-disk approach. As disk drives acquire more intelligence in the future, it is possible that intra-disk propagations can be controlled by the embedded controller inside the drive without outside intervention. Such an optimization can take advantage of the “free” bandwidth between the head and the platters without consuming valuable interconnect bandwidth and RAID controller processing. Unlike this intra-drive propagation, traditional mirroring relies on inter-drive propagation, which cannot avoid crossing the interconnect between disks.

An even more important difference is the throughput of queued requests that can be reordered. Under the intra-disk approach, all replicas exist on a single drive; so scheduling is trivial: requests are simply sent to the drive that is solely responsible for the data. Each drive queues requests locally and performs scheduling locally. A good local scheduler considers both seek distance and rotational replicas to maximize throughput. Furthermore, each drive is only responsible for a subset of the data, so we can reduce the amount of head movement by avoiding having to pass over the remainder of the data. In contrast, under mirroring, any request can be scheduled for any individual drive queue so the global scheduling is complicated. Furthermore, because each disk contains all the data, a naive layout policy or scheduling algorithm will result in a greater amount of head movement than the intra-disk approach as the head has to pass over the entire data set as opposed to a subset. Traditionally, the best throughput one can hope for with a mirrored system is linear scale-up as we increase the number of disks. In contrast, because the latency of individual requests improves as we add disks to an intra-disk system, we can achieve super-linear scale-up.

Altering Disk Geometry

Much of disk drive design is based on ad hoc historic reasons such as form factors. As disk capacity rapidly increases, disks are becoming increasingly unbalanced in the relationship between capacity and latency. The techniques of using extra disks to reduce latency are in effect emulating faster disks using existing slow disks. A more direct and more cost-effective approach to balancing capacity with lower latency is to simply build “better” drives to start with so we do not have to resort to either replication or discarding space. As array manufacturers such as HP [53] and EMC [5] explore intermediate storage levels between RAID-5 and memory, there is an added incentive for us to seriously examine the implication of “faster” drives.

The first approach to building faster disks is to reduce disk diameter. This reduces both seek distance and rotational delay as we are able to spin a smaller platter faster. Due to limited data channel rates, the internal data rate is *the* constant. A corollary is that the linear velocity must be kept constant as we vary the rotational speed. In other words, a reduction of diameter can be matched by an increase of the same factor in rotational speed without exceeding the allowable data rate. This leads to our second rule of thumb:

Rule 2 *By reducing the platter capacity by a factor of C , we can improve the overhead-independent part of random read latency by a factor of \sqrt{C} .*

The second approach to building faster disks is to employ multiple heads per surface. Figure 5 shows this approach. Increasing the number of heads per surface is very much like employing extra disks. In fact, we can show that a D -way mirroring system with synchronized spindles can emulate the behavior of a disk with D heads per surface by carefully choosing the rotational positions of the replicas. We summarize this in our third rule of thumb.

Rule 3 *By placing H heads on each surface, we can improve the overhead-independent part of random read latency by a factor of \sqrt{H} .*

Cost-Effectiveness

The technique of using extra disks is the least cost-effective because replication and discarding space are both wasteful. Maintaining replicas also introduces complexity. Reducing platter diameter is less wasteful because no bits are wasted; but this technique results in an increase of both the number of heads and spindles for the same amount of capacity, so it is more costly than the third technique of just adding heads. We note that there does not exist a single “perfect” drive that has the “right” diameter and “right” number of heads per surface. Instead, there may exist one “right” drive for every cost/performance specification.

An alternative to these techniques is to simply buy more memory to address low memory cache hit rates. This strategy is clearly viable if we have enough money to buy extra memory to significantly raise the cache hit rate. However, when

| | Barracuda | Elite |
|-------------------------------|-----------|-------|
| Diameter (inch) | 3.5 | 5.25 |
| Capacity (GB) | 18 | 47 |
| Heads | 20 | 28 |
| Spindle speed (RPM) | 7,200 | 5,357 |
| Maximum seek (<i>ms</i>) | 16 | 28.2 |
| Overhead (<i>ms</i>) | 5 | 5.7 |
| Average latency (<i>ms</i>) | 13.8 | 18.8 |

Figure 6: Parameters of the Seagate ST118273W (Barracuda 18) and ST446452W (Elite 47).

| | |
|--------------|-------------|
| Barracuda 18 | \$714/disk |
| Elite 47 | \$1382/disk |
| Memory | \$1/MB |
| Disk heads | \$13/head |

Figure 7: Component costs.

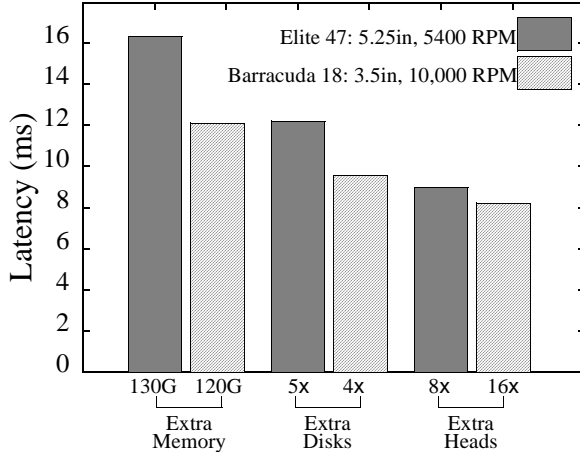


Figure 8: Cost/performance comparison of latency reduction techniques: memory caching, reducing disk diameter, using extra disks, and increasing heads per surface.

| | |
|---------------------------|-----------------------|
| Diameter | 3.5 inch |
| Capacity | 4.55 GB |
| Average sectors per track | 212 |
| Cylinders | 6962 |
| Heads | 6 |
| Spindle speed | 10,025 RPM |
| Interface | Ultra2 SCSI |
| Single track seek | 0.8 - 1.1 <i>ms</i> |
| Maximum seek | 12.2 - 13.2 <i>ms</i> |

Figure 9: Parameters of the Seagate ST34502LW (Cheetah 9LP).

operating under regimes where we cannot afford the large sum of money required to buy extra memory to raise the cache hit rate, but we can still significantly increase the number of the disk heads, our techniques are better.

This phenomenon can be explained by Amdahl’s Law, which states that the performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used. Although memory is much faster, the large cost gap between disk and memory dictates that only a small fraction of the accesses can enjoy this speedup. In contrast, when spent on extra disk heads, the same amount of money allows one to speed up a larger fraction of the accesses, achieving a larger overall performance gain. Of course, more disk heads also result in other benefits such as better write throughput.

As an example, consider the following simple hypothetical scenario. We are given \$160,000. We have one TB of data. The goal is to achieve the minimum random read latency of a single sector. We have the options of spending the money on some combination of the Seagate ST118273W (Barracuda 18) disks, ST446452W (Elite 47) disks, and DRAM. Figure 6 lists the important parameters of these disk drives. One notable feature in this table is the difference in diameter. These two types of drives are otherwise made of roughly similar technology. Figure 7 lists the component costs that we assume for our study.

Figure 8 shows the result of the cost/performance comparison. The latency reduction alternatives that we study are memory caching, reducing disk diameter at the expense of less memory caching while keeping the number of disks a constant, using extra disks at the expense of no caching, and increasing number of disk heads per surface at the expense of no caching. For this particular simple example, the performance increases in that order roughly by a factor of two while the total cost of the system remains constant.

Simulation and Prototype Measurement Results

To validate the proposed ideas, we have performed both simulations and prototype measurements. Both the simulator and the prototype are based on the Seagate ST34502 (Cheetah 9LP) disks whose key parameters are shown in Figure 9. Figure 10 shows the latency results of driving a TPCC disk I/O trace (from HP Labs [43]) through the simulator that implements the various latency reduction techniques. We see that the Square Root Rules of latency reduction are largely valid. Although the intra-disk approach has the best latency, the performance difference is small enough that its attractiveness will only become evident later. Similarly, although the techniques that alter disk geometry do not

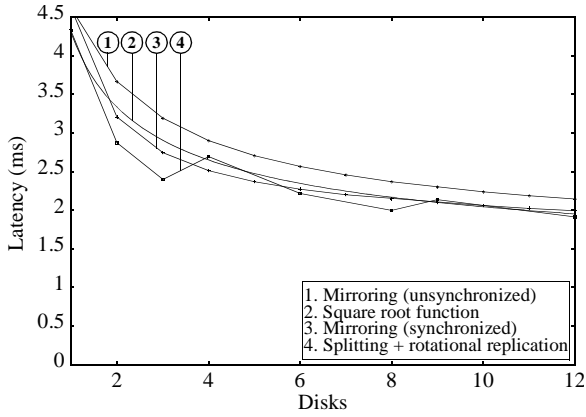


Figure 10: Simulation: latency of the transaction processing benchmark TPCC. When reducing platter size, the x-axis denotes the number of disks required to obtain the same capacity as the 4.55 GB Cheetah 9LP. When increasing heads per surface, the x-axis denotes the number of heads per surface.

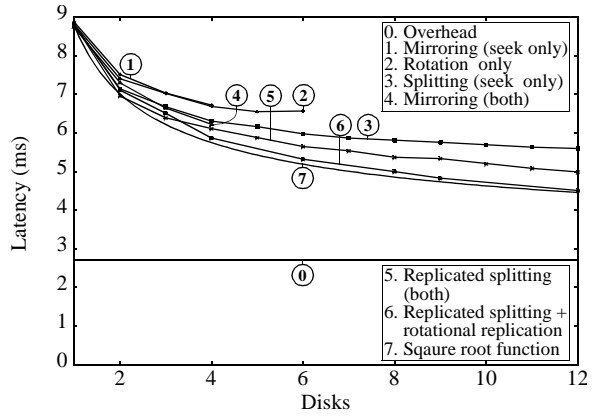


Figure 11: Prototype measurement: random read latency.

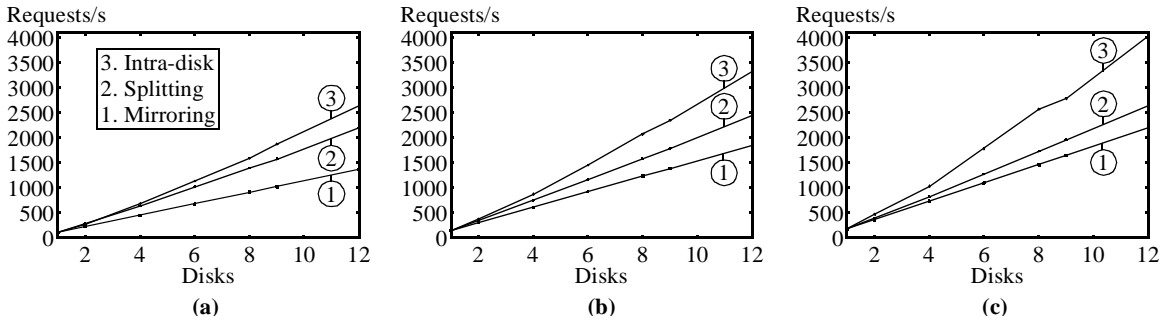


Figure 12: Throughput of random reads. The reads are queued and potentially scheduled out of order. (a) The per drive reorder queue contains 4 requests. (b) The reorder queue contains 16 requests. (c) The reorder queue contains 64 requests.

perform better than other approaches with the same number of disk heads, their main attractiveness is their simplicity and cost effectiveness as discussed previously. We have obtained simulation results from other I/O traces such as office-environment file system workloads and the results are similar to that of Figure 10. Figure 11 shows the random read latency measured on the prototype implementations. The curve labeled “6”, which plots the latency of an intra-disk MimdRAID implementation, follows closely the square root curve labeled “7”, confirming once again the Square Root Rule. The figure also shows the importance of addressing both seek and rotational delay.

So far, we have only considered latency. Now we consider throughput. Traditionally, in a well designed RAID, as we increase the number of disks, the latency experienced by each request remains constant; so the overall throughput increases linearly. In contrast, as we increase the number of disks in an intra-disk MimdRAID, the latency experienced per request decreases according to the Square Root Rule; so the overall throughput should achieve super-linear speedup. The line labeled “3” in Figure 12 shows this super-linear speedup; the intra-disk approach achieves a $21\times$ speedup with 12 disks, which is approximately 80% to 95% better than mirroring under various queuing conditions.

These preliminary results show promise. However, many questions remain unanswered. To name a few important ones here, we need to devise replica-propagation schemes using a combination of “free bandwidth” between the platters and the heads and extra resources (in the forms of extra disks, heads, and/or interconnect bandwidth) that are devoted to propagating replicas so that it does not severely interfere with user operations. We need to devise schemes that dynamically choose the candidates and degree of replication based on access patterns. We need to devise disk scheduling algorithms for replicated splitting, rotational replication, and mirrored systems. We need to refine our disk architecture models to complete our understanding of how to build an “optimal disk”. These models will include, for example, the question of the optimal inner and outer platter radius given power and data rate constraints that can deliver the best performance given a particular usage pattern. We will consider how a file system and applications can take advantage of a MimdRAID in later sections. In addition to these better defined research questions, we are also considering more radical and more speculative ideas: one apparent limiting factor in the MimdRAID systems is the latency incurred on acceleration and deceleration of heads; and we are exploring various architectural changes that can drastically reduce this latency component.

Industrial Collaborators

Our closest industrial collaborators are from the HP Laboratories’ Storage Systems Program directed by Dr. John Wilkes. Parts of our study are based on the traces collected at HPL [43]. As explained in the next section, our work is closely related. The HP AutoRAID [53] is one of the systems that inspired our research. The approaches taken by us and the HP Ivy project [30] may complement each other. And we share similar goals with the HP “attribute-managed storage” [15]. In addition, at the lower drive level, we have been mutually benefiting greatly through discussions with Drs. Honesty Young, Spencer Ng, and Ed Grochowski at IBM Almaden.

C.2.3 Related Work

In Section C.2.1, we have briefly described the HP AutoRAID [53] as an example system that systematically trades off capacity for performance. As with RAID-1 systems, however, its primary focus is solving the small write problem of RAID-5. Although mirrored systems can improve read performance by dynamically balancing the load, they have not aggressively sought low read latency. In order to achieve the low read latency goal, we have taken the mirrored approach further in a number of directions: exploring alternatives other than inter-disk mirroring, exploring alternatives that can take advantage of more than doubling the amount of resources, exploring the implication of altering disk geometry, and quantifying the difference among all these alternatives.

The HP Ivy project [30] is a simulation study of how a high degree of replication can improve read performance. Our proposal differs from Ivy in several significant ways. First, Ivy only explores reducing seek distance and leaves rotational delay unresolved. Second, Ivy only examined mirroring. The advantages of disk splitting compared to mirroring includes the elimination of the replication cost, better throughput, and simplicity. Indeed, the Ivy study shows that propagating the replicas in order to maintain a high degree of replication may result in significant increase of disk queue length, thus undoing some of the benefits of mirroring. The third difference is a feature of Ivy that we intend to incorporate into MimdRAID in the future: Ivy dynamically chooses the candidate and the degree of replication by observing access patterns. We are currently researching a wide range of access patterns that can be used to dynamically tune MimdRAID behavior. These include the read/write ratio to guide the placement of writes using different logging techniques, the relative contribution of seek and rotation to the total latency to guide which of these two dimensions should receive more resources, and the frequency and nature of accesses to guide the number and placement of replicas.

Ng examines intra-track replication as a means of reducing rotational delay [36]. We have explained that a disadvantage of this approach compared to inter-track replication is that it degrades large I/O bandwidth due to more frequent track switches.

The importance of reducing rotational delay has long been recognized. Seltzer and Jacobson independently examined a number of disk scheduling algorithms that take rotational position into consideration [24, 44]. Although these techniques are useful for the local scheduler on the individual drives, it is less clear how these techniques can be extended for replicated splitting, rotational replication, and mirrored systems. These are subjects of our current research.

The history of multi-headed disks dates back to drums (or fixed-head disks), a technology that is no longer viable due to its high cost. Multi-headed disks, however, survived long after the demise of drums. Each head is responsible for a subset of the cylinders to reduce seek distance. However, we are not aware of approaches that used multiple heads to improve rotational delay.

One of our goals of studying the impact of altering disk geometry is to understand how to configure a storage system given certain cost/performance specifications. The “attribute-managed storage” project [15] at HP shares this goal, although its focus is at the disk array level as opposed to individual drive level.

C.3 Software Systems for Low Latency I/O

C.3.1 Motivations

So far, we have only considered the architectural aspects of low latency I/O. This is important because it allows us to run legacy file systems and applications unmodified and still achieve significant performance gain. But we can do better: these legacy systems are seldom written in a way that is suitable for the low latency I/O architectures that we are developing. This mismatch is particularly apparent in light of the two to three orders of magnitude improvement on write latency that we will be able to achieve with *virtual logging* [51], the technique that I have developed as part of my thesis work. It implements a transactional log whose entries are not necessarily physically contiguous and are chosen based on the precise locational information of the disk head.

The mismatch is mostly due to the long-standing assumption that performing small I/Os is intrinsically expensive. This assumption has at least two implications. The first is that when small I/Os do occur, programmers typically do not pay much attention to software overhead (or even transfer size) because they assume that this time will be dwarfed

by the mechanical latency. The second is that programmers usually contort their data structures and algorithms in a way to favor large I/Os.

We illustrate this mismatch through several examples. The first example concerns the virtual memory (VM) system. Long ago, page sizes were chosen to be large enough (typically 4 KB or 8 KB) to amortize the high cost of disk accesses. Today, there is a trend that page sizes may increase to improve the effectiveness of TLB coverage. Many components of a modern operating system are tightly integrated with the VM system; file systems are an example. As a result of this tight integration, “small” operations more often than not translate into large page-sized operations, unnecessarily increasing I/O traffic in most cases. If we do not address this weakness, then we cannot realize the full potential of the low-latency architecture. The question that we are interested in is how to restructure a system that is not “small I/O-phobic.”

The second example is transactional applications as exemplified by the TPC benchmarks. TPCB [48] simulates debit/credit banking transactions. TPCC [49] simulates order entry activities of a wholesale supplier. Committing a transaction requires at least one synchronous disk I/O. In a traditional storage system, it is difficult to achieve latency that is faster than a half rotation delay. There are several existing techniques designed to circumvent this weakness. One is using NVRAM, which has its capacity, reliability, and cost limitations. Another technique is *group commit*: the strategy of amortizing the cost of disk writes by grouping a number of concurrent transactions into a single write. Unfortunately, group commit does not improve latency of single commits. Furthermore, the use of a redo log in a database implies that data is written twice: once to the log and once to the database itself. The throughput is limited by the log truncation performance. The question that we are interested in is how to build a new transaction system based on the low latency I/O architectures.

The third example concerns the rise of “information appliances” such as internet search engines [22], web servers [37], and file servers [21]. The appliance approach is gaining popularity because it allows one to take advantage of application-specific enhancements without getting bogged down in the goal of building an all-encompassing general-purpose system, which could be difficult to accomplish. Such application-specific information, however, seldom trickles down to the storage layer. The question that we are interested in is how the low latency I/O architecture can take advantage of the application-specific information.

In summary, I believe that the availability of low latency I/O can significantly change the landscape of all levels of software systems.

C.3.2 Approaches

In this section, I will consider four information appliance applications that are I/O intensive and can benefit greatly from low latency I/O architectures: a file server, a transactional system, a web server, and a graphics visualization engine. The specific techniques that I will explore are:

- Taking advantage of the intrinsic parallelism in a MimdRAID;
- Exposing the virtual log abstraction to the applications;
- Application-specific (or application-customized) file systems; and
- *Disk Active Messages*, application code injected dynamically into the storage system.

While variants of the last two ideas have been explored by others in different contexts, I believe that the low latency I/O architecture of Section C.2, as well as my virtual logging work, breathes new life into these ideas.

A File System for a MimdRAID

First, we expose the virtual log from the storage level to the file system level. The result is an integration of a *log structured file system* [42] and the virtual log. This system should deliver performance that is better than that is possible on either system alone.

More importantly, we need to investigate how to take advantage of the intrinsic parallelism of a MimdRAID. The issues that we plan to explore are:

- *A parallel virtual log*. By choosing a disk head that is closest to an available sector for a small write, and by committing a number of records in parallel to multiple disks, a MimdRAID can support a parallel virtual log whose performance should far surpass the single disk virtual log.
- *A parallel free space compactor*. When idle time is scarce, compacting and eager writing may compete with each other for resources and lead to performance degradation. With multiple disks, the compactor can run on some disks while others absorb normal writes without having to compete for resources with the compactor.
- *Reliability mechanisms*. The availability of low latency writes lowers the cost of mirroring and makes it more attractive than parity protection in many situations [53]. Furthermore, we will explore how this may interact with the “free bandwidth” between the heads and platters.

- *Data reorganization.* Reorganization techniques such as those presented in [32] can be extended to multiple disks so that we take into consideration not only temporal, spatial, and semantic locality, but also load balancing which maximizes the benefits of parallelism.

Although we have described these ideas in the context of multiple disks, these techniques can also be applied to non-traditional disks whose head to platter ratio is greater than one or smaller disks. The tradeoffs in this configuration will be different from the multiple disk case. For example, data written by one head is immediately available for reading by another head on the same surface in a multi-headed disk. As another example, we may need to use a mixture of different types of disks: reads go to disks with more heads or smaller disks while virtual logged writes can happen quickly on disks with fewer heads or larger ones. Given these different tradeoffs, it is important that we complement this investigation with the study of the optimal disk configuration in light of the Converse Amdahl's Dictum.

Our collaborators for this part of the research are Drs. Ed Lee and Chandu Thekkath at Compaq System Research Center (SRC). Ed and Chandu developed Petal [29], a network RAID system, and Frangipani [47], a distributed file system on top of Petal. The relationship between MimdRAID and our proposed network file system is analogous to that between Petal and Frangipani and we have a great deal to learn from each other.

A Virtual Log Based Transactional System

We believe that exporting the virtual log to the transaction system can significantly improve transaction latency. Furthermore, unlike a traditional write-ahead logging system, which employs *both* a log and an update-in-place file system, the virtual log file system consists *solely* of the log. In other words, the log *is* the file system. We will investigate a similar approach for database transactions: can we do away with the distinction between the database and the log by treating the entire virtual log as *the* database? This will eliminate the bandwidth waste consumed by log truncation. Our collaborator for this part of the research will be Dr. Avi Silberschatz from Bell Labs, a leading database researcher.

Web Servers and Application-Specific MimdRAID File Systems

Web servers are good examples of I/O intensive information appliances that possess a great deal of application-specific information that can be taken advantage of by the storage system. Some of these features are very attractive to a MimdRAID:

- *Files are mostly read and seldom written.* This implies that the overhead of propagating replicas in a MimdRAID is seldom incurred.
- *There is rich static semantic information that can be used as hints by the storage system.* Such semantic information may be obtained by parsing the HTML documents so that we can determine what files are related in what sequence. An interesting synergy between this idea and MimdRAID is that this semantic information can guide the choice of the candidates and degree of replication: files semantically related to multiple other files may be replicated.
- *There is rich dynamic semantic information that can provide similar hints.* For example, hotter files will be more heavily replicated by the MimdRAID or placed on "faster" disks.

We are exploring the idea of a *compiled file system* for a web server. A compiled file system is one that has multiple representations. So for example, we may maintain one representation for editing and "compile" this "source representation" into an optimized MimdRAID version that is optimal for retrieval. Such a customized file system also allows us to streamline the generic file system code, whose overhead is significant due to the outdated misperception that high overhead is not a concern compared to mechanical delays in the disk. We plan to collaborate closely with Professor Vivek Pai, who is the newest addition to our department and has done excellent thesis work on web servers.

Graphics Applications and Disk Active Messages

The last group of applications concern visualization engines that support graphics applications such as immersive environments [13] and isosurface rendering [11]. I plan to collaborate closely with the graphics faculty at Princeton: Professors Tom Funkhouser and Adam Finkelstein. These applications retrieve small elements from disks that are potentially noncontiguous. Secondary storage is likely to be the most vulnerable stage of the pipeline.

There are at least two reasons that argue for our approach. The first is that it is next to impossible for a naive run time system or the visualization application writer to keep up with the complexity of a modern secondary storage system. The second is that there will be simply more disks than computers in the system and the amount of aggregate internal disk resources, in terms of internal bandwidth between platters and heads and processing power near heads, are enormous. In short, disks know better and there are more of them.

Our inspiration is a programming model developed by the parallel computing community: Active Messages [50]. Under this model, each message transmitted over the network includes a pointer to a small piece of code. As the network

interface processor receives the message, it executes the code to integrate the message into the on-going computation, eliminating the need for complex and costly buffer management in conventional communication subsystems.

Similarly, we propose to associate an Active Message with each object on disk and make these Active Messages first class citizens of the storage system. These Active Messages will be executed by the disk controller. The Disk Active Message associated with each element may also be responsible for prefetching the next elements. Because these Active Messages are executed on an embedded processor that is aware of the details of the disk mechanisms and the MimdRAID layout, it can intelligently schedule the prefetches to minimize head movement. These prefetches can be aborted if the rendering engine indicates that they are not needed or if they can not meet the real time constraints. Under this approach, the visualization phase is driven by a “push” model from the data source (the disks) while augmented by hints from the data sinks (the display).

Associating an Active Message with each disk object allows us to easily support sharing of the disk by multiple users or multiple processes. Our model allows disk objects from different users/processes to be freely mixed when necessary (for example, to optimize write performance using the virtual log). During retrieval, the different Active Messages are executed in order as the disk head rotates atop the platters. This allows us to maximally utilize the disk bandwidth for both reads and writes even under multi-processing workloads.

C.3.3 Related Work

The proposed work on a MimdRAID file system and virtual log based transactional system is related to a number of existing techniques including reducing latency by writing data near the disk head [8, 14, 18, 34, 51], write-ahead logging [4, 10, 17, 45], file systems that support data location independence [8, 20, 21], and log-structured file systems [42]. The application-specific file system work is related to a large body of previous work including application control of disk layout [25] and caching algorithms [6, 39]. The main innovative thrust of the proposed approach is its taking advantage of the MimdRAID architecture.

The idea of programming a disk is not new. The IBM 360 allowed users to download *channel programs* into the I/O processors [19]. This could be used to, for example, traverse linked lists on disk. More recently, the SQL standard supports a stream based programming model via the *cursor* interface [33]. The Active Disk model [1] extends this stream based interface to allow more generality in terms of the type of operations that are allowed to execute inside the disk. Compared to these stream based approaches, the Disk Active Messages we propose are more fine grained: we associate code with individual objects instead of streams. This allows a larger degree of flexibility. For example, this flexibility can be used to efficiently handle sharing of the disk by multiple processes. Another important difference between the Disk Active Messages and other object based systems is that the Active Messages are executed inside the disk context to take advantage of the detailed states of the disk mechanism.

C.4 Ubiquitous Storage

C.4.1 Motivations

As disks become cheaper and as our society continues its analog-to-digital transformation, consumer devices of all shapes and forms are all starting to store some type of stable state. For example, digital cameras store images; phones store address books (or even yellow pages); and TVs store schedules. We call this type of state the “invisible bits.” It is not unreasonable to conjecture that these invisible bits may become the dominant form of storage in the near future, subsuming conventional disks enshrined in machine rooms, as decentralization is carried out to its logical next step.

In addition to the common theme of storing stable state, the second common theme of these ad hoc devices is that they need to communicate this state. For example, images in a camera need to be transmitted off the camera; address books (or yellow pages) need updates; and TV listings need to be refreshed periodically.

Today, each of these devices requires its own ad hoc manual management. It is not hard to imagine how this can become a management nightmare. One of the worst problems of these ad hoc approaches is lack of transparency. We require users to manage migration, hoarding, and coherence. For example, when a friend moves, the user must manually remember to track down all the devices that need updating: the phone that stores her phone number for automatic dialing, the car that stores her address for driving instructions, the printer that stores her address for printing mailing labels, etc. If the inconvenience of having to change all the clocks upon entering day light saving time is any indication, we should be able to appreciate the magnitude of this problem as the amount of information that we must manage will be far greater than the clock. In fact, there might be devices that depend on this address information in a more subtle way and the user might not even realize that they need updating. Related to the lack of transparency is other difficulties in management such as backup: it is unreasonable to expect the user to perform manual backup for his telephone, TV, camera, etc.

If we consider the combination of these two common themes, storage *and* communication, we are looking at the classic

“job description” of a network file system. As the amount of invisible bits is expected to explode, and as the companion progress made towards ubiquitous connectivity accelerates, we believe that time is right for us to start exploring how we can marry these two emerging developments. Our goal is to develop a form of “ubiquitous ad hoc network storage” to conquer the chaos of invisible bits.

Unfortunately, the state-of-art has not yet recognized the emergence of this new class of consumer storage devices as one of the network storage research issues. Furthermore, existing network file systems are ill-equipped to address the needs of such mobile devices. Interestingly, the state-of-art of file systems closely mirrors the state-of-art of mobile networking. For example, file systems such as Coda [26] are very much analogous to mobile IP [23]. They operate in two regimes: connected or disconnected. In the connected state, both rely on a centralized entity in the stationary network to act as a mediation agent. Disconnecting from the tethered world in both systems is viewed as a failure mode of operation. In the disconnected state, mobile devices become isolated entities. Our interest is in devising a scheme that does not depend on a centralized entity in the tethered world; in other words, The relationship between our proposed system and Coda is analogous to that between “ad hoc networking” [35] and mobile IP. We believe that there is considerable synergy between the goals and approaches of ad hoc networking and ad hoc storage systems. Without a storage system solution, however, we might force each application to reinvent the wheel. It is our conjecture that a common storage solution can significantly simplify the application efforts.

C.4.2 Approaches

We propose a solution that consists of two levels. At the lower level, we propose storage elements that we call “mobile network disk caches.” At the higher level, we propose an “ad hoc file system” that is a peer-to-peer system built on top of the lower level.

The low level mobile network disk caches are “intelligent disks” (disks with an embedded processor, memory, and a network) that provide transparency, mobility, and support for heterogeneity. To the host, the mobile cache appears as a regular disk, whose contents, however, may appear to belong to a remote agent. When idle, these caches hoard data that is likely to be accessed and flushes data that is not likely to be needed. When being used, they emulate the remote agent by operating out of the cache and communicate remotely only upon cache misses or when the space is exhausted. We envision these network disks to be small devices that can be easily slipped into a shirt pocket and can be plugged into any devices that need storage. Indeed, some of the important applications that can benefit from this type of common storage are not traditional computation applications: we can use these devices to cache music CDs, movie clips, images, phone books, or any form of digital information. In a more traditional role, the disk cache can be plugged into the workstation at any time to transparently hoard data using a high bandwidth connection. In the evening, it can be taken home to emulate the office file server disk using the hoard and a low bandwidth connection. Unlike disconnected operations, hoarding in this case is a mere optimization, not a necessity.

There are many challenging research questions. They include:

- How does the mobile network disk emulate a real disk interface?
- How do we perform transparent hoarding at the disk level?
- What disk layout do we use for the cache?
- How do we manage the cache?
- How do we use different levels of compression to tradeoff CPU time for more efficient use of network bandwidth and disk space?
- How do we manage power by, for example, intelligently choosing spindle speeds, disk geometry, and occasionally spinning down the disk?

At a higher level, the challenge is to provide shared file service in an ad hoc network. The approach of relying on a shared file server in the stationary network is clearly not sufficient. What we need is to extend the decentralization principle underlying ad hoc networking to other parts of the operating system, which includes the file system. In such a system, the “hoard” may become the primary storage area. In the past, we have had much success developing a “serverless” network file system for a network of workstations. Recall that a serverless file system is one that takes advantage of the aggregate resources in the network, including disks, memory, and CPU without any centralized bottleneck. We believe that this peer-to-peer architecture is a natural fit for an ad hoc network file system. However, it would take much more than a simple porting to successfully realize this idea in an ad hoc network. Many assumptions need to be reexamined. Some examples are:

- the difference in network performance will require a redesign of the file system protocol;
- the frequency at which hosts can join or leave the file system will require a redesign of the dynamic reconfiguration strategies; and
- the different reliability assumptions will require us to redesign redundancy schemes.

To fully exploit the synergy between the proposed ubiquitous storage project and the emerging field of ad hoc networking,

we are collaborating closely with Prof. Larry Peterson at Princeton, a leading networking researcher.

C.4.3 Related Work

Coda allows file system clients to continue operation using their local disk caches while disconnected [26]. The local disk cache emulates a central file server during disconnection and sends the changes back when reconnected. The server always contains the so called “truth.” One question that we would like to answer with the proposed research is whether it is possible to keep the “truth” locally and eliminate centralized entities which present performance, scalability, and reliability bottlenecks. Another goal of the mobile network disk is to export a storage-level interface that is simpler and more portable than that of a heavy-weight file system.

Bayou supports replicated databases and its focus is application-specific mechanisms for managing consistency [46]. One of our interests is in eliminating the concept of servers and building a pure peer-to-peer architecture, a natural continuation of our prior work on xFS [2, 7, 12]. We would also like to investigate consistency issues at a storage-level interface.

C.5 Education Activities

C.5.1 Prior Activities

During my first semester at Princeton, I taught a graduate class entitled “Advanced Topics in Distributed Storage Systems.” The class also admitted three advanced undergraduate students. The goals of this class were the following:

- *Learn about file systems.* We discussed a spectrum of cases ranging from classic literatures that were influential in the history of file system development to contemporary designs that are representative of the state-of-art research.
- *Gain a better understanding of building distributed systems in general* as the principles involved in building a high performance, scalable, and reliable distributed file system are applicable in a larger context.
- *Explore the interaction with the rest of the operating system.* File systems are not only the most visible parts of operating systems, they also interface with many other components. We examined the impact of the file system demands on the rest of the operating system.
- *Enlist student help* in new I/O research and steering future topics of research.

The topics that we discussed included disk modeling and technology trends, local file systems, file system interfaces to disk, RAID, communications, client/server file systems, file system measurements, caching, cluster file systems, disconnected operations, and database systems. The class was well received. One byproduct of the class is a web site that contains a large collection of my lecture slides which is widely visited from outside. One of the class projects was of publishable quality at a major conference and is currently under review. Partly due to this positive experience, I plan to continue to refine and update the materials for future offerings. Another positive outcome of the class was extensive interaction with graduate students. I am currently advising one student and interact frequently with many other faculties’ students.

In addition to advising graduate students, I served as the co-reader for three undergraduate thesis. Interestingly, these projects have opened doors to my latest research interest in ad hoc networking.

C.5.2 Planned Activities

My planned education activities specifically target the perceived weaknesses and strengths of state-of-art. One perceived weakness of operating systems education (from my own experience) is not enough emphasis on formal approaches. The second weakness is a lack of a coherent graduate operating systems curriculum at Princeton. The third concerns the strengths of Princeton: small class size, blurring of boundary between graduate students and advanced undergraduates, strength in theory, and proximity to prominent industrial partners.

Formal Approaches

There are two trends in systems design and implementation that are alarming. One is the increasing complexity, which is exacerbated by the fact that distributed computing fueled by the internet is becoming main-stream. The second is that rapid technology improvements constantly invalidate existing assumptions and approaches. From my own experience as a graduate student, I believe that there has been a lot of emphasis on empirical approaches but not enough on formal approaches. The result is frequently our tendency of replacing one set of ad hoc rules of thumb with a new set, which rarely pass the test of technology and complexity evolutions. In my latest work in graduate school, I consciously shifted towards a more formal approach. The result is a promise of deeper understanding, more robust performance, and more robust correctness. It is also more intellectually satisfying as we do “the science” along with engineering. To reflect this appreciation of the formal approach in the education process, I plan to take the following specific approaches:

- Include performance analysis material and literature on formal methods in the advanced operating system. The performance analysis material will be especially valuable since we do not have a separate class on this topic at Princeton.
- Analyze case studies that are success stories, failures, or controversies of systems projects as results of paying attention to or lack thereof formal approaches.
- As described in Section C.2, my planned research has a significant analytical modeling component, and will be used as examples and class projects.
- Leverage the strength of the theory group by exercising formulating systems problems in abstract terms and collaborating with the theory group in solving them.

Developing New Courses

Currently, Princeton has one “distributed systems” class that is for both undergraduates and graduates. It is a grab-bag class that includes materials on networking, security, and other distributed system issues. There is no advanced graduate class that surveys seminal literatures in operating systems. This is an unfortunate weakness as most graduate students are never exposed to some of the most important historic and state-of-art works. As Princeton is aggressively building up its systems faculty, plans are currently underway to offer a more complete systems curriculum. My interest is in the advanced operating systems class, which will be largely modeled upon the similar class at Berkeley, but with a heavier emphasis on performance modeling and formal methods as discussed above. I will also continue to refine the distributed storage system class.

Informal Activities

At the end of my graduate school experience, it became apparent that a successful career demands skills and insights that are not taught in formal class settings. With the help of colleagues, we are organizing the following informal activities:

- *An orientation series for incoming graduate students.* This will not only introduce students to the research activities in the department, it will also touch upon “meta-issues” such as why/how to widen one’s horizon in seemingly “unrelated” fields.
- *Research retreats.* This is a gathering by faculty, students, and industrial partners where people present and get feedback. In addition to their obvious technical value, these gatherings at Berkeley have proved to be extremely useful in honing the various non-technical skills that a successful graduate student should possess.
- *“Systems lunches.”* I have been responsible for organizing the weekly gathering of “systems” faculty and students. Activities at these lunches include presentations of ongoing research, paper reading and discussion, presentations by invited speakers, and reviews of submitted publications.

Contribution to Infrastructure

We are committed to eventually deploying the low latency storage systems (discussed in Section C.2 and C.3) and ubiquitous storage systems (discussed in Section C.4) in production use in the department.

C.6 Conclusion

My research agenda centers around the development of low latency I/O architectures and ubiquitous storage. My education plan centers around teaching operating systems. The intersections of these two parts include 1) the focus on analytical models and formal methods, 2) improving our understanding of how to build a high-performance, scalable, and reliable distributed system, and 3) deploying the fruits of the research so that they can contribute to the education infrastructure. I have made initial progress on both fronts during the first semester in terms of developing models and performing prototype measurements to validate the research ideas, and developing new courses. At a high level, this first semester, though a busy one, has conclusively confirmed to me the most important advantages of academia: freedom in pursuing one’s ideas and interacting with talented students.

C.7 Department Endorsement

Randy Wang joined this faculty on February 1, 1999 as a tenure track assistant professor. He joined us from the University of California at Berkeley where he had been a graduate student. His current contract runs through June 30, 2002. Our expectation, based on his performance to date, is that he will be renewed for a second 3 year term as an assistant professor in this department. It is impossible to project beyond that at this time.

Although he has been here for a short time, Randy has already made his mark on this department. He has started and has made important progress on the research on low latency I/O. This is clearly an important problem since disk usage is projected to grow at faster than Moore's Law rates for the next decade while our ability to build faster disks will grow at a slower rate.

Randy has already injected himself into the systems research activities of the department. he has established collaborations with Kai Li on low latency I/O, Kai Li, Adam Finkelstein, and Tom Funkhouser on building a scalable visualization engine, Larry Peterson on ad hoc networking and storage. He has done so while keeping his own research program going. He has achieved a remarkable amount in this short time.

Randy is supervising the dissertation research of one graduate student and is involved as co-advisor of others. He has also become an active participant in supervising undergraduate research. Students working under him for only the semester that he's been here are already producing drafts of documents that will ultimately become published papers.

Randy has shown a strong commitment to our graduate teaching programs. The lack of a strong curriculum of graduate level courses in systems has been a problem in this department for the past few years. Randy jumped in and has been organizing such a new curriculum. This effort involves creating a curriculum and selling it to faculty members many years his senior. I am quite impressed by his ability to do so. In addition to these efforts, Randy will be teaching our beginning course (to over 150 students) this coming semester.

This university is committed to supporting Randy's research. The department and engineering school provided him with startup funds of \$100K when he arrived. He has become involved in our NSF Institutional Infrastructure Grant and our Intel 2000 Grant. Through these, he has gained the equipment necessary for his lab as well as making contacts who will help him find equipment for future research projects.

I have read and endorse this Career Development Plan.

Sincerely

David Dobkin
Professor and Head
July 16, 1999

D Bibliography

- [1] ACHARYA, A., UYSAL, M., AND SALTZ, J. Active Disks: Programming Model, Algorithms and Evaluation. In *Proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VIII)* (October 1998).
- [2] ANDERSON, T., DAHLIN, M., NEEFE, J., PATTERSON, D., ROSELLI, D., AND WANG, R. Serverless Network File Systems. *ACM Transactions on Computer Systems* 14, 1 (Feb. 1996), 41–79.
- [3] BAKER, M., HARTMAN, J., KUPFER, M., SHIRRIFF, K., AND OUSTERHOUT, J. Measurements of a Distributed File System. In *Proc. of the 13th Symposium on Operating Systems Principles* (Oct. 1991), pp. 198–212.
- [4] BIRRELL, A., HISGEN, A., JERIAN, C., MANN, T., AND SWART, G. The Echo Distributed File System. Technical Report 111, Digital Equipment Corp. Systems Research Center, Sept. 1993.
- [5] BRANT, W. A., AND NIELSON, M. E. Disk based cache interfacing system and method. U.S. Patent 5805787 issued to EMC Corporation, March 1998.
- [6] CAO, P., AND FELTEN, E. W. Implementation and Performance of Application-Controlled File Caching. In *Proc. of the First Symposium on Operating Systems Design and Implementation* (November 1994), pp. 165–177.
- [7] CHANDRA, S., DAHLIN, M., RICHARDS, B., WANG, R. Y., ANDERSON, T. E., AND LARUS, J. Experience with a Language for Writing Coherence Protocols. In *Proc. of the 1997 Usenix Domain Specific Language Conference* (October 1997).
- [8] CHAO, C., ENGLISH, R., JACOBSON, D., STEPANOV, A., AND WILKES, J. Mime: a High Performance Parallel Storage Device with Strong Recovery Guarantees. Tech. Rep. HPL-CSP-92-9 rev 1, Hewlett-Packard Company, Palo Alto, CA, March 1992.
- [9] CHEN, P., LEE, E., GIBSON, G., KATZ, R., AND PATTERSON, D. RAID: High-Performance, Reliable Secondary Storage. *ACM Computing Surveys* 26, 2 (June 1994), 145–188.
- [10] CHUTANI, S., ANDERSON, O., KAZAR, M., LEVERETT, B., MASON, W., AND SIEDBOTHAM, R. The Episode File System. In *Proc. of the 1992 Winter USENIX* (January 1992), pp. 43–60.
- [11] CIGNONI, P., MARINO, P., MONTANI, C., PUPPO, E., AND SCOPIGNO, R. Speeding Up Isosurface Extraction using Interval Trees. *IEEE Transaction on Visualization and Computer Graphics* 3, 2 (June 1997), 158–170.
- [12] DAHLIN, M., WANG, R., ANDERSON, T., AND PATTERSON, D. Cooperative Caching: Using Remote Client Memory to Improve File System Performance. In *Proc. of the First Symposium on Operating Systems Design and Implementation* (November 1994), pp. 267–280.
- [13] FUNKHOUSER, T. A., TELLER, S. J., SEQUIN, C. H., AND KHORRAMABADI, D. The UC Berkeley System for Interactive Visualization of Large Architectural Models. *Presence* 5, 1 (1996).
- [14] GAWLICK, D., GRAY, J., LIMURA, W., AND OBERMARCK, R. Method and Apparatus for Logging Journal Data Using a Log Write Ahead Data Set. U.S. Patent 4507751 issued to IBM, March 1985.
- [15] GOLDING, R., SHRIVER, E., SULLIVAN, T., AND WILKES, J. Attribute-managed Storage. In *Workshop on Modeling and Specification of I/O* (October 1995).
- [16] GROWCHOWSKI, E. Emerging Trends in Data Storage on Magnetic Hard Disk Drives. In *Datatech* (September 1998), ICG Publishing, pp. 11–16.
- [17] HAGMANN, R. Reimplementing the Cedar File System Using Logging and Group Commit. In *Proc. of the 11th ACM Symposium on Operating Systems Principles* (October 1987), pp. 155–162.
- [18] HAGMANN, R. Low Latency Logging. Tech. Rep. CSL-91-1, Xerox Corporation, Palo Alto, CA, February 1991.
- [19] HENNESSY, J. L., AND PATTERSON, D. A. *Computer Architecture A Quantitative Approach*, second edition ed. Morgan Kaufmann Publishers, Inc., 1996.
- [20] HITZ, D., LAU, J., AND MALCOLM, M. File System Design for an NFS File Server Appliance. In *Proc. of the 1994 Winter USENIX* (January 1994).
- [21] HITZ, D., LAU, J., AND MALCOLM, M. File System Design for an NFS File Server Appliance. Tech. Rep. 3002, Network Appliance, March 1995.
- [22] INKTOMI CORPORATION. *The Inktomi Technology Behind HotBot*, May 1996. <http://www.inktom.com/whitepap.html/>.
- [23] IOANNIDIS, J., DUCHAMP, D., AND JR., G. M. Ip-based protocols for mobile internetworking. In *Proceedings of ACM SIGCOMM '91* (September 1991), pp. 235–245.
- [24] JACOBSON, D. M., AND WILKES, J. Disk Scheduling Algorithms Based on Rotational Position. Tech. Rep. HPL-CSP-91-7rev1, Hewlett-Packard Company, Palo Alto, CA, February 1991.
- [25] KAASHOEK, M. F., ENGLER, D. R., GANGER, G. R., BRICENO, H. M., HUNT, R., MAZIERES, D., PINCKNEY, T., GRIMM, R., JANNOTTI, J., AND MACKENZIE, K. Application Performance and Flexibility on Exokernel Systems. In *Proc. of the 16th ACM Symposium on Operating Systems Principles* (October 1997), pp. 52–65.
- [26] KISTLER, J., AND SATYANARAYANAN, M. Disconnected Operation in the Coda File System. *ACM Transactions on Computer Systems* 10, 1 (Feb. 1992), 3–25.

- [27] KRISHNAMURTHY, A., SCHAUSER, K. E., SCHEIMAN, C. J., WANG, R. Y., CULLER, D. E., AND YELICK, K. Evaluation of Architectural Support for Global Address-Based Communication in Large-Scale Parallel Machines. In *Proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VII)* (Oct. 1996), pp. 37–48.
- [28] LAMB, C., LANDIS, G., ORENSTEIN, J., AND WEINREB, D. The ObjectStore Database System. *Communications of the ACM* 34, 10 (October 1991), 50–63.
- [29] LEE, E. K., AND THEKKATH, C. E. Petal: Distributed Virtual Disks. In *Seventh International Conference on Architectural Support for Programming Languages and Operating Systems* (October 1996), pp. 84–92.
- [30] LO, S.-L. Ivy: A Study on Replicating Data for Performance Improvement. Tech. Rep. HPL-CSP-90-48, Hewlett-Packard Company, Palo Alto, CA, December 1990.
- [31] MASHEY, J. R. Big Data and the Next Wave of InfraStress. Computer Science Division Seminar, University of California, Berkeley, October 1997.
- [32] MATTHEWS, J. N., ROSELLI, D. S., COSTELLO, A. M., WANG, R. Y., AND ANDERSON, T. E. Improving the Performance of Log-Structured File Systems with Adaptive Methods. In *Proc. of the 16th ACM Symposium on Operating Systems Principles* (October 1997), pp. 238–251.
- [33] MELTON, J., AND SIMON, A. *Understanding the New SQL: A Complete Guide*. Morgan Kaufmann Publishers, Inc., 1993.
- [34] MENON, J., ROCHE, J., AND KASSON, J. Floating parity and data disk arrays. *Journal of Parallel and Distributed Computing* 17, 1 and 2 (January/February 1993), 129–139.
- [35] NATIONAL SCIENCE FOUNDATION. Research Priorities in Wireless and Mobile Communications and Networking. <http://www.cise.nsf.gov/anir/ww.html>, 1997.
- [36] NG, S. W. Improving disk performance via latency reduction. *IEEE Transactions on Computers* 40, 1 (January 1991), 22–30.
- [37] PAI, V. S., DRUSCHEL, P., AND ZWAENPOEL, W. IO-Lite: A Unified I/O Buffering and Caching System. In *Proc. of the Third Symposium on Operating Systems Design and Implementation* (February 1999).
- [38] PATTERSON, D., GIBSON, G., AND KATZ, R. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *International Conference on Management of Data* (June 1988), pp. 109–116.
- [39] PATTERSON, R. H., GIBSON, G. A., GINTING, E., STODOLSKY, D., AND ZELENKA, J. Informed Prefetching and Caching. In *Proceedings of the ACM Fifteenth Symposium on Operating Systems Principles* (December 1995).
- [40] PERL, S. E., AND SITES, R. L. Studies of Windows NT Performance Using Dynamic Execution Traces. In *Proc. of the Second Symposium on Operating Systems Design and Implementation* (October 1996), pp. 169–184.
- [41] PORTER, J. N. The Disk Drive Industry: Balancing the Demands of Technology and Markets. In *DiskCon'98* (April 1998).
- [42] ROSENBLUM, M., AND OUSTERHOUT, J. The Design and Implementation of a Log-Structured File System. In *Proc. of the 13th Symposium on Operating Systems Principles* (Oct. 1991), pp. 1–15.
- [43] RUEMLER, C., AND WILKES, J. UNIX Disk Access Patterns. In *Proc. of the Winter 1993 USENIX* (Jan. 1993), pp. 405–420.
- [44] SELTZER, M., CHEN, P., AND OUSTERHOUT, J. Disk Scheduling Revisited. In *Proc. of the 1990 Winter USENIX* (Washington, D.C., Jan. 1990), Usenix Association, pp. 313–323.
- [45] SWEENEY, A., DOUCETTE, D., HU, W., ANDERSON, C., NISHIMOTO, M., AND PECK, G. Scalability in the XFS File System. In *Proc. of the 1996 Winter USENIX* (January 1996), pp. 1–14.
- [46] TERRY, D. B., THEIMER, M. M., PETERSON, K., DEMERS, A. J., SPREITZER, M. J., AND HAUASER, C. H. Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System. In *Proc. of the 15th ACM Symposium on Operating Systems Principles* (December 1995), pp. 172–183.
- [47] THEKKATH, C. A., MANN, T., AND LEE, E. K. Frangipani: A Scalable Distributed File System. In *Proceedings of the ACM Sixteenth Symposium on Operating Systems Principles* (Oct. 1997).
- [48] TRANSACTION PROCESSING PERFORMANCE COUNCIL. *TPC Benchmark B Standard Specification*. Waterside Associates, Fremont, CA, Aug. 1990.
- [49] TRANSACTION PROCESSING PERFORMANCE COUNCIL. *TPC Benchmark C Standard Specification*. Waterside Associates, Fremont, CA, August 1996.
- [50] VON EICKEN, T., CULLER, D., GOLDSTEIN, S., AND SCHAUSER, K. E. Active Messages: A Mechanism for Integrated Communication and Computation. In *Proceedings of the Fifth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-V)* (May 1992), pp. 256–266.
- [51] WANG, R. Y., ANDERSON, T. E., AND PATTERSON, D. A. Virtual Log Based File Systems for a Programmable Disk. In *Proc. of the Third Symposium on Operating Systems Design and Implementation* (February 1999).
- [52] WANG, R. Y., KRISHNAMURTHY, A., MARTIN, R. P., ANDERSON, T. E., AND CULLER, D. E. Modeling Communication Pipeline Latency. In *Proc. of the 1998 SIGMETRICS* (June 1998).
- [53] WILKES, J., GOLDING, R., STAELIN, C., AND SULLIVAN, T. The HP AutoRAID Hierarchical Storage System. In *Proc. of the 15th ACM Symposium on Operating Systems Principles* (December 1995), pp. 96–108.
- [54] WOOD, D. A., AND HILL, M. D. Cost-effective Parallel Computing. *IEEE Computer* 28, 2 (February 1995), 69–72.