Performance Comparison of IDE and SCSI Disks

Brian White
University of Virginia
bsw9d@cs.virginia.edu

Wee Teck Ng, Bruce K. Hillyer
Bell Laboratories, Lucent Technologies
{weeteck, bruce}@research.bell-labs.com

Abstract:

It is widely believed that the IDE disks found in PCs are inexpensive but slow, whereas the SCSI disks used in servers and workstations are faster, more reliable, and more manageable. The belief that current IDE disks have performance and reliability disadvantages has been called into question by several recent reports. Thus we consider the possibility of achieving tremendous cost advantages by using IDE disks as the foundation of a storage system.

In this paper, we give an extensive performance comparison of IDE and SCSI disks. We measure their performance on a variety of micro benchmarks and macro benchmarks, and we explain these results with the help of kernel instrumentation and device activity traces collected by a SCSI analyzer. We consider the impact of several factors, including sequential vs. random workloads, file system enhancements such as journaling and Soft Updates, I/O scheduling in the kernel vs. in the disk drive (as enabled by tagged queuing), and the use of RAID technology to obtain I/O parallelism. In our testbed we find that the IDE disk is faster than the SCSI disk for sequential I/O, but the SCSI disk is faster for random I/O. We also observe that the random I/O performance deficit of the IDE disk is partly overcome by kernel I/O scheduling, and is further mitigated by scheduling in the drive (as enabled by tagged queuing), and by the use of journaling and Soft Updates. Taken as a whole, our results lead us to conclude that RAID systems based on IDE drives can be both faster and significantly less expensive than SCSI RAID systems.

1 Introduction

In this paper, we focus on a performance comparison of two popular hard disks types, IDE and SCSI. IDE (Integrated Device Electronics) disks, also referred to in the industry as ATA (AT attachment) or EIDE (Enhanced IDE), are the dominant storage for home computers and personal workstations, and have 85% share of the disk drive market [25]. By contrast, SCSI (Small Computer System Interconnect) disks dominate the workstation and server markets. IDE disks are much less expensive than SCSI disks: Chung et al. [24] note the potential to store a terabyte using IDE disks for less than $10,000. To capitalize on this cost advantage, companies such as Adaptec, Promise, Raidzone, Raidweb, Jetstor, and disk manufacturers Maxtor and Quantum have introduced RAID IDE controllers [1, 11, 23] and IDE RAID systems.

The desire to provide more cost-effective storage solutions, on various scales, motivates a thorough comparison of the IDE and SCSI technologies. We distinguish our work in this area from previous efforts (see Section 5) by combining the following properties in this study. Our experimental testbed is specified in detail, the device specifications of our IDE and SCSI configurations are very similar, our performance experiments are comprehensive and clearly defined, the experiments represent a wide-range of configurations and workloads, and the analysis of benchmark timing measurements is supported by kernel instrumentation and by physical I/O traces obtained via a SCSI analyzer. In addition to experimental analysis, we examine four factors that partly compensate for the native performance limitations of IDE, namely disk concurrency, disk scheduling, file system design, and I/O parallelism.

This paper is organized as follows. Section 2 describes differences between IDE and SCSI disks in terms of price, performance-related specifications, and interface-related specifications. Section 3 describes our experimental testbed and the benchmarks that we use for timing measurements. Section 4 presents the benchmark results and analysis. Section 5 covers related work, and in Section 6 we provide concluding remarks.

Samsung Ex. 1202, Page 1 of 14

2 IDE versus SCSI

In this section, we compare IDE and SCSI technology in terms of price, performance specifications, and interface specifications, because these considerations are important in assessing the relative utility of these two technology families. To give representative device characteristics, we display specifications for six current disks and controllers in Table 1. In Table 2 we show several interface characteristics that differentiate between IDE and SCSI devices (see [25] for a more detailed comparison of these interfaces).

	Model	Rotational Speed	Capacity	Avg. Seek Read/Write	Sustained Bandwidth	Buffer Size	Cost	Cost/ GB
IDE Disks	IBM 75GXP	7200 RPM	30 GB	7 ms	37 MB/s	2 MB	$159	$5.30
	Seagate Barracuda ATA II	7200 RPM	30 GB	8.9 ms	30 MB/s	2 MB	$165	$5.50
	Maxtor DM Plus40	7200 RPM	30 GB	8.7 ms	49.5 MB/s	2 MB	$155	$5.17
SCSI Disks	IBM Ultrastar 36LZX	10000 RPM	36 GB	4.9/5.9 ms	22-36 MB/s	4 MB	$550	$15.11
	Seagate Cheetah 36LP	10000 RPM	36 GB	5.2/6.0 ms	38.5 MB/s	4 MB	$633	$17.39
	Quantum Atlas 10K II	10000 RPM	36 GB	5.5 ms	18-26 MB/s	8 MB	$815	$22.39
	Model	**RAID Supported**	**Max. Device**	**Max. Bandwidth**	**Sustained Bandwidth**	**Cache**	**Cost**	**Cost/ Device**
IDE Controller	Promise Ultra100	None	4	200 MB/s		None	$25	$6.25
	Abit Hotrod100	0, 1, 0/1	4	200 MB/s		None	$45	$11.25
	3ware 6800	0, 1, 1/0	8	528 MB/s	84-100 MB/s	None	$500	$62.50
SCSI Controller	Adaptec 2940U2W	None	15	160 MB/s		None	$250	$16.67
	Adaptec 2200S	0, 1, 0/1, 5	15	160 MB/s		32	$499	$33.27
	Adaptec 3200S	0, 1, 0/1, 5	30	320 MB/s		32	$825	$27.50

Table 1: Disk and Controller Price Comparison. Price figures are from **www.dirtcheapdrives.com** and **www.buy.com** 22 November 2000.

From Table 1 we see that the price differential between SCSI and IDE is enormous. The lower price of IDE devices reflects ferocious competition and economies of scale. On balance, the IDE disks in the table have higher sustained bandwidths, but we see faster seek times and rotational speeds, and larger buffer sizes for SCSI disks. These differences largely reflect market forces rather than distinctions inherent to the technologies.

		IDE	SCSI
Structure		Interface to disks, CD-ROM	Bus supporting many types of devices
Controller		On-board integrated controller	SCSI host bus adapter
Width		16-bit	8-bit, 16-bit, 32-bit
Bus Bandwidth		3.3-100 MB/sec	5-160 MB/sec
Bus Bandwidth Roadmap	**Current**	100 MB/sec	160 MB/sec
	2001	150 MB/sec	320 MB/sec
	2004	300 MB/sec	640 MB/sec
	2007	600 MB/sec	?
Cable Length		46 cm (18 in) Supports only internal devices	1.5-12 meters (12 meters with differential SCSI). Requires termination
Number of Devices		2 per channel	15 per channel (wide SCSI)
Data Transfer Modes		PIO or DMA	DMA
Command Concurrency		Immature Tagged Queuing (Max 32 tags)	Tagged Queuing (Max 256 tags)
Bus Concurrency		Limited disconnect/reconnect	Disconnect/reconnect allows concurrent bus access
Manageability		S.M.A.R.T.	S.M.A.R.T., SCAM

Table 2: Overview of IDE and SCSI Interface. The IDE features are from the ATA/ATAPI-5 standards (T13/T1321D Rev 2). The SCSI features are from the SCSI-2 and SCSI-3 standards (ANSI X3.131, NCITS.306, x3.270, x3.292, x3.301). Some features (e.g. S.M.A.R.T) are specific to the disk drive vendors and are not part of the standards [6, 35]. The future bus speed projections are from [5, 10].

From Table 2 we see several characteristics in the SCSI specifications that could lead to higher performance. SCSI has higher bus frequencies, wider bus widths, greater I/O concurrency (up to 256 outstanding requests), and more extensive disconnect/reconnect handling to reduce bus idle time. In addition, SCSI supports independent transfer rates for a mixture of slow and fast devices, whereas IDE runs all devices on the bus at the "least common denominator" i.e., the slowest bandwidth and data transfer mode of any device connected to the bus.

The performance experiments in this paper show that factors such as disk concurrency, disk scheduling, file system design, and I/O parallelism can cause the observed system performance to differ significantly from what is suggested by a cursory examination of the disk drive specifications. For example, on a software development benchmark (SDET), the performance difference between IDE and SCSI disks is 22% using the default Unix Fast File System (FFS), but only 1% using the Soft Update enhancement to FFS.

SCSI systems generally are considered to be more robust, manageable, and scalable than IDE systems, but many of the perceived reliability problems of IDE systems stem from early implementations that had defects in chipsets and device drivers [7], or are the consequence of improper cooling and operation [34]. Current IDE and SCSI disks should have similar reliability since they share common technology and components [1]. Moreover, storage vendors that sell both IDE and SCSI RAID systems (and thus have no vested interest in either product) note no major differences in disk failure rates between the two types of disks within the disk warranty period [16]. A SCSI disk typically has a 5 year warranty versus 3 years for an IDE disk, but rapid advances in disk technology render disks obsolescent before the expiration of the warranty period [22]. In terms of manageability, SCSI provides a richer command set that enables a user to query and configure device characteristics, and to control recovery from errors. Moreover, the SCSI physical interface supports a larger number of devices per bus and longer cables. RAID boxes that are constructed from IDE devices can obtain many of the advantages of the SCSI interface by exporting a SCSI interface to the host.

3 Measurement Methodology

This section describes our testbed for performance measurements, and the benchmark programs that we use to generate storage workloads.

3.1 Testbed

Our platform is a Dell OptiPlex PC equipped with an Intel D820LP motherboard, an Intel 1 GHz Pentium III processor, 512 MB RDRAM, and a 30 GB IDE system disk connected to the on-board IDE controller. For the IDE experiments, we use IBM Deskstar 75GXP disks, an Abit HotRod IDE controller with HPT 370 chipset, and a Raidweb (www.raidweb.com) IDE RAID array. For the SCSI experiments we use IBM Ultrastar 36ZX disks, an Adaptec 29160 host bus adapter (HBA), and a TSR-2200 (www.terasolutions.com) SCSI RAID array. Table 3 shows the detailed disk specifications. The IDE and SCSI disk settings are mostly comparable: we enable the read and write cache on both IDE and SCSI disks, and both disks support readahead. We enable disconnects and queue management (DQue and Queue Algorithm Modifier) [35] on the SCSI disk, but we are unable to ensure similar settings on the IDE disk as these parameters are not exposed by the IDE interface. The SCSI and IDE RAID arrays are comparable: 128 MB of SDRAM, 8 disks (IBM Deskstar75GXP or IBM Ultrastar36ZX) configured as RAID 5, and an Intel i960 processor. IBM disks are advantageous for experimentation because of detailed documentation for configuration parameters (e.g. Queue Algorithm Modifier) and internal measurement features (e.g. cache statistics). To confirm the generality of our findings, we repeat selected tests on Seagate ST318203LW and Maxtor DiamondMax Plus 40 disks, and on Tekram DC390U2W and Promise Ultra100 host adapters. We use a Verisys SV8160 SCSI bus analyzer (www.verisys.com) to capture I/O traces from the SCSI bus.

The operating systems in our experiments are FreeBSD 5.0–20001108 and Windows NT 4.0 build 1381. To control disk concurrency (i.e., enable tag queuing and set number of tags), in FreeBSD we use the *camcontrol* command, and in Windows NT we use the *Registry Editor*. In FreeBSD experiments we control disk scheduling by marking requests with the *BIO_ORDERED* attribute to restrict request reordering. We cannot modify the disk scheduler behavior in Windows NT (we do not have the source code).

Samsung Ex. 1202, Page 3 of 14

	IDE	SCSI
Model	IBM Deskstar 75GXP	IBM Ultrastar36ZX
Capacity	30 GB	18 GB
RPM	7200	10000
Average Seek Time (read/write)	7.0 ms/8.0 ms	4.9 ms/5.9 ms
Interface	ATA-100	SCSI Ultra2 (Wide) LVD
Interface Bandwidth	100 MB/sec	80 MB/sec
Media Sustained	37.0 MB/sec	29.5 MB/sec
Predictive Failure Analysis	S.M.A.R.T. compliant	S.M.A.R.T. compliant
Buffer Size	2 MB	2MB

Table 3: Disk Specifications. All parameters are extracted from the respective disk drive manuals [6, 35].

3.2 Benchmarks

We choose our benchmarks to represent general-purpose computing and internet service environments. The goal of our evaluation is twofold. First, we seek to understand how IDE and SCSI perform under "realistic" I/O-intensive workloads. Second, we want to understand the impact of file system and storage system parameters on IDE and SCSI performance.

We use two types of benchmarks. Our micro benchmarks quantify the performance of basic I/O operations like reads and writes. Our macro benchmarks measure general purpose programming workloads (SSH and SPEC SDET) and internet workloads (PostMark and NetNews). We run all macro benchmarks on a newly created file system to eliminate the effects of file system aging [29]. For the micro benchmarks, we conduct 10 experiments for each data point. Due to the duration of the macro benchmarks, we were only able to run 2-3 iterations of each experiment, but we observed consistent results across these runs. We computed standard deviations and investigated or re-tested anomalies. We verified that the durations of the macro benchmarks and sizes of the micro benchmarks are sufficient to prevent hitting in the buffer cache and disk cache across experiments.

We begin with a set of **micro benchmarks** that read or write 10 MB from the disk under test, through the raw disk device to avoid the impact of the file system buffer cache [19]. The test forks off 128 children that each issue an 8KB I/O request, resulting in 128 outstanding I/O requests. We use semaphores and shared memory to coordinate the I/Os, and we verify that all 128 children are active concurrently via instrumentation and SCSI bus analysis. The *sequential* program accesses data stored sequentially. The *random* program accesses data stored at random addresses. The *interleave* test accesses data sequentially from two disk locations: one sequential stream starts at block zero, another sequential stream starts at the middle of the disk. The I/O requests from these two sequential streams are interleaved. This access pattern is common in some file systems like Microsoft Windows NTFS, which maintains log files at two disk locations (i.e. MFT and MFTmirror [30]).

The **SSH benchmark** is proposed by Seltzer [27] as a replacement for the popular Andrew File System benchmark. It unpacks, configures, and builds a medium-sized software package (SSH). The unpack phase extracts a compressed tar archive containing the SSH source tree. It thus reads a large file sequentially and generates many subsequent small metadata operations. The config phase determines what features are available on the host operating system and generates makefiles. To do this, it compiles and executes many small test programs, causing many file system metadata operations. The build phase compiles and links the ssh executable. We run the three phases of the benchmark consecutively, consequently the config and build phases run with the file system cache warmed by the previous phases.

The **SPEC SDET benchmark** [8] is designed to emulate a typical timesharing workload. It models a software development environment (e.g., editing, compilation, and various UNIX utilities) on a time-sharing host, and makes fairly extensive use of the file system. Although this benchmark models a time-sharing environment rather than today's client-server environment [27], it is nonetheless useful because it has tunable concurrency, and its basic operations are still representative of the software development workload.

Samsung Ex. 1202, Page 4 of 14

The **PostMark benchmark** is designed to model the workload seen by Internet Service Providers under heavy load [13]. It is meant to model a combination of electronic mail, netnews, and web-based commerce transactions. It creates a large set of files with random sizes within a set range. The files are then subjected to a number of transactions like file create, delete, read and append. We run the benchmark with 25,000 files, 100 subdirectories, and 20K transactions, which results in a data size of about 400 MB.

The **NetNews benchmark** is developed by Karl Swartz to simulate the workload of a NetNews server [33]. It creates a large number of files representing news articles, and deletes old articles by replaying traces that were collected from a live server. This benchmark differs from the other benchmarks in that it has very little locality of reference, and its large data set (2 GB) overwhelms the file system buffer cache [27].

4 Performance Results

We first compare the IDE drive with the SCSI drive on various benchmarks. Our results indicate that IDE is faster on a sequential workload, but not on a random workload. We then analyze the results and study four factors that affect performance: concurrency (tagged queuing), scheduling, file system features, and parallelism (with multiple disks). We find that although IDE disks are slower on the random workload, the performance deficiencies can be mitigated with judicious system design choices.

4.1 Single Disk Performance

Table 4 compares the performance of IDE and SCSI disks on the various benchmarks, and computes the percentage differences in the last column. We use the Unix fast file system (FFS) [18] for the application benchmarks. In this table we use a tag setting of 32 in the driver.

Benchmarks		**IDE**	**SCSI**	**% Perf. Diff**
Read Micro benchmark	Sequential	31.6 MB/sec	15.8 MB/sec	-100%
	Interleave	23.5 MB/sec	17.4 MB/sec	-35%
	Random	1.5 MB/sec	2.5 MB/sec	40%
Write Micro benchmark	Sequential	27.1 MB/sec	19.2 MB/sec	-41%
	Interleave	19.2 MB/sec	16.8 MB/sec	-14%
	Random	1.4 MB/sec	2.1 MB/sec	33%
SDET	1 job	10.1 sec	12.3 sec	-18%
	5 jobs	78.2 sec	63.8 sec	23%
	10 jobs	178.5 sec	129.4 sec	38%
	20 jobs	377.1 sec	283.0 sec	33%
SSH		88.5 sec	74.9 sec	18%
PostMark		502 sec	387 sec	30%
NetNews		1100 sec	809 sec	36%

Table 4: SCSI versus IDE Performance.

The micro benchmark results indicate that the IDE disk is about 33% and 40% slower on random writes and reads, respectively. This is consistent with the disk specifications in Table 3: a small random I/O operation is dominated by the seek time [35, 6], and the random I/O performance difference between the IDE and SCSI disks is similar to the ratio of disk seek times (36% and 43% for read and write, respectively). The IDE disk is significantly faster (40-100%) than the SCSI disk on benchmark workloads with highly sequential access patterns: this result is also consistent with Table 3.

On the application benchmarks, the SCSI disk is 18% (SSH) to 36% (NetNews) faster than the IDE disk. Since these benchmarks have a large number of synchronous writes to random addresses, and these experiments use the Unix FFS file system [27], the faster performance of the SCSI disk is consistent with its faster seek time. Because the SSH benchmark is less I/O bound, it shows a smaller advantage for SCSI.

4.2 Factors Affecting Disk Performance

In Section 4.1 we have seen that the IDE disk performs well for sequential access, but not as well as the SCSI disk for random access. We now investigate the effectiveness of four approaches that attempt to improve the random-access performance of the IDE drive. In brief, these approaches are as follows.

In Section 4.2.1 we examine how the random-access performance varies depending on the degree of concurrency (tagged queuing). I/O concurrency enables the drive to reorder requests to reduce the seek time. Our results indicate that a tag size of 4 appears adequate for a *single* disk for these application benchmarks, so IDE's maximum tag limitation of 32 (versus 256 for SCSI) does not impair performance.

In Section 4.2.2 we investigate whether I/O scheduling in the host is an adequate substitute for tagged queuing and scheduling in the disk, as suggested by prior disk scheduling studies [26, 12]. If we compare ideal conditions for the host-based elevator scheduling (thousands of dirty pages in the buffer pool) with ideal conditions for disk scheduling (tagged queuing with concurrency > 32), the random I/O micro benchmark indicates that disk scheduling is significantly more effective than host scheduling. But the application benchmarks show little performance difference between host-based and disk-based I/O scheduling—these benchmarks do not generate sufficiently high concurrency to favor either scheduler.

In Section 4.2.3 we evaluate the impact of two file system enhancements, Soft Updates (which reduces synchronous metadata writes by careful write ordering) and Journal file system (which performs sequential log writes for metadata updates). The results indicate that, when running benchmarks on the Soft Updates or Journal file systems instead of FFS, the IDE and SCSI drives have similar performance (under certain conditions described in Section 4.2.3). We use a novel low-level SCSI trace approach to explain the qualifying conditions.

In Section 4.2.4 we explore using disk arrays [21] with random I/O workloads. We present results obtained on hardware SCSI and IDE RAID arrays, and on the Vinum software RAID [15] on FreeBSD. Our measurements indicate that IDE RAID hardware is now faster than host-attached SCSI disks or entry-level SCSI RAID arrays.

We now elaborate on these topics.

4.2.1 Disk Concurrency

Table 5 and Table 6 summarize the performance of SCSI and IDE disks for various tag, scheduler, and file system settings. We first study disk concurrency using the default FFS file system, with I/O scheduling enabled in both the disk drive and the kernel.

We observe from Table 5 that there is no improvement in performance for a sequential workload with higher concurrency (i.e. larger number of tags). This is expected since sequential I/O is already sorted in an optimal manner. Concurrency improves the performance of SCSI disks on random I/O micro benchmarks by 25-50%, but for most application benchmarks by only 3-5%. We observe very little improvement in performance beyond 4 tags, and no improvement in performance beyond 64 tags. To confirm that the increase in tags does result in higher concurrency at the disk, we measure the average queue depth in the SCSI disk using a SCSI bus analyzer. With 64 tags, we observe that the average number of outstanding requests at the disk is 60 for reads and 46 for writes, yet with 4 tags the disk has already reached its peak rate (220 requests/sec for 8KB random I/O).

For the IDE disk, tagged queuing gives improvements of 15-40% on random reads, and no improvement on random writes. This latter result is suggestive of problems with IDE tagged queuing, which is investigated in more detail in Section 4.2.2. Even with tagged queuing, IDE random read performance is nearly 50% worse than the corresponding SCSI performance. We conclude that tagged queuing in its current state is insufficient to overcome IDE's larger seek latencies.

No. of Tags	Scheduler	IDE						SCSI					
		Read			Write			Read			Write		
		Sequential	Interleave	Random	Sequential	Interleave	Random	Sequential	Interleave	Random	Sequential	Interleave	Random
none	**Both**	30.8	24.8	1.3	25.1	18.5	1.4	16.9	21.0	1.6	21.4	19.0	1.6
4		31.9	24.7	1.6	26.7	17.4	1.4	15.0	18.6	2.2	19.6	17.8	2.0
32		31.6	23.5	1.5	27.1	19.2	1.4	15.8	17.4	2.5	19.2	16.8	2.1
64		(Not Supported in IDE)						16.0	15.8	2.4	19.1	15.8	2.1
128								15.8	15.2	2.4	19.1	15.6	2.1
None	**Disk Only**	31.4	1.9	0.9	23.3	10.7	1.1	14.4	1.8	1.1	19.4	0.9	1.0
4		32.7	2.0	1.1	24.2	10.7	1.1	14.1	1.9	1.5	18.2	1.7	1.6
32		31.6	2.0	1.3	24.5	10.5	1.1	14.4	9.4	2.2	19.3	11.8	1.8
64		(Not Supported in IDE)						14.1	8.9	2.3	19.2	10.5	1.9
128								14.8	8.9	2.3	19.0	10.4	1.9
None	**Host Only**	30.7	24.4	1.3	24.3	18.2	1.4	16.5	21.0	1.6	21.1	18.7	1.6
4		32.4	24.7	1.6	24.3	17.0	1.4	14.7	18.9	1.8	19.7	17.8	1.6
32		29.4	24.5	1.5	26.7	18.1	1.4	13.8	15.0	1.7	19.1	12.3	1.6
64		(Not Supported in IDE)						14.1	11.1	1.7	18.2	8.9	1.5
128								14.3	11.1	1.7	18.4	9.0	1.5
None	**None**	32.4	1.8	0.9	23.0	10.6	1.1	15.3	1.8	1.1	19.7	0.9	1.0
4		32.2	1.9	1.1	24.7	10.5	1.1	14.4	1.4	1.2	18.3	0.9	1.0
32		31.1	2.5	1.3	25.8	10.5	1.1	14.7	1.5	1.2	17.8	0.9	1.0
64		(Not Supported in IDE)						14.6	1.5	1.2	18.2	0.9	1.0
128								14.5	1.5	1.2	17.9	0.9	1.0

Table 5: Micro benchmark Results with Different Tag and Scheduler Settings. Results are reported in Mbyte/sec. Shaded columns under IDE contain questionable results that may reflect bugs in queued DMA features of IDE disks—see text.

Number Of Tags	Scheduler	Unix FFS (Sync Metadata and Async Data Update)								Soft Updates (Async Metadata and Data Update)							
		IDE				SCSI				IDE				SCSI			
		SDET	SSH	NetNews	PostMark	SDET	SSH	NetNews	PostMark	SDET	SSH	NetNews	PostMark	SDET	SSH	NetNews	PostMark
none	**Both**	376.1	86.8	1100	500	292.2	80.9	840	406	19.8	63.5	837	199	20.0	64.2	658	181
8		(Not Supported by Driver)				282.7	80.3	809	383	(Not Supported by Driver)				19.5	63.9	574	158
32/64		374.5	86.6	1100	502	283.0	80.3	809	387	19.9	66.5	835	202	20.0	63.7	574	166
none	**Disk Only**	399.0	88.2	1234	571	348.2	83.3	971	485	20.5	64.0	931	250	19.4	65.9	824	248
8		(Not Supported by Driver)				302.8	81.1	855	407	(Not Supported by Driver)				20.3	64.4	610	182
32/64		399.9	87.9	1234	573	291.6	81.3	840	402	20.2	67.1	1001	252	20.4	64.5	585	173
none	**Host Only**	378.5	87.7	1020	500	295.8	81.3	841	405	19.9	63.6	833	200	19.8	63.8	660	174
8		(Not Supported by Driver)				287.2	81.4	828	402	(Not Supported by Driver)				20.2	64.2	652	172
32/64		380.9	87.5	1027	501	301.2	81.4	866	431	20.3	66.8	841	201	19.7	63.9	741	208
none	**None**	471.8	94.8	1141	573	345.6	83.1	974	485	19.6	68.2	940	249	21.5	65.7	823	249
8		(Not Supported by Driver)				333.7	83.1	950	471	(Not Supported by Driver)				20.7	65.9	801	245
32/64		460.2	95.0	1139	572	338.1	83.1	955	471	21.7	71.2	939	249	19.8	65.8	805	246

Table 6: Application Benchmark Results. All results are seconds of elapsed time. We vary the number of tags, disk or host scheduler, and type of file system. Due to space constraints we report SDET results only for a script concurrency of 20, and only the total elapsed time for SSH. Note that the maximum number of tags for the IDE disk is 32, versus 64 for the SCSI disk.

Samsung Ex. 1202, Page 7 of 14

4.2.2 Disk Scheduling

Scheduling may be performed at the disk, which has intimate knowledge of the disk geometry. Alternatively, scheduling may occur within the device driver or kernel, either of which may have access to a larger pool of schedulable requests in the form of dirty blocks residing in a buffer cache. Throughout this paper we refer to the former as *disk scheduling*, and the latter as *host scheduling*.

We first make the obvious observation that it is crucial to have some type of scheduler (disk, host or both) in the system. From Tables 5 and 6, we see that for many of the benchmarks, having both schedulers provides a substantial benefit over no scheduler, for both SCSI and IDE disks. We now compare the performance of host and disk schedulers. We observe from Table 5 and 6 that when concurrency is below 32, the host scheduler consistently out-performs the disk scheduler. This is expected, because the host scheduler is working on a larger pool of I/O requests (maximum of 4096 versus 32 on disk). However, when the disk has sufficient concurrency, it can outperform the host scheduler. For example, we see from Table 5 that on random I/O, the SCSI disk scheduler outperforms the host scheduler by 13-35% when the concurrency exceeds 32. In the SCSI application benchmark results in Table 6, we see less drastic differences: the disk scheduler outperforms the host scheduler for NetNews by 12%. (SSH and SDET show less improvement as they are less I/O bound.) On PostMark and NetNews, the performance gap is more obvious on the Soft Update file system than on FFS (20-27% versus 3-12%). This is because synchronous I/O dominates the application performance on FFS, whereas the Soft Update file system exploits a large pool of asynchronous buffers in both the host and disk schedulers.

A non-intuitive result is that the host scheduler's performance on SCSI disk decreases with increasing number of tag (e.g. random and interleave macro benchmarks). This is due to a complex interaction between the host scheduler and disk's tag queuing mechanism. We explain a simple scenario in Figure 1, which shows a host scheduler with a buffer size of 2 scheduling I/O requests. The shaded boxes represent new requests, which alternate between the interleaved addresses. Figure 1a shows that when there is no tagged queuing, the disk sees more sequential requests (i.e. 100, 101, 102, N+100, N+101, N+102...). This schedule incurs less costly disk seeks than that generated by a host scheduler with tagged queuing enabled (see Figure 1b). The latter schedule incurs more disk seeks since it alternates between interleaved addresses (i.e. 101, N+100, 102, N+101...). In general, it is crucial that tag queuing is only enabled when the disk scheduler is active.

The results for disk scheduling in an IDE disk are puzzling. We fail to see the expected performance increase of the interleaved micro benchmarks with increasing numbers of tags under disk scheduling. Also, there is no performance difference between disk scheduling and no scheduling. These results are indicative of problems in the current implementation of tagged queuing DMA, either in FreeBSD or in the disk firmware. Instrumentation shows that the disk is not performing any scheduling for interleaved or random writes. In the case of interleaved reads, traces indicate that requests are initially scheduled by the disk, but revert to their original interleaved ordering after a few hundred have been satisfied. **Note to referees: We are working with the FreeBSD IDE driver author to locate the source of this problem (i.e. driver, IDE controller or IBM disk?) and expect to have conclusive results in the final version of this paper**.

Because our experiments use the default I/O scheduler in FreeBSD, our results do not reflect the full potential of host scheduling. Scheduling algorithms that utilize detailed knowledge of physical data layout on disk, and that accurately track the disk arm position, can outperform BSD's elevator algorithm [12, 26, 36].

Time	Disk Addresses		RequestServed	Disk Addresses		RequestServed
T_1	100	101	100	100	101	100,101
T_2	N+100	101	101	N+100	102	N+100,102
T_3	N+100	102	102	103	N+101	N+101,103
T_4	N+100	N+101	N+100	N+102	104	N+102,104
T_5	103	N+101	N+101	105	N+103	N+103,105
T_6	103	N+102	N+102	N+104	106	N+104,106
	...				...	

(a) Host Scheduler, no Tagged Queuing **(b) Host Scheduler, # of Tag=2**

Figure 1: I/O Schedule as Seen by the Disk During an Interleave Micro Benchmark.

Samsung Ex. 1202, Page 8 of 14

4.2.3 File System Designs

Table 6 presents application benchmark results comparing FFS with Soft Updates on both IDE and SCSI disks. We observe that on FFS, SCSI consistently out-performs IDE (15-47%) on most application benchmarks (except SSH which is less I/O bound). On the Soft Updates file system, SCSI and IDE disks have comparable performance for most benchmarks except NetNews: The NetNews buffer footprint exceeds the system buffer pool, so the application is occasionally blocked while the buffer pool is cleaned. This cleaning causes synchronous random writes, a workload for which the SCSI disk has significantly better performance. We also note that on the Soft Updates file system, higher concurrency settings increase the performance advantage of SCSI over IDE (e.g., from 10% to 22% on PostMark). This is mainly due to the buggy IDE tagged queuing, so we expect this advantage to diminish in the future. On balance, Soft Updates helps compensate for the slower seeks of the IDE disk.

We now explore the performance of Soft Updates and journaling in more detail by examining a low-level trace of physical disk accesses during benchmark runs, as recorded by a SCSI bus analyzer. We limit our discussion to the PostMark benchmark due to space constraints, but our observations are similar for the SDET and NetNews benchmarks.

Figure 2 shows the sequence of disk accesses during a PostMark run on three file systems: FFS, Soft Updates, and NTFS. Each point indicates the time and corresponding disk address of a physical disk write.

Figure 2a shows the SCSI trace during a PostMark run on FFS. The PostMark benchmark has 3 phases: **create** files, **run** transactions, and **delete** files. The create phase touches inodes in many cylinder groups [19], thus we see the gradual march from the first to *Nth* cylinder group, where *N* is determined by the data set. The run phase updates data created during the create phase, and thus we see disk activity from the first to the *Nth* cylinder group. Because the run phase also creates files, we see additional activity above the *Nth* cylinder group (i.e. disk address from 180000-240000). The vertical lines in the run phase represent update daemon activities, occurring once every 30 seconds [19]. These writes are asynchronous and are sorted by the host scheduler in ascending order to minimize disk seeks, and appear as vertical lines due to the compressed time scale. The horizontal lines in Figure 2a represent synchronous metadata updates for each cylinder group. These synchronous writes, spread over many cylinder groups, make random writes the dominant factor. Thus, for this workload on FFS, IDE is much slower than SCSI.

Figure 2b shows the same experiment on the Soft Updates file system. This file system has very few synchronous metadata writes [27], and thus we do not see any horizontal lines. Almost all writes are asynchronous and can be scheduled nicely (thus the straight vertical lines). The delete phase in the Soft Update file system is considerably faster than on FFS, because most files are created and deleted entirely within the buffer cache, so relatively little physical disk I/O occurs [27]. With writes not being a factor, the determining factor for the Soft Update file system is concurrency (i.e. how fast the disks absorb the sorted asynchronous writes). IDE and SCSI are somewhat comparable with no tagged queuing, but current IDE systems improve less than SCSI does when tagged queuing is enabled, as seen in Table 6.

Our measurements show that IDE is faster (10%) than SCSI on the PostMark benchmark when running on the Windows NTFS file system. NTFS is a journaling file system, so its metadata write activity is largely sequential. Consequently, IDE has comparable performance to SCSI, as seen in Figure 2c. We observe that the writes are clustered in 3 regions. The middle region contains asynchronous writes to the file data. The two horizontal lines in Figure 2c represent sequential writes to the log files in the Master File Table (MFT) and the MFT mirror [30] (magnified view in Figure 2d). The MFT is located near the start of the disk, and the MFT mirror is located in the middle of the disk. Because the sequential writes alternate between these two regions, the I/O accesses resemble the interleaved I/O modeled in Table 5. We know that SCSI and IDE have comparable performance for that access pattern.

In summary, we find that file system design has great potential to enable IDE to achieve performance comparable to SCSI. In journaling or log-structured file systems or with workloads dominated by sequential or interleaved writes, IDE performs comparably to SCSI. IDE also performs well under Soft Updates. Disregarding tagged queuing, IDE achieves 80% of SCSI's performance for NetNews and 90% for PostMark, and would be comparable to SCSI in this environment if IDE's tagged queuing performed as expected.

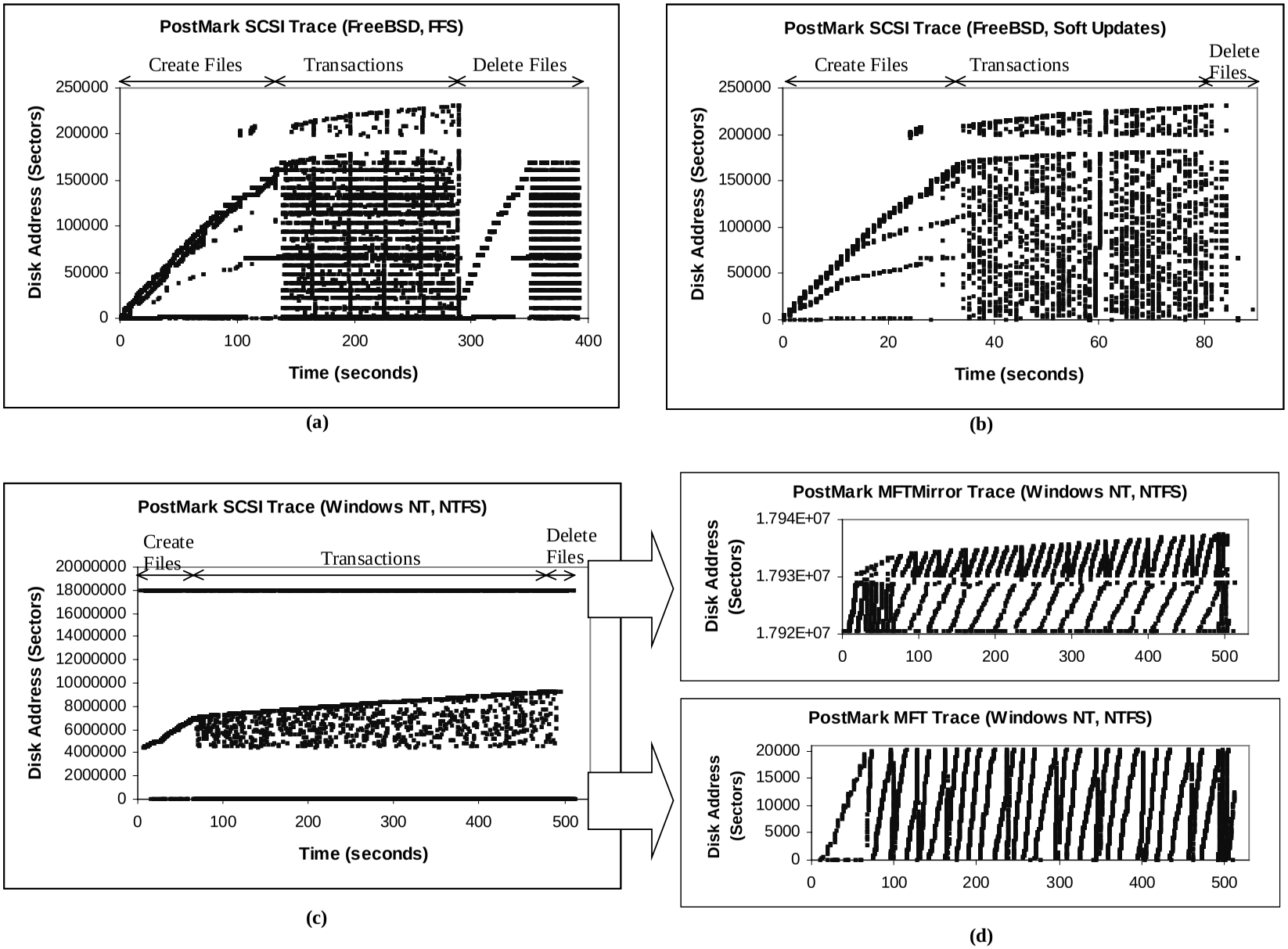

Figure 2: SCSI Trace of PostMark Benchmark. **The I/O traces are collected from the SCSI bus using a SCSI bus analyzer.**

Samsung Ex. 1202, Page 10 of 14

4.2.4 Parallelism

In this section we explore the performance of disk arrays. The software disk array is configured as RAID 0 (striping only, no parity computation) since a RAID 5 configuration would result in a bottleneck caused by parity I/Os, masking the IDE vs. SCSI performance issues that we are investigating. The hardware array and the software array both use a 64KB stripe size.

		Software Disk Array (Vinum)			Hardware Disk Array		
		IDE	**SCSI**	**% Faster**	**IDE**	**SCSI**	**% Faster**
Write Micro benchmark **(MB/sec)**	Sequential	6.9	23.9	-246%	15.1	35.2	-133%
	Interleave	7.2	23.5	-226%	15.4	25.8	-68%
	Random	5.4	4.2	22%	3.8	3.3	13%
FreeBSD Unix FFS **(seconds)**	SDET	303.1	244.9	-19%	21.5	43.1	100%
	SSH	86.8	78.3	-10%	64.6	65.1	1%
	NetNews	542	420	-23%	410.0	326.0	-20%
	PostMark	205	237	16%	57.0	106.0	86%
FreeBSD Soft Update File System **(seconds)**	SDET	16	16.9	6%	16.5	16.2	-2%
	SSH	62.5	64.1	3%	63.2	63.3	0%
	NetNews	344	255	-26%	309.0	301.0	-3%
	PostMark	74	96	30%	49.0	82.0	67%

Table 7: SCSI versus IDE Disk Array Performance. The hardware disk array has 8 disks configured in RAID5 with 7+1 spare. The software disk array has 4 disks configured in RAID 0 (striping only without redundancy). The % Faster column compares IDE versus SCSI performance.

Table 7 compares the performance of software and hardware disk arrays for the write micro benchmark and for the application benchmarks running on the FFS and the Soft Updates file systems. We omit reporting the read micro benchmarks and other variables (Windows NT, tag settings) as they merely reinforce the results.

The write micro benchmark results indicate that SCSI is faster on sequential I/O, but slower on random I/O. These results contradict the single disk results in Table 5. These differences are largely due to system design factors. In particular, both the hardware and software IDE disk arrays have lower bus contention because they use 1 bus per IDE disk, whereas the SCSI disk array has one (Vinum software array) or two (hardware RAID) SCSI buses in total. The IDE hardware RAID's slower sequential performance may reflect an inefficient I/O scheduling algorithm, as other vendors (e.g. 3ware and JetStor) have demonstrated much greater sequential bandwidth.

The micro benchmark results provide clues into the macro benchmark results. We first look at the software IDE array. The software IDE array is mostly slower when running on the FFS file system. Both SDET and SSH have a small buffer footprint, and their I/Os are smaller than the stripe unit (2KB versus 64KB) and tend to concentrate on several cylinder groups in a single disk. Thus the benchmark workload degenerates into random I/O on a single disk, so the SCSI array performs better. The single disk bottleneck can be mitigated with NVRAM to absorb metadata updates for hot cylinder groups. Both hardware disk arrays have large (128 MB) buffer caches, thus the remaining performance obstacle is random writes. The software IDE array is faster than the SCSI array on configurations that have large amounts of asynchronous random writes, thus it performs better on the SDET, SSH, and PostMark benchmarks when using the Soft Update file system. NetNews' buffer footprint exceeds the system buffer pool and thus is dominated by metadata writes (both random and interleaved), thus it exhibits the worst performance under both file system configurations. Since the IDE hardware disk array has faster random writes, it performs better on most benchmarks.

We can tailor a disk array to take advantage of IDE technology. In particular, IDE controllers are significantly less expensive than SCSI controllers (e.g. the Promise PDC20267 chip is at least 6 times less expensive than Adaptec's 78xx controller), so one can eliminate I/O bus contention in an IDE array by using a separate bus for each disk. The disk array can also export a SCSI interface to the host to overcome the lack of concurrency at the IDE interface.

5 Related Work

5.1 IDE versus SCSI Test Reports

The debate over the relative performance of IDE and SCSI has been fought hard and long. The Internet and USENET (e.g. see *comp.periphs.scsi*) are replete with comparisons of the two technologies.

The online PC Guide [22] provides an excellent overview of SCSI and IDE. The site discusses the relevant metrics of comparison, while stopping short of a full-fledged quantitative evaluation. Another excellent source for SCSI and IDE information is the book by Schmidt [25].

Cardenas and Catena [3] compare popular drive performance characteristics and costs. Their benchmark highlights the performance of a wide range of disks under expected usage in a digital audio recording and production environment. Their workload is largely sequential and has real-time constraints. Their main conclusion is that for digital audio, the disk media transfer rate is the most important factor. They do not treat accesses to multiple devices or general-purpose workloads.

Murakami provides another good source on the performance characteristics of modern SCSI and IDE disk drives [20]. He examines the performance of SCSI and IDE disks in single disk and RAID configurations under the Linux operating system. He measures the raw device bandwidth using the Linux hdparam program, and file system performance using the bonnie and bonnie++ workloads. The later workloads invoke various file system operations like sequential I/O, random seeks, and directory operations. Murakami's results indicate that among IDE and SCSI drives of comparable specifications, IDE performance is comparable or faster. The difference is especially pronounced for write operations. Murakami provides no insight into his results. We suspect that the difference in write performance may be a consequence of a disabled write cache (WCE [25]) on the SCSI drive.

Stam [31] examines SCSI and IDE disks with similar specifications in a Windows environment. He did not yield conclusive disparities, although SCSI provides performance superior to IDE in the presence of heterogeneous devices such as a CD-ROM and disk. No explanation is given for any of his results.

Martin and Scholl [17] shows that a 10,000 RPM Wide Ultra SCSI disk achieves a 50-60% improvement over a "Best Ultra DMA" IDE disk on the ThreadMark and WinBench benchmarks. The paper touts SCSI's ability to issue multiple I/O requests through tagged queuing, but leaves the details of the experiments and the impact of tagged queuing insufficiently well specified to enable the replication of these results.

Stone [32] utilizes the same ThreadMark and WinBench benchmarks as Martin and Scholl, but obtains a completely different result. Promise's FastTrak IDE RAID controller is contrasted with IDE and Ultra Wide SCSI single disk configurations. The results indicate that IDE disks are faster than SCSI on these benchmarks. FastTrak's speedup over its competitors is not explained.

Recent work at Microsoft has targeted I/O performance. Riedel et al. [24] investigate the performance of the Windows NT File System, and provide a thorough discussion of bus, controller, and file system overheads, but only address sequential I/O on SCSI disks. Follow-on work [4] continues in a similar vein, but also incorporates studies of random workloads and IDE. This paper suggests that aggregating IDE disks through IDE RAID leads to a viable, inexpensive, and efficient storage system. The authors report that IDE has a 25% higher 'I/Os per second per dollar' ratio than SCSI for random I/O, while approaching the performance of SCSI for sequential workloads. They attain linear improvement in read throughput and scalability for writes using up to three IDE disks with 3ware's IDE RAID adapter card [1]. Measurements are obtained via micro benchmarks on Windows NT and 2000.

We find that much of the above work suffers from an incomplete specification of the test environment and system configuration. Some studies compare widely different products. Most studies also do not use realistic application benchmarks and rely on micro benchmarks alone. Finally, most studies do not provide sufficient insight to explain their results. By contrast, in this paper we present a thorough comparison of SCSI and IDE across a range of popular application benchmarks. We use similar disks throughout the comparisons and provide a full description of the system configuration. Further, low-level micro benchmarks, kernel instrumentation, and SCSI analyzer traces illuminate our findings.

Samsung Ex. 1202, Page 12 of 14

5.2 Disk Performance Studies

Kerns provides a detailed introduction to SCSI tagged queuing, and assesses the importance of this capability via micro benchmark studies [14]. His results suggest that a small number of tags (<32) is sufficient to achieve good performance.

Besides concurrency, disk scheduling is also crucial in reducing seek latencies [26, 12]. Scheduling may be performed at the disk, which has intimidate knowledge of the disk geometry, or within the kernel, which may have a large pool of schedulable requests. The tradeoff between these two approaches is studied in detail in [26], where the authors show the merits of host scheduling. Recent work on extensible kernels also advocates fine-grain control of disk resources for maximum flexibility and adaptivity (e.g., Nemesis [2]).

The file system design has a significant effect on disk performance as it determines the data layout and physical access patterns. Unix FFS [18] statically allocates regions of the disk for inodes, and tries to locate data and metadata in rotationally optimal positions. However, metadata operations still incur significant seek and rotational latencies [9]. Moreover, FFS typically writes its metadata synchronously to disk to maintain file system integrity. Many recent studies reduce synchronous and non-sequential accesses at the file system (see [27] for an excellent introduction). In this paper we evaluated two modern file systems. Soft Updates [9] tracks dependencies between metadata blocks such that metadata writes may be delayed, thus significantly reducing the number of synchronous I/Os and allowing for more effective scheduling. A journaling file system writes its metadata sequentially to disk, greatly reducing disk seeks [30].

6 Conclusions

We have presented a thorough examination of IDE and SCSI performance through a combination of micro benchmarking and macro benchmarking. The principal independent variables for the benchmarking are workload factors such as sequentiality, locality, and read/write ratio, and system aspects such as I/O concurrency (as enabled by tagged queuing), disk scheduling in the host or on the disk, file system features such as journaling and soft updates, and I/O parallelism via disk arrays.

The IDE disks that we measured are generally faster than the SCSI disks for sequential I/O, but slower for random I/O. Our experimental results indicate that we can mitigate the random I/O handicap of IDE by appropriate choices with respect to the system aspects mentioned above.

We also measured the performance of software and hardware RAID arrays built from IDE disks and from SCSI disks. The IDE and SCSI arrays exhibit similar performance. We note several techniques that may give substantial additional performance improvements to the IDE array.

7 Acknowledgements

We would like to thank Liddy Shriver for suggesting that we compare host and disk scheduling. Along with Liddy, Margo Seltzer did the initial work in this area and allowed us to follow in her tracks. We are indebted to Keith Smith for help in acquiring and running the benchmarks.

8 References

1 3ware. www.3ware.com
2 P. Bosch. *Mixed Media File Systems*. PhD thesis. University of Twente, 25 June 1999.
3 D. Cardenas, J. Catena, "SCSI vs. IDE: Bus Mastering for DAWs", Audio Amiga Report, 2000. http://www.txconnect.com/home/ignot/article/scsi_ide.htm
4 L. Chung, J. Gray, B. Worthington, R. Horst, "Windows 2000 Disk IO Performance", Microsoft Research Technical Report MS-TR-2000-55, June 2000.
5 M. Delsman, "The Future of SCSI", http://www.scsita.org/aboutscsi/pub/PCDC9905_RoadMap.pdf
6 Deskstar 40GV & 75GXP Hard Disk Drive Specifications, Version 1.2, IBM Corp. May 2000.
7 P. Den Haan, "Enhanced IDE FAQ and Utilities", http://thef-nym.sci.kun.nl/~pieterh/storage.html

8 S. Gaede, "Perspectives on the SPEC SDET Benchmarks", http://www.spec.org/osg/sdm91/sdet/index.html.

9 G. Ganger, Y. Patt, "Metadata Update Performance in File Systems", *OSDI Conf Proc.*, pp. 49-60, Monterey, CA, Nov. 1994.

10 K. Grimsrud, "Serial ATA Overview", http://serialata.org/F9pp.pdf

11 Highpoint Technologies, Inc. www.highpoint-tech.com

12 D. M. Jacobson, J. Wilkes, "Disk scheduling algorithms based on rotational position," HP Laboratories technical report HPL-CSP-91-7rev1, Feb 1991 (revised Mar 1991)

13 J. Katcher, "PostMark: A New File System Benchmark", Technical Report TR3022. Network Appliance Inc, Oct. 1997.

14 R. Kerns, "SCSI command tag queuing and cached disk performance", *Storage Management*, 6(11), pp. 18-11, Nov. 1998.

15 G. Lehey, "The Vinum Volume Manager," *Proceedings of the FREENIX Track: 1999 USENIX Annual Technical Conference*, Monterey, CA, June 1999.

16 G. Leyzarovich of Advanced Computer & Network Corp and P. Neff of RaidWeb Inc. Personal Communications.

17 T. Martin, A. Scholl, "SCSI Remains the I/O Interface of Choice for Workstations: An Analysis Comparing SCSI and Ultra DMA", SCSI Trade Association white paper, Jan 1998. http://www.scsita.org/whitepaper/SCSIUDMA.pdf

18 M. McKusick, W. Joy, S. Leffler, R. Fabry, "A Fast File System for Unix", *ACM Trans. On Comp. Sys.* 2(3), pp. 181-197, Aug. 1984.

19 M. McKusick, K. Bostic, M. Karels, J. Quarterman, *The Design and Implementation of the 4.4BSD Operating System*. Addison-Wesley, 1996.

20 G. Murakami. http://www.research.att.com/~gjm/linux/ide-raid.html

21 D. Patterson, G. Gibson, R. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)," *Proceedings of the SIGMOD Conference*, June 1998.

22 PC Guide. http://www.pcguide.com

23 Promise Technologies, Inc. www.promise.com

24 E. Riedel, C. Ingen, J. Gray, "Sequential I/O on Windows NT: Achieving Top Performance", Microsoft Research Technical Report MSR-TR-97-34, 1997.

25 F. Schmidt, *The SCSI Bus & IDE Interface, Protocols, Applications & Programming.* Addison-Wesley, 1998.

26 M. Seltzer, P. Chen, J. Ousterhout, "Disk Scheduling Revisited", *USENIX Winter Conf. Proc.*, pp. 313-324, Washington, DC, Jan. 1900.

27 M. Seltzer, G. Ganger, M. McKusick, K. Smith, C. Soules, C. Stein, "Journaling versus Soft Updates: Asynchronous Meta-data Protection in File Systems", *USENIX Conf. Proc.*, pp. 71-84, San Diego, CA, June 2000.

28 A. Silberschatz, P. Galvin, *Operating System Concepts.* Addison-Wesley, 1994.

29 K. Smith, M. Seltzer, "File System Aging – Increasing the Relevance of File System Benchmarks", *Sigmetrics Conf. Proc.*, pp. 203-213, Seattle, WA, June 1997.

30 D. Solomon, *Inside Windows NT, Second Edition.* Microsoft Press. 1998.

31 N. Stam, "SCSI vs. EIDE: The Real Story", PC Magazine Reports, July 1996. http://www.zdnet.com/pcmag/pclabs/report/r960702a.htm

32 M. D. Stone, "Mixing IDE and SCSI Hard Disks", PC Magazine Reports, Nov 1999. http://www.zdnet.com/filters/printerfriendly/0,6061,2354187-50,00.html

33 K. Swartz, "The Brave Little Toaster Meets Usenet", *LISA '96*, pp. 161-170, Chicago, IL, Oct. 1996.

34 N. Talagala, D. Patterson, "An Analysis of Error Behavior in a Large Storage System", 1999 Workshop on Fault Tolerance in Parallel and Distributed Computing, Jan 1999.

35 Ultrastar 36ZX and 18LZX Functional/Hardware Specification, Version 2.01, IBM Corp., Sep. 1999.

36 G. Ganger, B. Worthington, Y. Patt, "Scheduling Algorithms for Modern Disk Drives", *Sigmetrics Conf. Proc.*, pp. 241-252, Nashville, TN, May 1994.