The Design of Large-Scale, Do-It-Yourself RAIDs

Satoshi Asami, Nisha Talagala, Thomas Anderson, Ken Lutz, and David Patterson
387 Soda Hall
Computer Science Division
University of California
Berkeley CA 94720-1776
November 10, 1995
Draft 1.0
patterson@cs.berkeley.edu; 510-642-6587; FAX 510-642-5775

Editor's Note: This draft of the paper has been submitted to a conference. As we will revise the paper, please beware this is a working document and the information content will change. Given that you have requested a copy of this draft, we would be delighted to get your reactions and comments on how to improve the paper. (In fact, by asking for his early draft we hope you feel an obligation to send comments!) Please send them via email, fax, or US mail via the address above.

Abstract
In this paper we explore the design of "Do-It-Yourself" RAIDs: RAID systems that can assembled by the end user from commercially available disks, enclosures, cables, racks, computers, and networks. We quantitatively evaluate the tradeoffs in cost, performance, and reliability of these DIY-RAID systems. Our principal result is an architecture that scales from 10s to 1000s of disks; we demonstrate that a 1995 implementation would have much lower cost, better and more scalable performance, and roughly the same reliability as commercially available hardware RAID systems. Furthermore, if current trends continue, these DIY-RAIDs will replace near-line tape libraries within a few years.

1. Introduction

Recently, a number of applications have emerged that require active use of very large storage systems. Examples include multimedia digital libraries, video on demand, and enterprise-wide decision support. To address this opportunity, a number of researchers are designing systems software to treat a large collection of disks on a network as a Redundant Array of Inexpensive Disks (RAID) [8, 20, 31, 33,22, 9,27, 26]. In contrast to conventional hardware controller-based RAID solutions, these "Do-It-Yourself" RAIDs use only off-the-shelf components: disks, CPUs, switched LANs, enclosures, and racks. High availability is provided by redundant storage, and high performance comes from having many disks.

In this paper we present an architecture for "Do-It-Yourself" RAIDs (DIY-RAIDs) that can scale from tens to tens to thousands of disks. We then evaluate the tradeoffs in implementations of this architecture in terms of cost, bandwidth, capacity, availability, power and footprint. We identify key technical issues that would significantly improve cost/capacity, cost/performance and cost/reliability, and we show that several recently proposed technologies for connecting disks yield surprisingly little benefit in this environment. Finally, we quantify the cost-performance advantages of these DIY-RAIDs over hardware RAIDs and near-line tape libraries.

Our approach relies on 1) personal computers (PCs) to connect disks to a switched local area networks

(LAN); 2) Switched LANs to match the available bandwidth of client computers; 3) Letting users purchase disks directly to reduce lag time between new technology and its use; 4) Constructing a reusable infrastructure that can service multiple disk lifetimes so as to amortize the cost of the infrastructure; and 5) Taking advantage of the read mostly nature of large storage applications to use large RAID groups to maintain reliability while keeping costs low via software RAID and large parity groups.

One key technology trend we are exploiting is that disk cost/capacity is dropping, at almost a factor of two per year; put another way, by 7% per month. This rate puts a heavy penalty on any lag time between when a disk is manufactured and when it can be included in a production system. DIY-RAIDs avoid the lag time by allowing “just-in-time” assembly by the user. Other key technologies include fast and inexpensive computers, expandable buses, switched LANs, and RAID software. Moreover, all these technologies are available off-the-shelf.

In contrast, by the time hardware-controller based RAIDs can be designed, engineered, manufactured, qualified, tested, marketed, and purchased, their disks are typically a generation behind. Hardware RAID systems are further crippled by their need for custom, low volume manufactured parts, which drive up their costs relative to using high volume commodity building blocks connected by switched networks.

This paper also evaluates Serial Storage Architecture (SSA)[1] and FibreChannel Arbitrated Loop (FC-AL) [2], two new proposed successors to Small Computer System Interconnect (SCSI). Even given optimistic assumptions about the cost/performance of these two standards, traditional SCSI is superior for DIY-RAIDs.

Remarkably, the argument for DIY-RAID versus hardware RAIDs is nearly identical to those made for clusters or Networks of Workstations (NOW) versus MPPs [18, 4]. The NOW argument is based on just-in-time assembly (to reduce lag time for incorporating rapidly improving microprocessors), the high cost of low volume manufacturing of MPPs, and the emergence of switched LANs. A major tenet of both studies is improving *cost* as well as performance [14, 34], so to make the case we must determine realistic costs to see if the hypothesis holds. If DIY-RAIDs and NOWs are successful, they may be the examples of a new branch of computer architecture that explores how to assemble high volume components into interesting systems, joining the traditional study of the design of those high volume components.

Using commercial products available today, implementations of the DIY-RAID architecture result in an incrementally expandable disk system with a cost/megabyte within 25% above that of a single internal PC disk, with greater reliability than a single disk, and as reliable as commercially available hardware RAIDs. In addition, these DIY-RAIDs can scale to hundreds of terabytes, with nearly constant dollars/megabyte and linear improvement in system bandwidth. At a 1995 cost of about $300,000 per terabyte, and with costs expected to almost halve each year, DIY-RAIDs call into question the viability of near-line tape libraries.

The rest of the paper talks about these issues in detail.

2. Components of a Do-It-Yourself RAID

This section lists the important trends of the disks, interfaces, networks, computers, cabinets, and software from the perspective of building a DIY-RAID.

RAID

The original RAID work gives a scheme for using multiple disks to provide fast and highly-available disk storage [10]. A stripe of data is divided into G–1 data blocks and one parity block, calculated as the exclusive-OR of all data in the associated data blocks. The size of a *stripe* or *parity group* is then G blocks. Each block is stored on a separate disk, with the parity blocks providing enough redundancy to recreate missing data in case of a single disk failure. Large transfers, or large number of concurrent small transfers, can use multiple disks for better throughput. Using the nomenclature of the first RAID paper, this organization is called a RAID level 5. For even greater protection, there is RAID level 6, or P+Q RAID. By having two redundant data blocks, one being parity and the other a slightly more complex calculation than simple parity, a parity group can recover from two disk failures.

One enhancement on either RAID 5 or RAID 6 is “Orthogonal RAID”: parity groups are arranged to be orthogonal to the strings, power supplies, and enclosures. Such an organization ensures that if any of these units fail they disable at most one disk per parity group, which a RAID scheme can recover from. Thus the RAID ideas can mask environmental failures as well as disk failures. A second enhancement is “hot standby” spare disks, used to reduce repair time. After failure, the system immediately reconstructs the missing data onto the spare rather than waiting for the end user to replace the failed disk. Hot spares thus reduce the time the system is vulnerable to another failure. These standby disks need not be empty; spreading the equivalent space among all disks means the extra disks can help performance [17].

RAIDs have two drawbacks. If the system does not simultaneously write all G–1 data blocks at once, then each write turns into four disk accesses: read the old data and old parity to calculate new parity and then write the new data and new parity. Several techniques exist for batching writes in files systems to eliminate this overhead [28, 19, 20, 31, 32]. The second drawback is cost: generally the cost per megabyte of a hardware RAID is 2.5 to 10 times that of a single personal computer disk[25].

Disks

At the head of the list of DIY-RAID components are disks; their cost and performance drive the rest of the system. The fundamental technical measure of disks is areal density, which is the product of tracks-per-inch and bits-per-inch (BPI). Areal density improved by 27% per year until about 1990, when the switch to magneto-resistive heads accelerated the increase to 60% per year. One of he state-of-the-art disks in late 1995 is the IBM UltraStar XP, with an areal density of 544 Million bits per square inch.

One consequence of this increased density has been increased performance: the disk must be able to transfer the BPI times the rotation speed, so increased BPI translates directly into greater bandwidth. The 3.5 inch UltraStar XP disk spins at 7200 RPM and delivers 9 to 12 MB/sec, depending on where the track is located on the platter. Between occasional increases in rotation speed and regular increases in BPI, we can expect disk bandwidth to increase by at least 20% per year, which must be accounted for in disk interfaces.

Another consequence of faster improvement in areal density is improved cost/megabyte. Conventional wisdom has been that improvements in cost exactly match improvements in areal density. Figure 1 disproves conventional wisdom: between 1991 and 1995 the rate of improvement in cost per megabyte of personal computer disks was about 2.0 per year, a considerable increase over the previous rate of about 1.35 per year between 1986 and 1991. Reasons for this higher rate include the simultaneous shrinking of disk electronics due to semiconductor advances, reductions in manufacturing cost due to much higher volume, and the switch to constant bit density per track. In this high volume and rapidly changing industry, a typical disk manufacturing process is used for just 9 months before switching to a new technology.

This five-year trend, which was discovered because only because we examined costs, will prove crucial in the design of DIY-RAIDs.

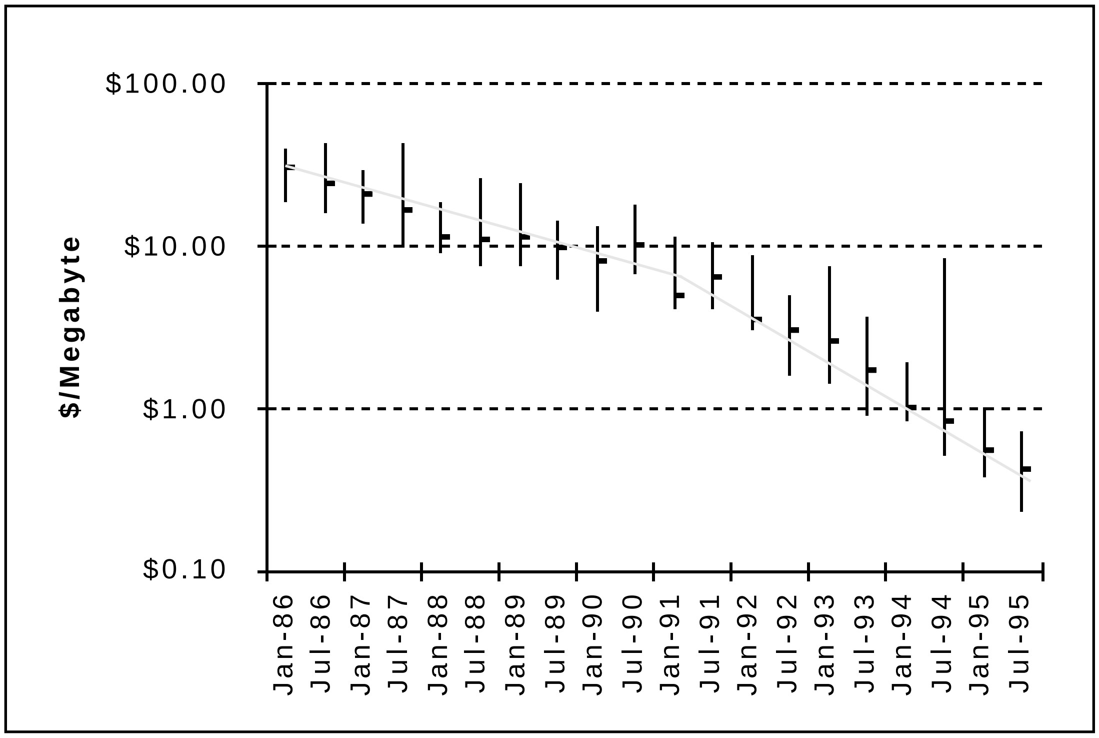

FIGURE 1. Maximum, minimum, and median dollars per megabyte of personal computer disks over time. The top of each vertical line is the maximum price per megabyte for that month, the bottom is the minimum price, and the box shows the median price. For example, in July 1995 the respective prices per megabyte were $0.71, $0.24, and $0.42. Note that in the 5 years between January of 1986 and January of 1991 the median price fell by a factor of 6, but in the 4.5 years between January 1991 and July 1995 the median price fell by a factor of 12. Prices were collected from advertisements in the January and July editions of Byte magazine, by Mike Dahlin of U.C. Berkeley.

Disk Interfaces

The second component of a DIY-RAID is the interface to the disks. A SCSI disk interface can be either 8-bit and 16-bit (wide), can connect up to 7 and 15 disks, respectively, and run at 5 MHz, 10 MHz (fast), and 20 MHz (ultra). A 16-bit SCSI adapter can work with any version of SCSI disk, since the SCSI protocol negotiates the transfer speed and there are simple bridges to plug a 16-bit SCSI disk into an 8-bit connector. The maximum number of disks and bandwidth of the SCSI strings affect the cost of the DIY-RAID, and the support of heterogeneous SCSI interfaces enable the reuse of the strings and adapters across generations of disks. Since SCSI is well established, its costs are low and the components are reliable. Both issues will be important to DIY-RAID cost-performance.

There are also new proposed serial interfaces, such as SSA and FC-AL. Both are serial connections that can connect many disks per string: 126 for FC-AL and 127 for SSA. SSA consists of four 20 MB/sec serial cables, supporting two reads and two writes simultaneously. FC-AL offers 100 MB/sec on a single string; the typical FC-AL disk is dual ported for higher availability (see Figure 17 on page 23). These multiple paths can provide advantages in performance and reliability for DIY-RAIDs, as we will see later.

Disk manufacturers expect high-end disks with each interface at an extra charge relative to the price of 16-bit SCSI of roughly $30 for SSA and $100 for FC-AL[29]. Even these small extra costs significantly affect the overall cost-performance of DIY-RAIDs.

Computers and Buses

A natural candidate for connecting LANs to many disks is the personal computer, especially when cost is a consideration. At about $3400 for a 100-MHz Pentium-based PC in late 1995, it is an attractive engine to support an array of disks.

The standard bus of the personal computer is Peripheral Component Interconnect (PCI), which has a peak rate of 132 MB/sec. A crucial characteristic of PCI is its ability to extend beyond a PC box, at the cost of eight-clock cycle latency per bridge to a PCI extension box [13]. Multiple bridges can be chained together. At seven I/O adapters per PCI extension box, this bus offers an attractive building block for a DIY-RAID.

Switch-based LANs

Switch-based LANs provide two important functions. The first is an economical and scalable interface between the DIY-RAID and the client computers. Hardware RAIDs, because of their high bandwidth per box, often use HIPPI or FibreChannel to connect to the outside world. These expensive buses affect the cost the RAIDs and the type of computers to which they can be connected. On the other hand, low cost interfaces exist for 155 Mbit per second ATM and 640 Mbit per second Myrinet [5, 6], allowing direct connections to desktop computers.

The second function of switch-based LANs is to enable the personal computers within the DIY-RAID to communicate and access information on disks on other PCs, making it plausible to stripe data across several computers, thereby increasing both bandwidth and availability.

Enclosures and Racks

The disks of a DIY-RAID need power, cooling, and a place to live. Fortunately, several companies sell disk enclosures which include power supplies, fans, and sheet metal to hold disks. Many enclosures provide cannisters in which to place disks, allowing them to slide in and out of the enclosure for easy disk installation and repair.

Enclosures typically hold 3 to 16 disks. Large systems can be assembled by stacking many enclosures in a standard 19-inch wide by 7-foot tall rack, and then using as many racks as necessary. Standard racks allow PCs, disk enclosures, and LANs to be co-located.

System Software

The final component of a DIY-RAID is the software to turn a collection of disks on a network into a usable storage system. The issue of how this software should be designed is beyond the scope of this paper. However, commercial software already exists for RAID: for example, Microsoft Windows NT implements RAID 5 across disks connected to a single CPU as a standard part of the operating system. An extension to support “hot swap”, key to achieving high-availability in a DIY-RAID, will be available in Spring 1996. In addition, research prototypes have existed for several years that extend a software RAID over the network.

A key observation is that many applications needing large storage—including digital libraries, video on demand, and decision support systems—read much more often than they write. For example, decision support requires about four 8-KB reads for every gigabyte of storage, and is read mostly[7]. As another example, we intend to construct a DIY-RAID prototype to store a local copy of the HTML pages of the World Wide Web. We expect virtually all the accesses to be reads, with writes limited to adding new home pages found in overnight searches of the Web. Such applications reduce the importance of small writes and thus hardware RAIDs, and allow larger parity groups, thereby reducing parity overhead.

An Example DIY-RAID

Figure 2 shows a simple example: 8 disks spread over two 16-bit SCSI chains connected to a PC which is connected to LAN. The total cost of this small DIY-RAID—PC, adapters, cables, enclosures, and disks—is about $14,500, and it has a data capacity of 30 GB. In contrast, an equivalent hardware RAID costs 2.5 times more: a DEC StorageWorks 410 for the same capacity costs $36,500. This small system could run at file system like Windows NT, so the PC would perform parity calculations needed for writes.

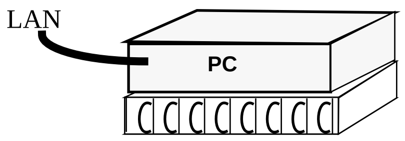

FIGURE 2. A small DIY-RAID example using 16-bit SCSI disks and enclosures and PC. The PC includes a network interface and two SCSI adapters so as to get better performance from the eight disks. Each disk enclosure hold eight 4.2 GB 3.5” disks and cannisters, dual power supplies, dual fans, and a portion of the SCSI chain. Each cannister has a handle to simplify disk installation and removal. The parity groups size is eight, so the data capacity is seven disks.

3. An Architecture for DIY-RAIDs

Figure 2 shows the simplest DIY-RAID: use the PC to connect to the network, and then connect the disks to the PC. Alas, there are three severe weaknesses to this organization. The first is size: PCs have a limited number of slots for bus adapters, typically two or three, so this organization cannot scale up to 1000 disks. The second weakness is performance: even if you did connect hundreds of disks, the performance would be limited by the speed of the single processor, the speed of the single PCI bus, or the speed of single network interface. The third fatal weakness is reliability: the reliability is limited to that of the PC; if PC or the network interface fails, the disks are unavailable.

Figure 3 shows the proposed architecture of a DIY-RAID to address these weakness. The basic orga-

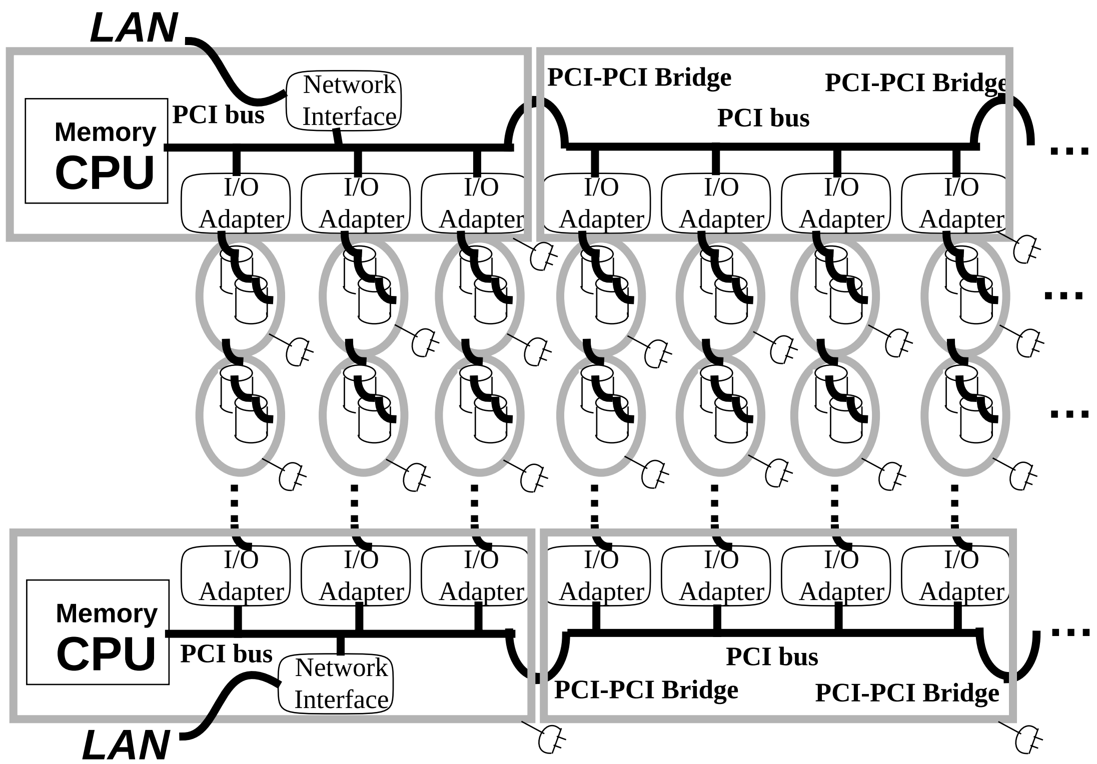

FIGURE 3. Logical organization of a DIY-RAID. Each PC is connected via a PCI-PCI bridge to a chain of PCI extension boxes. Each PC is also connected to a LAN. I/O adapters are connected to PCI buses and to strings of disks. To improve availability, each string has two I/O adapters, allowing every disk to be connected to two PCs. This organization applies to 16-bit SCSI, and with a little change, to 8-bit SCSI, SSA, and FC-AL. Disk enclosures are shown as grey ovals and PC and PCI extensions as grey rectangles. Each grey oval or rectangle in this figure includes its own power supply.

nization is to use PCs plus bus extension boxes to connect the I/O adapters and to the LAN, with each disk connected to two PCs.

This architecture can scale in size by adding bus extension boxes and adapters to increase the number of strings and thereby connect to hundreds of disks. It can scale in performance by using the network to connect multiple PCs to the external computing system. By having the external computing system stripe data across the network ports, the disk bandwidth can scale and yet any PC can access any disk. Increasing the number of PCs increases both the processing power and the bus bandwidth. Finally, the DIY-RAID can scale in reliability: each disk is connected to two PCs via two I/O adapters so that no single failures of any part of the system will result in unavailable data. Also, this architecture follows the orthogonal RAID design mentioned above, with parity groups organized across the strings and enclosures to avoid a single point of failure.

As we shall see, varying the number of PCs and the number of disks per string determines the cost-performance of the DIY-RAID architecture. The cost/reliability is determined by varying the parity group size and the number of hot standby spare disks.

4. One Implementation of the DIY-RAID Architecture

Our goal is a DIY-RAID that can scale performance and capacity with up to 1000 disks while maintaining cost/megabyte close to a single PC disk and reliability greater than a single disk. To make our point, we need concrete examples. To show the scaling, we show examples at 16 disks, 112 disks, and 1120 disks.

The 16 disk version is similar to Figure 2, except we add a second enclosure to hold more disks and add a second PC and adapters to increase reliability. The second PC also helps with performance. This version is now about half the price of an equivalent DEC StorageWorks 410.

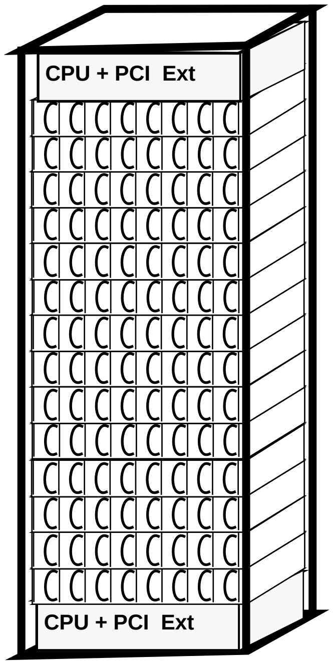

FIGURE 4. DIY-RAID example using 16-bit SCSI enclosures and PCI bus extension boxes. A rack holds 2 PCs, 4 PCI Extension boxes, and 14 disk enclosures. The enclosures are the same as in Figure 2. We chose one enclosure per string, so a rack supports 14 SCSI chains and 112 disks. The maximum length of a SCSI chain is about 20 feet (6 meters) and the maximum distance between a PC box and a disk is about 7 feet (1.8 meters), so the length of the SCSI chain should not be a problem. Assuming a rack is 19 inches wide, 19 inches deep, and 90 inches high (0.5 x 0.5 x 2.2 meters), the footprint for a rack is 2.5 square feet (0.25 sq. meters). Each disk dissipates about 13 watts, making the total power requirement 1.5 KW for 112 disks and 0.25 KW for PCs and PC expansion boxes, or $1850 per year at current power prices. Using 4.2 GB disks available in 1995, a parity group size of 14 and 8 hot spare disks, the total disk capacity of a rack is about 400 gigabytes. The 1995 cost is about $150,000.

Figure 4 shows the 112-disk implementation based on 16-bit SCSI disks and strings.A rack, with a footprint of 2.5 square feet, has 2 PCs, 4 PCI extension boxes, and 14 of the Sigma Designs SA-H346 enclosures.

There are 14 strings, so at one enclosure per string a rack holds 112 disks. Using 4.2 GB Segate Hawk disks, a rack consumes about 1500 watts for the disks and 250 watts for the PCs and PCI Extension boxes. With a parity group size of 14 and 8 spare disks, the effective useful capacity is 400 GB. Standard RAID software would perform the parity calculations on the PCs.

The 1120 disk system comes from putting 10 of these racks together. One difference is we can now stripe across the network interfaces, allowing a much larger parity group. Another difference is that we can use a larger parity group size of 35 disks and provide just 16 hot spare disks for all 10 racks, so the total usable capacity is 4.5 terabytes—12% more than 10 times the capacity of a single rack. We also need the client computers to be running a software system such as Zebra [20] or xFS [31] which will stripe data across the network ports, which means the client computers calculate parity. Both of these software packages use batch writes to a log to reduce the amount of parity calculation. To connect the 20 network interfaces to he external network we use three 16-port switches.

The examples presented in this section are certainly not the only solutions, and perhaps not even be the best, but they are sufficient to make our point. As we shall see in the following sections, this organization is capable of delivering the full bandwidth of more than 1000 disks for small reads and almost one gigabyte/second for large reads with reliability greater than a single disk at a cost/capacity of 25% over a single disk.

5. Cost/Performance of DIY-RAID Alternatives

In this section we present models of cost and performance, present guidelines to improve cost and performance, and then quantitatively evaluate trade-offs in cost-performance for several DIY-RAID options.

Cost Model

To be able to talk about cost/performance, we must be able to talk about cost. In particular, we need to quantify trade-offs of one component against another. This trade-off is especially challenging because we are comparing very different objects: rotating storage devices, integrated circuits, wires, and sheet metal boxes with printed circuit boards, power supplies and fans.

We chose price as the measure of cost. When mainframes ruled the earth, price had little to do with cost. In this PC world of 1995, low margins mean prices must track costs for high volume components, and so there is less danger in being misled by prices. Figure 5 shows the prices of the components of a 10 rack version of the system in Figure 4.

What is important about Figure 5 is not the actual dollars, but the relative costs. Figure 6 shows the prices normalized to a 16-bit SCSI disk. We'll use these ratios throughout the section, allowing us to trade, say, more adapters and PCI extension boxes for fewer disks. Such costs will be expressed in 1995 dollars, however, since dollars are more intuitive than "disk-equivalent units". At the end of this section we'll explore other ratios as a result of: 1) amortizing the cost of the DIY-RAID infrastructure across multiple disk lifetimes and therefore lowering its costs, and 2) assuming there wasn't a cost advantage to high volume and so the costs of all disk interfaces are the same.

Performance Model

Since I/O performance is limited by the weakest link in the chain, we evaluate the maximum perfor-

Computers	PC	PCI Ext. Box	Network Interface	Switches	Uninterruptable Power Supply	Total
Number	20	40	20	3	1	--
Cost each	$3,400	$859	$1,300	$9,600	$13,380	--
Total	$68,000	$34,360	$26,000	$28,800	$13,380	$170,540
SCSI-16	**Adapters**	**Cables**	**Enclosures & Cannisters**	**Racks**	**Disks**	**Total**
Number	280	280	140	10	1120	--
Cost each (list)	$276	$79	$2,192	$350	$1,064	--
Cost each (qty 100)	$276	$79	$1,134	$350	$997	--
Total (list)	$77,280	$22,120	$306,835	$3,500	$1,191,680	$1,601,415
Total (qty 100)	$77,280	$22,120	$158,760	$3,500	$1,116,640	$1,378,300
System Total (qty 100 prices)						$1,548,840

FIGURE 5. Number and price in late-1995 of computer and storage components for fast 16-bit SCSI for the 10-rack version of Figure 4.

Item	Units	Item	Units
PC	3.4	Enclosure	1.1
PCI Extension Box	0.8	I/O Cable	0.1
Network Interface	1.3	Rack	0.4
I/O Adapter	0.3	Disk	1.0

FIGURE 6. Proposed relative costs of DIY-RAID components, with disk cost set as 1. If we have 12 strings with 8 disks per string and a parity group size of 12, we can calculate the cost impact of changing to 24 strings with 4 disks per string and a parity group size of 24. We save 4 disks of parity information or 4 units at a cost 10 units: 24 adapters, 2 PCI extension boxes, and 12 cables. Thus this hypothetical change actually increases cost by the equivalent of 6 disks as opposed to the savings of 4 disks that one would expect if usable storage capacity was our indirect measure of cost.

mance of each link in the I/O chain to determine the maximum performance of the whole organization, following the I/O performance model of [21]. We assume that the workload is evenly divided between all disks.

Figure 7 shows the maximum performance of several components for 8-KB reads and 1-MB reads. We show both the theoretical peak performance and the practical peak performance due to limitations of memory

	PC CPU	PCI/ Mem	Net. Int.	Disks	SCSI 8-bit		SCSI 16-bit		SSA		FC-AL	
					Str.	Adpt.	Str.	Adpt.	Str.	Adpt.	Str.	Adp.
Theor BW	--	132 MB/s	80 MB/s	8 MB/s	10 MB/s	10 MB/s	20 MB/s	20 MB/s	4 x 20 MB/s	4 x 20 MB/s	2 x 100 MB/s	2 x 100 MB/s
Peak BW	--	80 MB/s	80 MB/s	6 MB/s	6/8 MB/s	6/8 MB/s	12/16 MB/s	12/16 MB/s	4 x 17 MB/s	4 x 17 MB/s	2 x 80/90 MB/s	2 x 80/90 MB/s
% BW used	100%	40%	80%	100%	80%	100%	80%	100%	80%	100%	80%	100%
Laten-cy	0.25 ms	--	--	12.2 ms	--	1.0 ms	--	1.0 ms	--	0.5 ms	--	0.5 ms
8 KB reads/ sec	4000	4096	8,192	63	614	434	1229	606	6,963	1,626	12,288	1,587
1 MB reads/ sec	4000	32	64	5.5	6	8	13	16	66	54	128	77

FIGURE 7. Bandwidth, latency, 8 KB reads per second, and 1 MB reads per second of DIY-RAID components. The difference between the theoretical and peak performance is due to environmental factors: for PCI bus it is due to the speed of memory in the PC, for disk it is the difference between the outer track and the average track, and for the I/O strings we subtract out the command overhead. For strings, the lower number in the peak bandwidth row is for 8-KB accesses and the higher for 1-MB accesses.We limit strings and PCI buses are limited to at most 80% of their peak bandwidth when calculating reads per second. As explained in the text, each disk access must go over the PCI bus *twice*, and so the bus performance is about half what you might expect. The SSA we use has two loops per adapter; it can read 2 times 17 MB/second on each loop. Similarly, FC-AL also offers double-ended disks (see Figure 17 on page 23) and hence two loops, so FC-AL strings can perform about twice as many reads as you would expect from the bandwidth. Ultra 16-bit SCSI as not listed for space reasons, but is assumed be twice the bandwidth as 16-bit SCSI but the same adapter overhead. The purpose of the UPS is explained in Section 6.

systems or command protocol overhead.Queuing theory tells use that you cannot use 100% of a bus without introducing inordinate queueing delays, so we use 80%. The percentage used is a matter of experience and the application: the lower the percentage the better the response time, the higher the percentage the better the bandwidth. We show a sample set of percentages to give an example. The basic calculation for this figure is the time of an access: latency plus the block size divided by the product of the bandwidth and the percentage that the unit can be used. The I/Os per second in the last two rows is just 1/latency.

The number of I/Os per second (IOPS) for the CPU was determined from a series of experiments run by ourselves and others [35, 15, 12]. We assume each request runs entirely inside the kernel to avoid context switches. The steps are: 1) the network packet comes in requesting a read; 2) the kernel makes a disk request, setting up DMA; 3) the kernel handles the disk request completion interrupt; and 4) network packet goes out with data. We also assume the DMA from the disk lands in a buffer aligned to the message packet header, so no further copy is needed to send out the data over the network.We believe a 100 MHz Pentium PC will take 250 microseconds per 8KB request, or 4000 IOPS. Our network costs are based on a reliable network (Myrinet), so we do not include a checksum calculation in the CPU overhead. (See the discussion on end-to-end reliability in Section 6.)

The PCI bus maximum performance is currently limited to 80 MB/second by the PC memory system. Backing off by 20% yields gives a usable bus/memory bandwidth of about 8192 8-KB reads and 64 1-MB reads. Since each block must go over the bus into memory and back out of memory over the bus to the network, the effective number is 4096 8-KB reads and 32 1-MB reads. Note that for 8-KB reads, the CPU and PCI bus/memory system are evenly matched.

The next link in the I/O chain is the SCSI adapters. We used the numbers for SCSI adapter from [21] as 1 ms plus the transfer time. We estimated that the overhead for SSA and FC-AL would be half that of SCSI.

Now that we have calculated the performance of the components, we can calculate the I/O rate for the whole system. For the DIY-RAID implementation in Figure 4, we have 20 CPUs and PCI buses, 8 disks per string, and 140 fast, wide SCSI strings each connected to two PCI buses and two CPUs. Let's assume there is sufficient external network bandwidth. Using the format

Min(CPU limit, PCI bus limit, strings x bandwidth limit per string, strings x 8 x disk limit, strings x 2 x controller limit)

The maximum performance is limited by the bottleneck (underlined in boldface):

$$\text{Min}(20 \times 4000,\ 20 \times 2096,\ 140 \times 1229,\ 140 \times 8 \times 63,\ 140 \times 2 \times 606) =$$

Min(80000, 81920, 172060, **<u>70560</u>**, 169680) = 70560 8-KB reads per second

Thus this organization is limited by disk speed.

Let's redo this calculation for 1 MB reads. Using the performance of components for 1-MB reads under 16-bit SCSI in Figure 7, the calculation becomes:

$$\text{Min}(20 \times 4000,\ 20 \times 32,\ 140 \times 13,\ 140 \times 8 \times 5.5,\ 140 \times 2 \times 16) =$$

Min(80000, **<u>640</u>**, 1820, 6160, 4480) = 640 1-MB reads per second

This configuration is limited by the bandwidth of the PCI bus.

Before presenting the detailed cost and performance of several organizations, we give guidelines on cost and performance for implementations of the DIY-RAID architecture in Figure 3.

Low Cost Guidelines

For a fixed number of disks, to get even lower costs:

- *Reduce the number of PCs.* The lowest cost point is simply two PCs for all racks, replacing the other PC with PCI extension boxes. This version would string PCI extension cables between the racks. Of course, this guideline lowers cost by lowering performance.

- *Maximize the number of disks per string.* This advice reduces the number of adapters and PCI Extension boxes by reducing the number of strings. Again, this guideline trades performance for lower cost.
- *Maximize the number of disks per parity group.* Reducing the number of redundant disks improves the cost/megabyte of the system. This advice trades write performance for lower cost.
- *Use disk enclosures that allow customer installation and replacement*. When the cost per megabyte improves 7% per month, a solution that adds three to six months between when disks are available and when they are put to use is expensive. Conversely, installing hundreds of disks six months before anyone puts data on them is also expensive. The system must allow "just in time" population of the DIY-RAID. Hardware RAIDs proved users are willing to replace disks when they fail; DIY-RAIDs take it the next step to enable users to *place* disks when they are installed.

The physical construction of the DIY-RAID places practical limits on the values of these architectural parameters. For example, disk interface standards determine maximum number of devices per string. Because dependent failures must not result in the loss of multiple disks per parity group, another restriction is to balance the number of strings and parity group size so as to avoid placing disks from the same group in a single enclosure. Later in this section, we'll show how to pick these parameters to optimize cost.

Having covered costs, let's switch to performance.

Performance Guidelines

Fundamental to I/O is that performance is set by the weakest link in the chain between the client computer and disk, so we strive for a balanced design: too much performance on one "link" wastes money, and too little on another throttles performance. The performance guidelines are:

- *Determine the maximum I/O rate of the disks.* Over-designing the infrastructure adds cost but not performance.
- *Make the number of PCs match the I/O rate target.* The number of PCs determines both the number of CPUs, important for small accesses, and number of PCI buses, important for large accesses. Without sufficient PCs, any application performance target cannot be met.
- *Make the number of disks per strings match the I/O rate target.* Given that the cost of the disks on a string are likely to be much more than the string and adapters, to get the best cost-performance we must ensure that disks limit string performance, not the adapters or the string. Given a fixed number of disks, this parameter also determines the number of strings and hence the number of adapters per PCI bus. Thus we can also think of this parameter as ensuring that there are enough adapters and strings to drive each PCI bus at its target rate. An imbalance can result in strings or adapters being the weakest link in the chain.

Quantifying Cost-Performance of Several DIY-RAIDs

The prior sections gave the cost and performance of the DIY-RAID implementation in Figure 4. In this section we explore the cost-performance for 8-KB and 1-MB reads for a variety of configurations, including those using fast 8-bit SCSI, ultra 16-bit SCSI, SSA, and FC-AL. Unlike the earlier example in Section 4, where we picked a strawman and then evaluated its cost, in this section we explore the limits of cost and performance by finding the minimal cost system that meets a performance level.

We pick as a target a system with 1120 disks, a RAID 5 group size of 35, and 16 spare disks. The next section evaluates the reliability of that configuration. We assume the load is balanced across the disks, and that the 16 hot spares really consist of space being reserved across 1120 disks so that the system can gain the benefit of both hot spares *and* the performance advantage of more disks [17].

		Adapters	**Cables**	**Enclosures & Cannisters**	**Disks**
SCSI-8 Fast	**Cost each (qty 100)**	**$150**	**$49**	**$1,134**	**$934**
SCSI-16 Ultra	**Cost each (qty 100)**	**$250**	**$79**	**$1,134**	**$997**
SSA	**Cost each (qty 100)**	**$750**	**$49**	**$2,123**	**$1,027**
FC-AL	**Cost each (qty 100)**	**$1,000**	**$49**	**$2,123**	**$1,097**

FIGURE 8. Cost of storage components for 8-bit SCSI, SSA, and FC-AL. The prices for SCSI-8 are actual; prices for ultra, SSA and FC-AL adapters for disks are based on averaging predictions for 1996 from several people in the industry [3], [11], [16], and [29]. The enclosures and cannister prices for SSA and FC-AL are based on the Sigma H347 designed for FC-AL. The cost of SCSI enclosures is the same 16-bit enclosure price at quantity 100 for the Sigma H346.

Similar to Figure 5, Figure 8 shows the costs for the I/O components for the other string interfaces. We chose current prices except for SSA and FC-AL because it would be unfair to SSA and FC-AL given their youth. For example, a FC-AL adapter costs $4000 in 1995. Hence we optimistically use estimates of mid-1996 prices for SSA and FC-AL adapters and disks[3,12,16,29].

Given the costs, we use the performance model above to determine cost-performance. Figures 9 and 10 compare cost-performance of several organizations for 8-bit SCSI, 16-bit SCSI, SSA and FC-AL. The X-axis is the increase in percent cost/megabyte over a quantity-one 16-bit SCSI disk cost/megabyte, which we call the "DIY-RAID tax". The tax includes the cost of other components, parity, and spare disks. The Y-axis is the number of reads per second that the system can deliver; Figure 9 shows 8-KB reads and Figure 10 shows 1-MB reads. The number of PCs for each configuration is identified on the left of the figures; it is key to performance because it gives both the number of CPUs and the number of PCI buses. The single number near each point is the number of disks per string. Since the total number of disks are fixed in these figures, this number also determines the number of strings and therefore adapters. Both the number of PCs and number of disks per string were picked to minimize cost at a given performance level.

Let's start with Figure 9. At the low end, performance is limited by the number of CPUs. Given sufficient number of CPUs, it is limited by the number of disks at the high end. In terms of costs, higher priced enclosures plus the extra cost per disk explains the cost difference between the SCSI options and SSA or FC-AL. Although SSA allows for many more disks per string and hence fewer adapters and PCI extension boxes, this doesn't make up for the added costs. Unlike SSA, FC-AL doesn't really work well with large numbers of disks per string at the low end of performance. As we shall see in the next section, a large number of disks on the FC-AL string means fewer strings and so the parity group size must be shrunk to match the number of strings resulting in lower effective usable storage. (The point with 124 disks per string is included to show

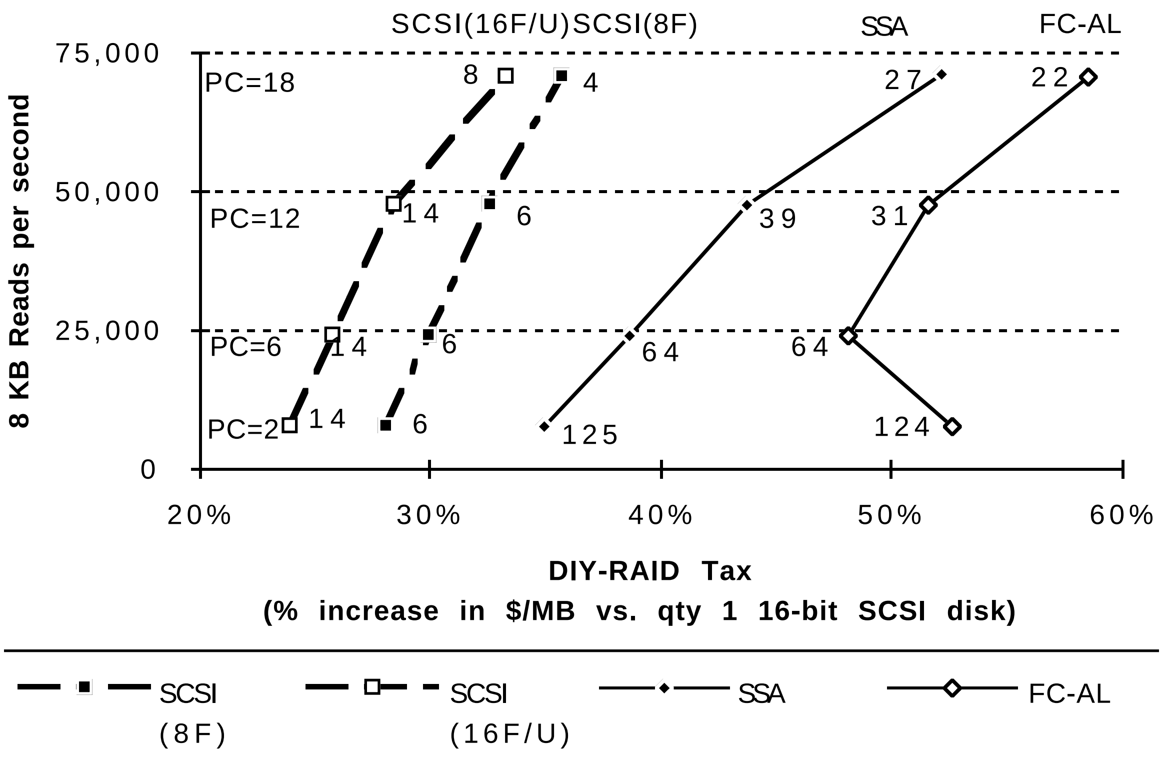

FIGURE 9. Improvement in small read performance vs. cost as number of PCs and strings increase for a 1120 disk DIY-RAID. The notation gives number of PCs on the left and number of disks per string at each point. To get as much performance from the disks as possible for small reads, we need 18 CPUs. For 16-bit SCSI to operate at full bandwidth, we need more strings which implies decreasing disks per string. This ratio affects the total cost of disk enclosures: because we must ensure that dependent failures do not result in the loss of multiple disks per parity group, we assume each enclosure has only one string. At 14 disks per string and using 8-disk enclosures, we waste two slots. The next smaller and efficient size is 8 disks per string, which fully utilizes 8-disk enclosures. With 18 PCs and 8 disks per string the SCSI design gives virtually the full bandwidth of 1120 disks performing 8 KB reads at a premium of 37% over a single versus a single disk. The cost for 8-bit SCSI is due to having many more strings and thus many more adapters and PCI extension boxes, even though the disks and adapters are cheaper. The higher cost per disk interface and higher cost of disk enclosures for SSA and FC-AL offset the advantage of using larger number of disks per string. FC-AL needs smaller parity groups when the disks per string is large which increases the cost per megabyte for 124 and 64 disks per string.

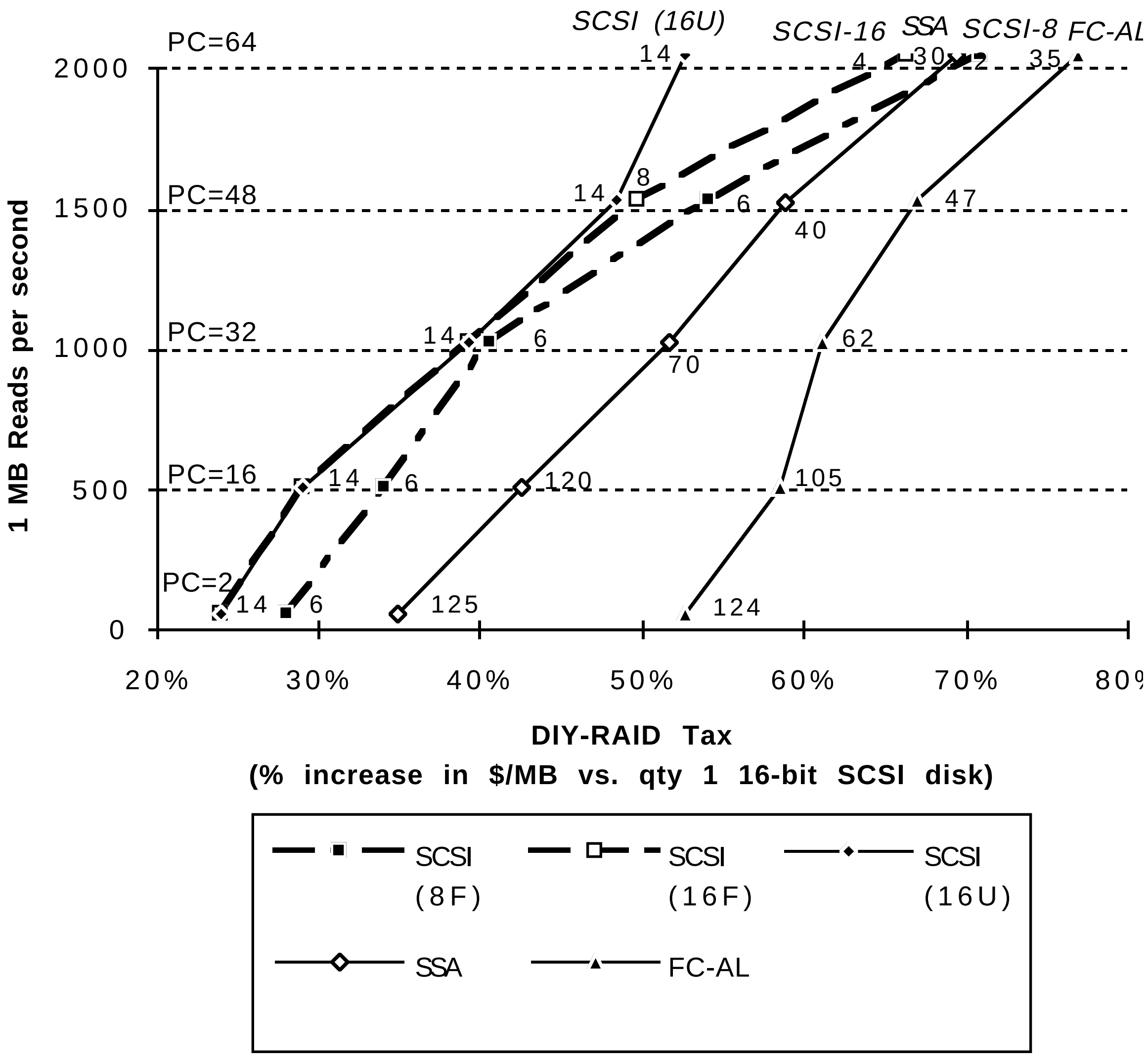

FIGURE 10. Improvement in large read performance as PCs and strings increase for a 1120 disk DIY-RAID. The notation is the same as in Figure 9. Since performance is limited by PCI bus at 32 1-MB reads per second, each low cost organization can deliver just 64 reads per second. For 16-bit Fast SCSI we need more strings to achieve higher performance, obtained by reducing the disks per string from 14 to 8; with a tax of about 65%, 64 PCs with 4 disks per string delivers 2048 MB/sec. 8-bit SCSI needs even more strings to keep up, and so the number of disks per string is reduced from 4 to 2. The apparent upturn in cost-performance of 16-bit ultra SCSI is because when we add 16 CPUs to go from 48 to 64 we can avoid all the PCI Extension boxes.

this effect; obviously no one would purposely build a system with lower performance *and* higher cost.) As performance demands increase, FC-AL and SSA need more strings which leads to many more adapters and PCI extension boxes for SSA and FC-AL, which further separates the costs from SCSI.

One cost issue is sensitivity of small accesses to CPU I/O overhead. For example, if overhead for network and file access was 500 microseconds instead of 250 microseconds, the tax would rise 7% for high read rates to pay for more PCs. If it could be reduced to 125 microseconds, cost could not change! At 250 microseconds of overhead, a CPU supports 4000 8-KB reads per second, which is slightly slower than the limit of 4096 of the PC memory (see Figure 7). Such a change merely makes memory the weakest link. Thus reductions in CPU overhead, which we can expect from faster CPUs and faster network protocols[23,30], must be accompanied by faster PC memory systems to help our application.

We can also calculate the cost impact that comes from changing the adapter overhead for high read rates: doubling overhead adds 2% to 6% to costs as a result of increasing the number of strings, and halving overhead lowers costs by 3% to 6%. No combinations of doubling or halving adapter overheads would change our conclusions about the relative cost-performance of the disk interfaces.

As a basis for performance comparison, Brady gives a rule of thumb for decision support applications of four 8-KB reads per second per gigabyte of storage[7]. Thus 1120 disks require 18000 reads per second for decision support, met by DIY-RAIDs with 25% overhead.

For a workload of 1-MB reads in Figure 10, at the low end of performance the CPUs are no longer become the bottleneck; we rely on DMA to transfer the data to and from memory. Instead, the 1-MB transfers will put the pressure on the PCI buses: Figure 7 shows that a PCI bus can sustain 32 1-MB reads per second. Thus the PC is still the bottleneck even though the CPU isn't. Thus doubling or halving the CPU overhead has virtually no impact because the bottlenecks are the times to transfer on the PCI bus. The same is true for doubling or halving adapter overhead and the string speed.

At the high end of performance, 1120 disks can potentially deliver about 10,000 1-MB reads per second. Few external computer systems can absorb that much information, so we only plot up to 2048 MB/second. Over the entire range, the number of PCs set the performance limit. Although the higher cost of the enclosures and disks still make SSA and FC-AL more expensive, their higher bandwidth strings are closer in performance to fast SCSI at higher performance levels.

FC-AL is still handicapped at the low performance end by the smaller parity group. At the high end its string bandwidth could support many more disks per string. The problem is we also need more PCs at the high end. Since we must have one string for each pair of PCs, it is the PCs that end up determining the number of disks, not the FC-AL link bandwidth.

In both figures we also plotted the performance of ultra SCSI system with twice the clock rate and the same width, which will be available at the about the same time as SSA and FC-AL systems are economical. We use the same adapter overhead as the fast 16-bit SCSI adapter. There is no difference in performance for 8-KB reads, but there is above 1500 MB/second. Ultra SCSI is fast enough to maintain the 14 disks per string all the way to 2048 MB/second.The apparent jump in cost-performance of ultra 16-bit SCSI is because 64 CPUs and 14 disks per string require no PCI Extension boxes. Thus we save the cost of 48 PCI extension boxes when we go from 48 to 64 PCs, thereby gaining performance at much less cost than from going from 32 to 48 PCs.

Ignoring Volume

These conclusions are based on the best information we have available, but they are tentative given that the true cost of SSA and FC-AL are yet to be determined. Given the long history of SCSI and the uncertain commercial future of SSA and FC-AL, the SCSI price and performance predictions are likely the most accurate and the most conservative. In particular, a key to the cost of the system is the price of the disks. Our model is based on estimates of the disk manufacturer's cost for each interface assuming similar volumes. If we based prices on volume and competition, then it would seem that SCSI might have a even greater cost advantage over SSA and FC-AL. We are surprised that our potentially optimistic assumptions of the newer disk interfaces do not make them look more attractive.

To bound the impact of volume, Figures 11 and 12 plot the same performance as Figures 9 and 10, but this time we make the costs for 16-bit SCSI, SSA, and FC-AL the same as for 8-bit SCSI. As we can see, even with this radical assumption, the advantages of SSA and FC-AL are not pronounced over 16-bit SCSI.

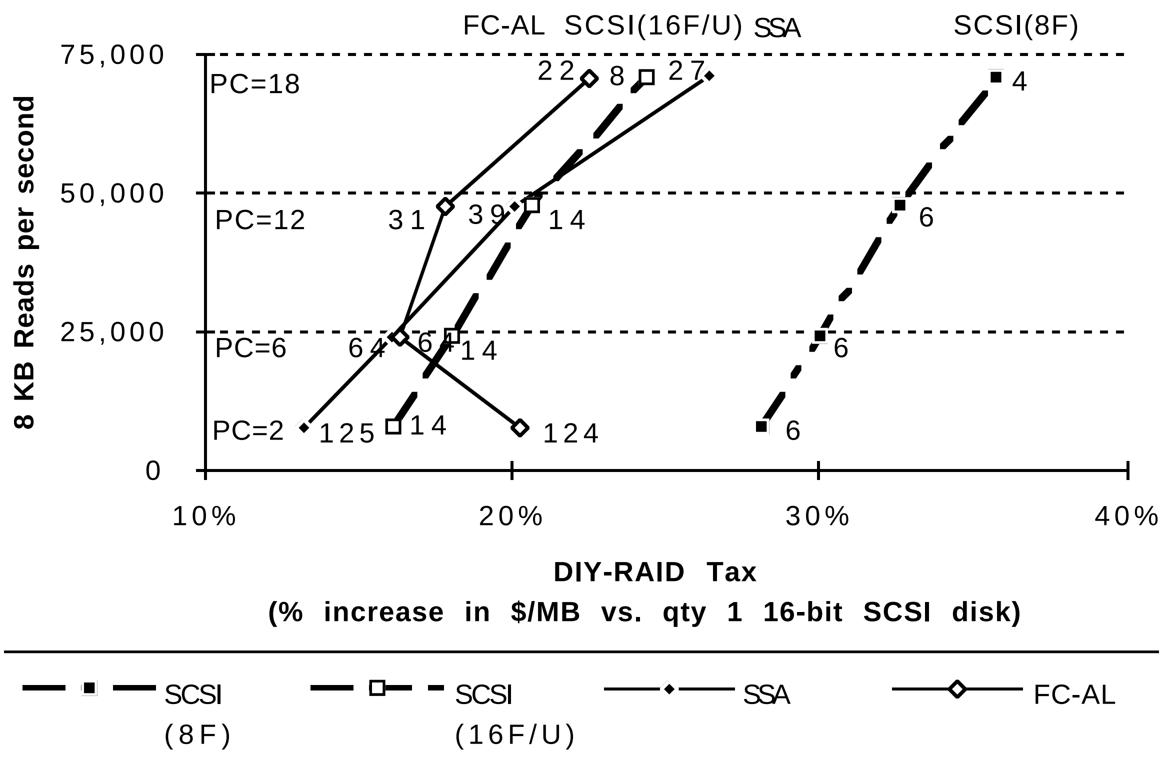

FIGURE 11. Same as Figure 9, except the costs of components for different strings are assumed to be the same. Clearly 8-bit SCSI is attractive only because of its cost.

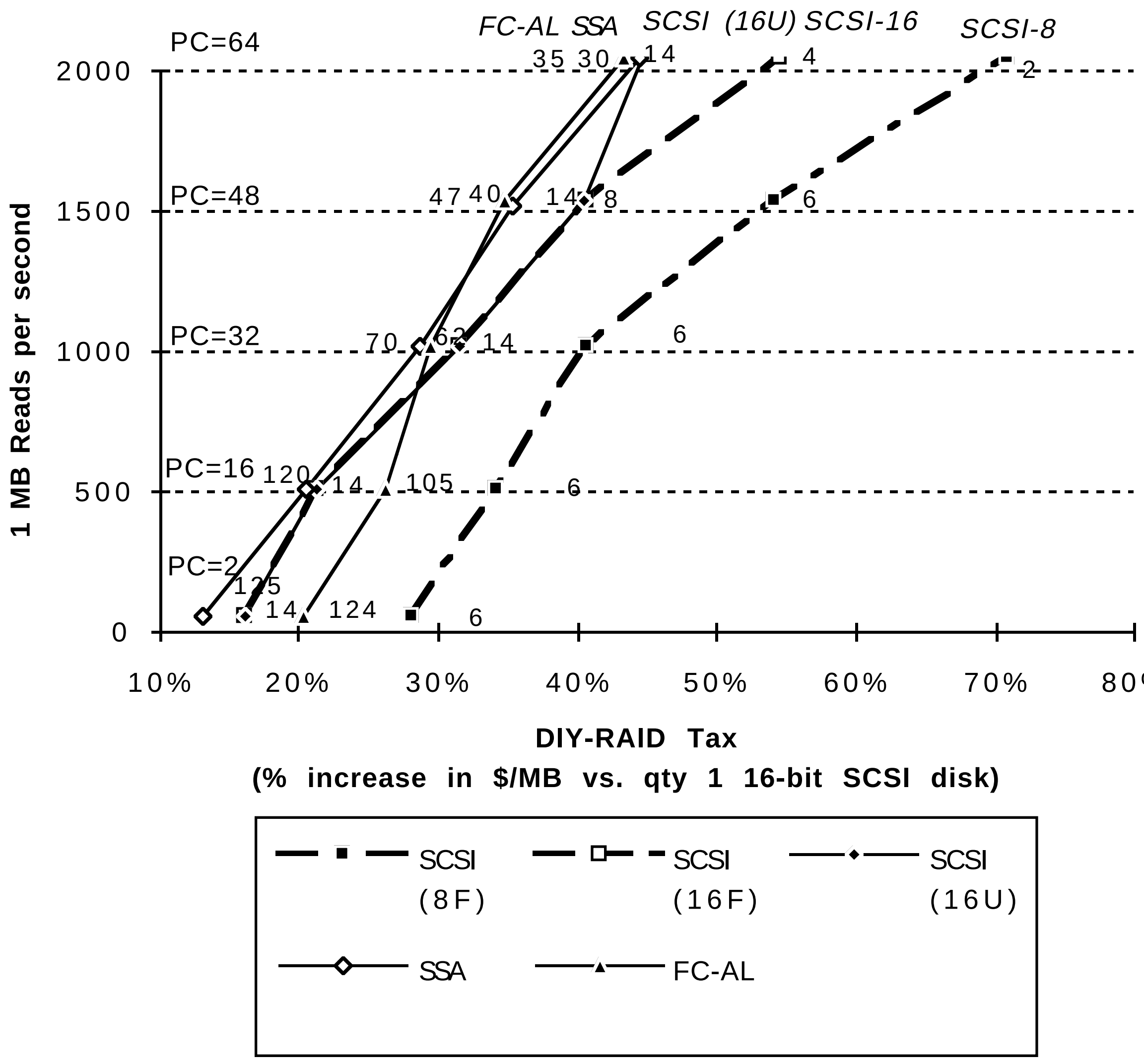

FIGURE 12. Same as Figure 10, except the costs of components for different strings are assumed to be the same.

Recycling Storage Infrastructure

The costs presented in Figures 9 and 10 assume we would buy everything when we want to upgrade the

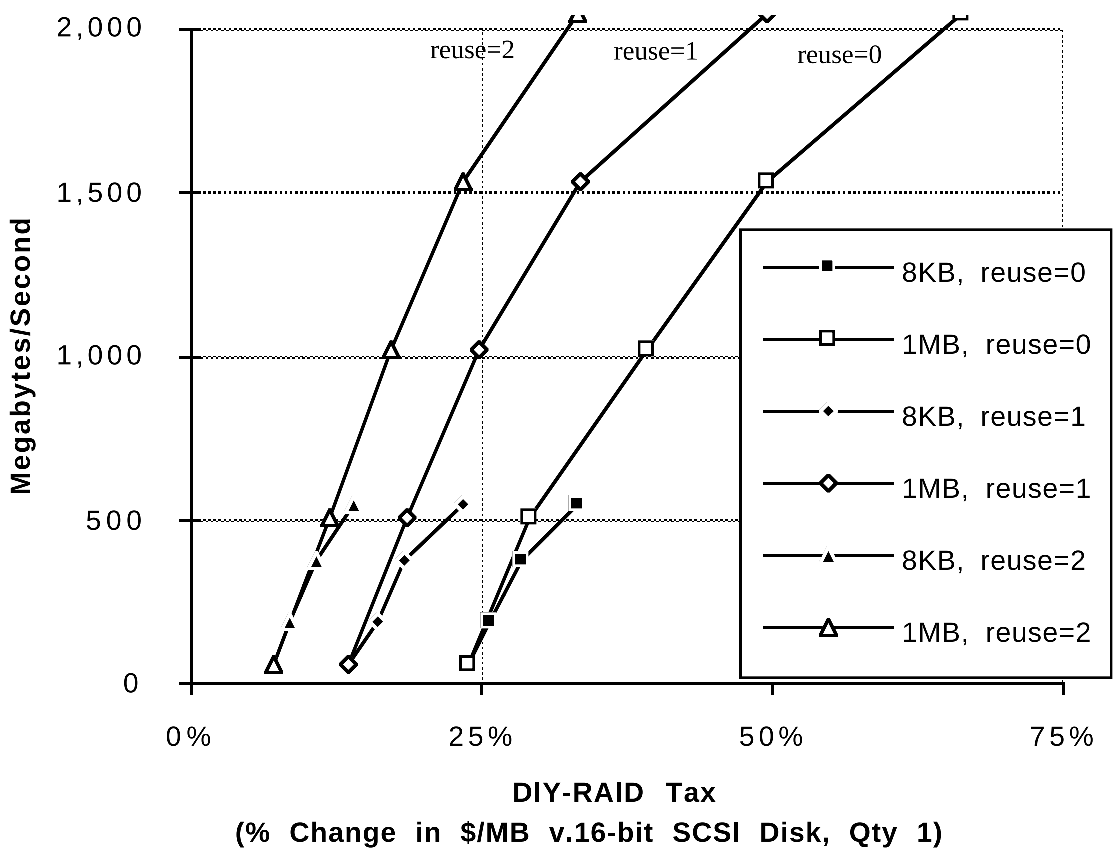

FIGURE 13. Performance in MB/sec vs. relative cost in $/MB for 8KB and 1 MB accesses as infrastructure is re-used. Reuse of 0 means buy everything new each time; reuse of 1 means buying disks, PCs, and adapters when disks are replaced; and reuse of 2 means only buying disks and PCs twice over three disk lifetimes.

disks: strings, disk cannisters, disk enclosures, I/O adapters, PCI extension boxes, network interface cards, switches, PCs, UPS, and racks. Given that disks are improving so rapidly, it seems likely that much of this equipment could be reused for two or even three disk lifetimes. At current trends, in just over three years you could replace 10 disks by 1 equally expensive disk; in five years the total capacity of the old disks is just 3% of the new disks! Another calculation for disk lifetime is based on the 5-year warranty: replace disks once their warranty expires. Thus in 3 to 5 years you might want to replace all disks.

Recycling the enclosures and cannisters presumes that the disk form factor remains the same and that the power supplies in the enclosures are adequate to run the next generation of disks. The 3.5-inch disk form factor seems likely to survive generations because the size is a good match to desktop and server computers; it has been in products since 1987 and does not seem likely to disappear soon. The enclosure power supplies must be sized to cover multiple generations, but the rate of increase in power consumption per disk has been slow and predictable. Fortunately, the cables are reusable even if faster adapters are acquired. The UPS and racks are the simplest to recycle.

For example, RAID-II was constructed in 1989-1990 using 120 IBM 0663 3.5-inch disks, each containing 320 MB[37]. The disks were 8-bit SCSI. Six years later they could be replaced by 4.2 GB, 8-bit, 3.5-inch SCSI disks simply by replacing the disks in the cannisters. Moreover, such drives look to be available for the foreseeable future.

Figure 13 shows cost-performance of two recycling options: 1) upgrade the PCs and I/O adapters as well as the disks and amortize the rest of the equipment over two lifetimes; 2) upgrade just PCs and disks while amortizing over *three* lifetimes. The underlying principle is to replace the items that are changing fastest--PCs and disks--and to reuse the components that change slowly. The argument for replacing the adapter is that it may have less overhead and it may have a faster transfer rate, increasing the bandwidth of the string and controller. Note that the cannisters used to enable just-in-time assembly also simplify recycling the infrastructure.

Depending on which components are recycled and the number of times they are reused, Figure 13 shows that a 1000 disk system is about 25% more per megabyte than a single disk, even at high read rates.

6. Reliability of DIY-RAID Alternatives

Now that we have demonstrated the cost-performance of several alternatives, it’s time to assess the reliability of a prototypical DIY-RAID and the new disk interfaces. In this section we will calculate the reliability of the SCSI DIY-RAID organization in Figure 3 using the techniques of [10][17], and then compare SCSI, FC-AL and SSA reliability characteristics.

We will consider three types of failures; disk failures, SCSI (adapter and string) failures and CPU (system) failures. Disk failures can be masked using RAID 5 (rotated parity) or RAID 6 (P+Q parity). RAID 5 can survive one disk failure per parity group, while RAID 6 can survive two. To mask cable failures, which take down a whole string, we organize the system into an Orthogonal RAID. Here the SCSI strings, enclosures and power supplies are placed orthogonal to the parity groups, enabling the system to survive the failure of any one of these components. Also, the design in Figure 3 has two adapters per string, so that it takes two adapter failures to make a string inaccessible. If the parity groups are also orthogonal to the CPUs, the system can mask the failure of up to 4 CPUs. A single system failure can result in the parity and the disk being inconsistent on writes, however, if the data is updated and the parity is not, or vice versa. If a disk failure occurs before this is corrected, the data cannot be reconstructed [10]. Also, uncorrectable read errors could cause the wrong data to be reconstructed after a disk has failed. The MTTDL (Mean Time To Data Loss) of the system is the harmonic sum of the MTTDL from each of these failures.

Assuming a total of 1120 disks, this system would have 10 racks, each with 2 CPU’s and 112 disks. Each SCSI string will have 8 disks. We will use a group size (G) of 35 for RAID 5 and 70 for RAID 6, so the redundancy overhead is the same for both systems.

The MTTF (Mean Time to Failure) of a disk is assumed to be 250,000 hours. Although the quoted MTTF for disks today is between 800,000 and 1,000,000, in practice it is closer to 250,000 hours due to short product lifetimes of today’s disks and thus the limited time to remove bugs from manufacturing process [11].

The MTTR (Mean Time to Repair) of a disk can be broken up into two parts, the time to purchase a new disk plus the time to reconstruct data on to the new disk. We will call the latter MTTR (disk-recovery). When a hot spare is available, MTTR(disk) actually becomes MTTR(disk-recovery). We consider 16 hot spare disks, whose space is distributed over the array [24], is essentially infinite because we would need three string failures before any were replaced to cause data loss.

The time taken to reconstruct a failed disk is a function of the parity group size G. If we assume each PC reads its disks in parallel, and that parity is computed in a tree of exclusive ORs, and that there are enough disks in a parity group to drive the PCI bus at the maximum practical bandwidth, then at a group size of 35, MTTR(disk-recovery) becomes 0.8 hours for a 4.2 GB disk. Figure 14 gives the values of MTTF and MTTR for the disks as well as the other components.

Bit Error Rate (BER)			1 in 10^14 bits, 1 in 2.4 10^10 sectors
p(disk) The probability of reading all sectors on a disk.			99.96%
RAID 5: MTTF(disk)	250,000 hours	MTTR(rec)	0.8 hours
RAID 6: MTTF(disk)	250,000 hours	MTTR(rec)	1.6 hours
MTTF(string)	120,000 hours	MTTR(string)	0.9 hour
MTTF(sys)	1 month	MTTR(sys)	1 hour

FIGURE 14. Reliability Parameters. We assume 1120 disks, using 512 byte sectors and an 4.3 GB disk cacity.This table lists parameters used for reliability calculations in this section.

		Formula	**MTTDL** (k hours)
#1	Double Disk Failure	$DL = \frac{MTTF(disk)^2}{N \times (G-1) \times MTTR(disk-recovery)}$	2,113
#2	Disk Failure + String Failure	$\frac{\frac{MTTF(disk)^2}{N(G-1)\,MTTR(disk-recovery)}}{(1+\alpha_F) + \left(1+\alpha_F\left(\frac{\phi G}{N}\right)\right)\alpha_F}, \alpha_F = \frac{MTTF(disk)}{MTTF(2Strings)}$	2,049
#3	Sys Crash + Disk Failure	$\frac{MTTF(sys) \times MTTF(disk)}{N \times MTTR(sys)}$	161
	Disk Failure + Bit Error	$\frac{MTTF(disk)}{N \times (1-(p(disk))^{G-1})}$	17
	Total	(harmonic sum of two above)	15

FIGURE 15. Mean time to data loss in hours (MTTDL) for RAID Level 5 (rotated parity) Disk Arrays with 1120 disks and a group size of 34. Option #1 assumes only disk failures. Option #2 includes failures due to loss of a whole string. Option three accounts for system crashes during writes and uncorrectable bit errors during disk recovery.

Figure 15 gives the MTTDL due to each of these failures. When only disk failures are considered, using RAID 5 with a group size of 35 gives an MTTDL of 2.1 million hours or 8 times the reliability of a single disk. When string failures are also considered, the MTTDL drops to 2.0 million. However, the MTTDL due to a system crash plus a disk failure is only 160 thousand hours, or half that of a single disk. Also, the MTTDL due to disk failure followed by a bit error is 17 thousand hours. If all of these failures are taken into account, the MTTDL of the system drops to 15 thousand hours, which is a small fraction of the reliability of a single disk.

Fortunately, RAID 6 gives much better results. A single parity group can survive two disk failures before data is lost. This means that we can use a much larger group size, such as 70, and still get acceptable values for MTTDL. Figure 16 shows these calculations. For disk failures alone, RAID 6 gives an MTTDL of 2.5 billion hours. When string failures are included, the MTTDL can be approximated to be that of an Orthogonal RAID 5, with the failure of one extra disk in the parity group needed for data loss. This gives an MTTDL of 1.9 billion hours. While the chances of losing data due to a system crash and disk failure are not reduced, RAID 6 greatly improves the MTTDL due to disk failure followed by a bit error, which becomes 39,950,997 hours. If we include a non-volatile RAM, an uninterruptible power supply (UPS), or a logging file system so as to remove system crash plus disk failures as a reason to lose data, then the MTTDL for the system becomes 37 million hours, or about 150 times the reliability of a single disk.

		Formula	**MTTDL** (k hours)
#1	Triple Disk Failure	$\frac{\mathrm{MTTF}^3(\mathrm{disk})}{N \times (G-1) \times (G-2) \times \mathrm{MTTR}^2(\mathrm{disk-recovery})}$	2,464,752
#3	Orthogonal RAID 6	$\mathrm{Orthogonal\ RAID5} \times \frac{\mathrm{MTTF}(\mathrm{disk})}{\mathrm{MTTR}(\mathrm{disk-recovery}) \times (G-2)}$	1,911,200
	Sys Crash + Disk Failure	$\frac{\mathrm{MTTF}(\mathrm{sys}) \times \mathrm{MTTF}(\mathrm{disk})^2}{N \times (G-1) \times \mathrm{MTTR}(\mathrm{sys})}$	161
	Double Disk Failure + Bit Error	$\frac{\mathrm{MTTF}(\mathrm{disk}) \times \mathrm{MTTF}(\mathrm{disk})}{(N(G-1))\ (1-p(\mathrm{disk})^{G-2})\ \mathrm{MTTR}(\mathrm{disk-recovery})}$	39,951
	Total	(harmonic sum of prior 3)	160
	Total	(harmonic sum excluding system crash+ disk failure)	36,736

FIGURE 16. Characteristics for a RAID Level 6 (P+Q) disk array with 1120 disks and a group size of 70. Note the results are in thousands of hours.

Failure Models of New Disk Interfaces

The conclusion of the calculations above is that we can construct a thousand disk system with acceptable reliability with SCSI. The question then is, how do FC-AL and SSA differ? Figure 17 shows SCSI, SSA and FC-AL. At the level of the CPU, all three are susceptible to the same failures. They differ in their handling of disk failures, disk enclosure failures, string failures and adapter failures.

In SCSI, the failure of a disk or disk enclosure will not affect the other disks on the string unless the interface fails in a way that it jams the bus. Failure in any part of the SCSI string will make all disks on the string unavailable since the SCSI string requires termination to operate correctly. If an adapter fails without termination failing, the string is still accessible through the other adapter. However if termination fails, the entire string becomes inaccessible.

Both SSA and FC-AL enclosures have bypass circuits so that an individual disk or adapter can be logically disconnected from the chain. This is essential, since these are both active circuits. SSA is shown as a single loop built of point to point bidirectional links. So if a single pair of links fail, all disks are still acces-

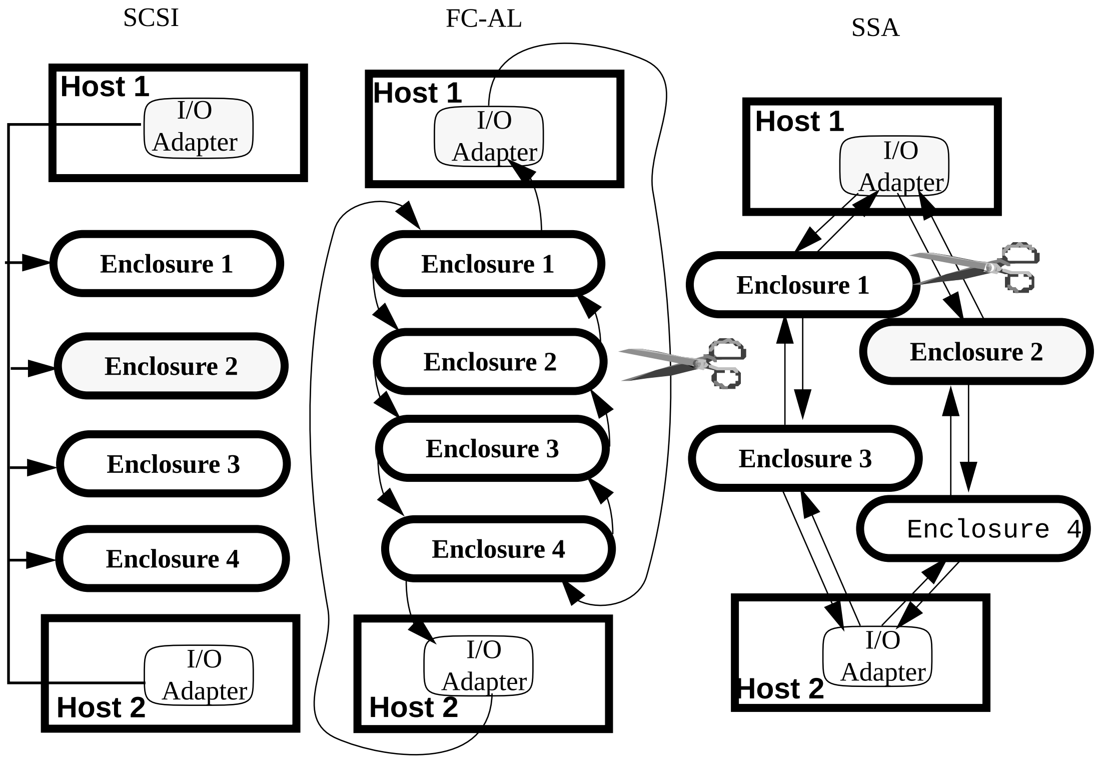

FIGURE 17. Comparison of SCSI, FC-AL, and SSA strings under failure. Shaded areas and scissors suggest single failures that a system can survive. In all three cases, if a I/O Adapter in Host 1 fails then Host 2 takes over, and all disks are still available. SCSI is a passive connection, so if Enclosure 2 fails, Enclosures 1 and 3 still can send data (unless a broken interface jams the SCSI string). If the SCSI cable fails, then all disks are unavailable. FC-AL can be built with two loops so as to survive a string failure, but an enclosure failure will disconnect all disks in the loops since the loops are independent, and FC-AL cannot route around the failed module. (Even though a some adapters have two loops, the FC-AL adapter firmware and the FC-AL standard do not support routing around failed parts[3].) SSA can survive a break of a single segment since there is another path. It can also survive an enclosure failure, because the interface will route signals to avoid the failed module. This example is a single SSA loop; the SSA adapter we used in our cost-performance evaluations assumed two loops. For this investigation, we consider an SSA “string” as all the disks between two adapters.

sible. SSA can survive the failure of any single link, enclosure or adapter.

FC-AL, on the other hand, consists of two unidirectional loops, each connected to one adapter and all the enclosures. It can survive a single adapter or string failure but not an enclosure failure. An enclosure failure will disconnect both loops, as FC-AL is not able to route around the failed enclosure like SSA. As a result, in Figures 9 to 12 we assume FC-AL must use an orthogonal RAID organization (group size ≤ number of strings), while for SSA we can assign multiple disks in the same group to the same string. To get the same functionality in FC-AL, the enclosures must be connected to hubs. At $5,000 for a 16-way hub, these solutions are more expensive than shrinking the parity groups. The practical consequence is that its hard for our application to take advantage of FC-AL's ability to put a large number of disks per string.

A final consideration is the undetectable bit error rate of disks and buses. Given the short product lifetimes and high variability of disks, we may see higher uncorrectable bit errors than given in the production specs. With 1000 disks transferring data, we may want to consider extra protection to make the chances remote of returning incorrect data. For example, the file system might include a correcting code at the end of each file to protect data against normally very rare events. Given the end-to-end argument—instead of calculating checksums across networks and again across internal busses and again across disks—do it once. It means we can avoid calculating checksums over unreliable networks on reads, although we still would have to calculate on writes to avoid writing incorrect data. Such a scheme might be also help with the RAID problem of an uncorrectable error during a reconstruction. If the file system were going to compress the data to improve network performance and storage efficiency, such a check sum could be included at little extra cost.

7. DIY-RAID vs. Hardware RAIDs and Tape Libraries

The speed of DIY-RAID makes it competitive with hardware RAIDs and the capacity of DIY-RAID makes it competitive with tape libraries. Figure 18 compares the cost/megabyte and 8-KB reads/second of three hardware-RAIDs, three tapes libraries, and a small, medium, and large DIY-RAIDs.

The performance model of the tape libraries assumes a optimistic time of 30 seconds to switch tapes and find a file; most libraries take much longer. DIY-RAIDs cost about 2.5 to 7.0 times as much per megabyte and are about 200,000 to 1,000,000 times faster than tape libraries for 8 KB reads. Our investigation suggests that tape libraries have improved in cost/megabyte by 25% per year for the last several years, while disks have improved at 100% per year over the same period. If these trends continue, DIY-RAIDs would be less expensive than tape libraries in 2 to 4 years.

The cost per megabyte of hardware RAIDs is about 2.5 to 7.3 times higher than DIY-RAIDs. (The I in RAID stands for Inexpensive, but that term does not apply to the hardware RAID systems.) DIY-RAIDs are about the same speed to 30 times faster than hardware RAIDs for 8 KB reads, depending on DIY-RAID size.

The reliability of tape libraries is considerably worse, typically with a MTTF of 50,000 hours. Hardware RAID companies do not quote reliability, but we assume that they are a small multiple of a single disk. There are two key points about hardware RAIDs: 1) they typically connect to a single computer, which if you look a the whole system means a low MTTF, and 2) they aren't that large, which means you need many of them to get high capacity. Alas, reliability is inversely propositional of the number of systems: 50 systems have 1/50th the reliability of a single system. Hence DIY-RAIDs are roughly 750 times more reliable than tape libraries and, depending on the reliability of a single unit and the number of them, as reliable or more than hardware RAIDs. We could apply DIY-RAID techniques to stripe across hardware RAIDs to get better reliability and performance, but they will be much more expensive than a DIY-RAID.

The individual PC disk is the right building block for large scale storage systems.

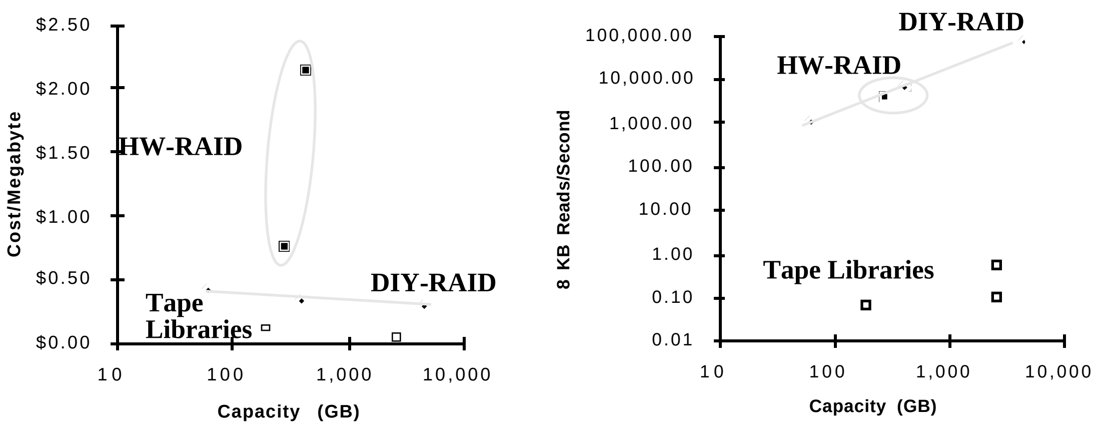

FIGURE 18. Cost/Megabyte and 8 KB reads per second vs. versus capacity for DIY-RAIDs, tape libraries, and commercial RAIDs. For tape libraries we used STK 9704, DEC TL820, and ATL ACL 264. For hardware RAIDs we used DEC StorageWorks 410 and Maximum Strategy RAID Gen 4 XL Storage Server. The disk speeds were assumed to be the same as in DYI-RAIDs in Section 4 and the parity group size was set at 8 and 12, respectively. The performance is based on the number of drives and the speed of the bus, assuming a limit of 80% per bus. The DIY-RAIDs were the 16 disk, 112 disk, and 1120 disk systems described above. We assume a reuse of one when determining cost per megabyte.

8. Expanding the Definition of Computer Architecture

As mentioned in the introduction, DIY-RAIDs and NOWs may be forerunners of an new branch of computer architecture research. It would seem that a new branch of computer architecture must have an interesting design space and possibly new metrics to evaluate success.

The design space is large for DIY-RAIDs. We explored low-cost and the low-end of cost-performance designs for a read-mostly workload. The tradeoffs might be much different for systems targeted at different workloads; systems oriented towards low latency rather than high bandwidth; systems whose performance did not degrade during a failure; ultra reliable systems; and systems where the disks are always the performance limit.

A major figure of merit for this style of architecture is cost. For high volume components, price is probably a better predictor of cost than more indirect measures like number of chips or memory size. The shrinking size of systems and low margins probably make it easier to estimate costs than at any previous time in computer history. We can even find the prices over time if the components are used in a PC, allowing discovery of historical trends, as in Figure 1.

Our results would have been very different if we had ignored cost. For example, had we not looked at prices, we would not have discovered that the cost-improvement rate exceeded areal-density improvement rate for the last 5 years. Given the fast rate, we can find explanations in higher disk volumes and shrinking electronics. The rate in turn determined the importance of just-in-time installation of disks and an infrastructure that allows replacement of the fast changing technologies while amortizing the costs of the rest. Finally, when cost is included, surprisingly SSA and FC-AL are no more attractive than the traditional interface, SC-

SI. Reasons for higher prices include the large number of ports and higher speed interfaces; without looking at costs, complexity is always better.

A second metric for this new branch of architecture is reliability, again often ignored in traditional computer architecture research. Perhaps because the reliability of the building blocks is well documented it is easier to calculate the reliability of the resulting system, which in turn leads reliability to be addressed in the design.

Looking at both metrics can reveal complex relationships in an implementation. For DIY-RAID, there is an interesting dependency between number of disks, capacity of disk enclosures, parity group size, and string technology. Focussing on low cost encourages large numbers of disks per string—and thus fewer strings— and large parity groups to reduce the number of redundant disks. But orthogonal RAIDs require that the number of strings be a multiple of the parity group size. For FC-AL the balance is between the useful data capacity, which is shrunk with smaller groups, versus the extra adapters and PCI enclosures to accommodate the extra strings. For the costs and number of disks assumed here, the lowest cost is at 64 disks per string or 18 strings with a parity group size of 18.

Let's put this expansion of architecture research in context. In the 1950s computer architecture consisted primarily of research in ALUs. In the 1960s it moved to design of the instruction set. In the 1980s attention was paid to the implementation of the processor, and for the last 10 years research has focused on the memory system as well as the processor. Following our tradition of expanding the field as new opportunities arise, perhaps it is time to expand the research agenda of computer architecture to include construction of interesting large systems where the building block itself is much larger and more self-contained. This expansion may even enhance traditional topics, offering new challenges and new figures of merit.

9. Conclusions

The thesis of this paper is proved by example: disk storage systems can be designed that scale in capacity and performance from 10 disks to 1000 disks while maintaining the cost/capacity and reliability of a single disk. Moreover, these systems are designed to better match the performance demands of switched local area networks that offer higher performance via multiple network interfaces. DIY-RAIDs may be able to scale up to 10,000 disks provided there was 250 square feet of floor space and 175 kilowatts of power.

These results are based on variations of the DIY-RAID architecture in Figure 3. Assuming a fixed number of disks, fixed group size and access size, we identified two parameters that predict cost/performance: number of PCs and number of disks per string. Cost/reliability is determined by two other parameters: number of spare disks and parity group size.

The architecture of DIY-RAID is made fault tolerant by placing multiple adapters and PCs on a string. A different architecture could be made if the bus could be made fault tolerant: multiple CPUs on a PCI bus. This architecture would halve the adapters per string. An even lower cost version would come from keeping the prices of SCSI but offer strings that allow many more disks. Surprisingly, the multiple loop approach of SSA is not required because orthogonal RAID can also overcome such reliability weaknesses of dependent failures.

Our future work on DIY-RAID includes examining more realistic workloads, the impact on cost and reliability of powering down idle disks, examining even larger DIY-RAIDs, and construction of prototypes. Our prototype plans are for a 30 disk prototype by January 1996 and a 100 disk prototype in by Spring of 1996. Pending evaluation of these prototypes and sponsorship of disk vendors, the 1000 disk system would follow. As mentioned before, we hope the WWW application will give us a more realistic workload to evaluate performance.

To evaluate cost/performance we addressed cost as well as performance. The components in DIY-RAIDs, as well as the hardware RAID and tape library competition, are extraordinarily diverse. It is difficult to imagine some other accurate cost function that would allow tradeoffs between bus interfaces, network interfaces, rotating magnetic media, sequential magnetic media, robot arms, cables, sheet meal, power supplies, fans, and integrated circuits. Fortunately, today relative prices predict relative cost for high volume components, so we use this indirect measure to compare. One conclusion from our example is that DIY-RAID infrastructure cost is dominated by the mechanical devices and passive electronics: cannisters, enclosures, cables, UPS, and racks. Thus lowering such costs must come either from amortizing over multiple disk lifetimes or from innovations in fields other than computer architecture.

Some designers are considering having network interfaces directly on the disks as the building block for DIY-RAIDs. A key question will be the extra cost of network disks and the switches to connect them versus the standard disks and the PCs we used. The overhead for PCs, adapters, and network interfaces—in other words, the components that aren't in common with network-attached disks—are roughly 13% of the costs of a large scale DIY-RAID, and its hard to imagine switched networks and interfaces for 1000 disks that cost that little. It is even harder to imagine how to take advantage of reuse when the network is part of the disk. Finally, network-attached disks may also need to deal with security issues.

As part of this study we also got new insights into upcoming serial standards proposed to follow SCSI. We fully expected that the results of the paper would show that SSA or FC-AL would be much better than SCSI, and were surprised how well SCSI fared in comparison. The fault characteristics of FC-AL mean that the cost savings for a large number of disks on the string are more than lost as a result of the shrinking of the parity group size. The FC-AL string bandwidth is high, but high performance is limited by the number of PCs; since we must have one string per pair of PCs, so the number of PCs sets an upper limit on disks per string no matter how fast the strings are. SSA is a better match to the DIY-RAID architecture, but even so, no better than 16-bit SCSI.

The low costs for high volume components versus attractive features of new standards is a continuing dilemma. Given we have assumed a best case for SSA and FC-AL—small differences in cost per disks—we expect SCSI to remain attractive for DIY-RAID applications for several years.

This paper suggests a DIY-RAID can be constructed in 1995 with cost only 2.5 to 7 times the cost of a tape library for the same capacity, at the cost of a larger footprint and more power. Such systems would provide a hundredfold improvement in reliability, a hundredfold to thousandfold improvement in the bandwidth and at least a hundred-thousandfold improvement in latency. If DIY-RAIDs can track the relative improvements in cost/megabyte of disks over tape libraries, then we may see "tertiary disk libraries" replace "tertiary tape libraries" within the next five years.

10. Acknowledgments and References

We wish to thank Jim Gray of Microsoft for the initial suggestion to investigate DIY-RAIDs, Ed Frank of Net-Power for ideas on PCI extension buses, Bob Selinger of Adaptec for SSA information, Dave Anderson of Seagate for FC-AL information, and reviewers of early drafts: Jim Brady of IBM, Pete Chen of Michigan, Mike Dahlin of U.C.Berkeley, Garth Gibson of CMU, Randy Katz of U.C.Berkeley, Ed Lee of DEC, John Ousterhout of Sun Microsystems, and Dick Sites of DEC. This research was performed in conjunction with the NOW project and is funded by the ARPA Robo-line Storage contract N00600-93-C-2481.

[1] Serial Storage Architecture ANSI X3T10.1, 1995.

[2] FibreChannel Arbitrated Loop ANSI X3T9.3, 1995.

[3] David Anderson. Personal communication, Sept. 1995.

[4] T. Anderson, D. Culler, D. Patterson, and the NOW team. A Case for NOW (Networks of Workstations). *IEEE Micro*, pages 54–64, February 1995.

[5] ATM Forum. *The ATM Forum User-Network Interface Specification, version 3.0*. Prentice Hall Intl., New Jersey, 1993.

[6] N. Boden, D. Cohen, R. Felderman, A. Kulawik, C. Seitz, J. Seizovic, and W. Su. Myrinet – A Gigabit-per-Second Local-Area Network. *IEEE Micro*, pages 29–36, February 1995.

[7] Jim Brady. Personal Communication, October 1995.

[8] L. Cabrera and D. Long. Swift: A Storage Architecture for Large Objects. In *Eleventh Symposium on Mass Storage Systems*, pages 123–128, 1991.

[9] P. Cao, S. Lim, S. Venkataraman, and J. Wilkes. The TickerTAIP Parallel RAID Architecture. In *Proc. of the 20th Symp. on Computer Architecture*, pages 52–63, May 1993.

[10] P. Chen, E. Lee, G. Gibson, R. Katz, and D. Patterson. RAID: High-Performance, Reliable Secondary Storage. *ACM Computing Surveys*, 26(2):145–188, June 1994.

[11] R. Clewett. Personal Communication, 1995.

[12] Kevin Fall. Personal Communication, 1995.

[13] E. Frank. Personal Communication, 1995.

[14] S.H. Fuller. Price/Performance comparison of C.mmp and the PDP-ll. In *3rd Annual Symposium on Computer Architecture*, January 1976.

[15] Greg Ganger. Personal Communication, October 1995.

[16] Garth Gibson. Personal Communication, 1995.

[17] Garth Alan Gibson. *Redundant Disk Arrays: Reliable Parallel Secondary Storage*. Ph.D. thesis, University of California at Berkeley, April 1990.

[18] Gregory Pfister. *In Search of Clusters*. Prentice Hall Intl., New Jersey, 1995.

[19] J. Hartman. *The Zebra Striped Network File System*. Ph.D. thesis, University of California at Berkeley, 1994.

[20] J. Hartman and J. Ousterhout. The Zebra Striped Network File System. *ACM Trans. on Computer Systems*, August 1995.

[21] J. Hennessy and D. Patterson. *Computer Architecture A Quantitative Approach 2nd Ed.* Morgan Kaufmann Publishers, Inc., 1995.

[22] E. Lee. Highly-Available, Scalable Network Storage. In *Proc. of COMPCON 95*, 1995.

[23] R. Martin. HPAM: An Active Message Layer for a Network of HP Workstations. In *Proc. 1994 Hot Interconnects*, August 1994.

[24] R. Muntz and J.C. Lui. Performance Analysis of Disk Arrays under Failure. In *16th Conference on Very Large Data Bases*, 1990.

[25] Jussi Myllymaki. RAID Technologies. http://www.cs.wisc.edu/ jussi/raid.html, May 1994.

[26] Mark Porter and Bill Bailey. The Design and Implementation of a Large-Scale Real-Time Media Server. In *Proc. 1995 Symposium on Hot Interconnects*, August 1995.

[27] R. Rashid. Microsoft's Tiger Media Server. In *The First Networks of Workstations Workshop Record*, October 1994.

[28] M. Rosenblum and J. Ousterhout. The Design and Implementation of a Log-Structured File System. In *Proc. of the 13th Symp. on Operating Systems Principles*, pages 1–15, October 1991.

[29] Bob Selinger. Personal Communication, 1995.

[30] T. von Eicken, A. Basu, and V. Buch. Low-Latency Communication Over ATM Networks Using Active Messages. *IEEE Micro*, pages 46–53, February 1995.

[31] T. Anderson, M. Dahlin, j. Neefe, D. Patterson, D. Roselli, and R. Wang. Serverless Network Files Systems. In *Proc. 15th*

Symp. on Operating Systems Principles, pages 71–78, December 1995.

[32] J. Wilkes, R. Golding, C. Staelin, and T. Sullivan. The HP AutoRAID Hierarchical Storage System. In *Proc. of the 15th Symp. on Operating Systems Principles*, December 1995.

[33] Robin Williams. Interactive Video Servers. In *Proc. 1995 Symposium on Hot Interconnects*, August 1995.

[34] D.A. Wood and M.D. Hill. Cost-Effective parallel computing. *Computer*, pages 69–72, February 1995.

[35] Bruce Worthington. Personal Communication, October 1995.

[36] Drapeau, A.L.; Shirriff, K.W.; Hartman, J.H.; Miller, E.L.; and others. RAID-II: a high-bandwidth network file server. In *Proc. the 21st Annual International Symposium on Computer Architecture*, Chicago, IL, USA, 18-21 April 1994.