



V
E
R
I
T
A
S

W
H
I
T
E

P
A
P
E
R

Performance Comparison:

VERITAS File System™ Version 3.4 Patch 1 vs. Solaris 8 Update 4 UFS



Table of Contents

Executive Summary	1
Availability — Why Journaling is Mandatory	2
Scalability	2
Management	3
1. Performance of File System Recovery (full fsck)	4
1.1 Introduction	4
1.2 Test Configurations	4
1.3 File Sets	5
1.4 Experiments	6
1.5 Results	7
2. Performance as an NFS Server (SPECsfs97 Benchmark)	10
2.1 Introduction	10
2.2 Test Configurations	10
2.3 Overview of Results	10
2.4 Detailed Results	11
2.5 SPECsfs97 Defects and Our Use of the Benchmark	15
3. Performance in an Oracle Environment (TPC-C Benchmark)	16
3.1 Introduction	16
3.2 Test Configurations	16
3.3 Results	18
4. Performance of Miscellaneous Commands (tar extract, cp, mkfile, touch)	22
4.1 Introduction	22
4.2 Test Configurations	22
4.3 Overview of Results	23
4.4 Detailed Results	24
4.5 Summary	29
5. Performance of Sequential File I/O (vxbench)	30
5.1 Introduction	30
5.2 Test Configurations	30
5.3 Experiments	31
5.4 Results	31
Appendix A: Configuration Information for SPECsfs97 Benchmarks	40
1. E6500 (8X8)	40
2. E6500 (12X12)	43
Appendix B: Additional Software Configuration Information for TPC-C Benchmark	46
Appendix C: Source Code for touch_files Program	48
Appendix D: Complete vxbench Results	50

Executive Summary

This paper compares the performance of [VERITAS File System™](#) version 3.4 Patch 1 and Sun UNIX File System (UFS) on Solaris 8 Update 4 (64-bit unless otherwise specified) under a variety of workloads. The following chart outlines the benchmarks and primary conclusions:

Benchmarks	UNIX File System (UFS)	UFS + Logging	VERITAS File System	Advantage
Recover after a system crash (e.g., power failure), fsck	48 minutes	seconds	seconds	Logging is a necessity in today's environment; true for both file systems.
Recovery from unusual event (e.g., physical disk corruption), fsck	48 minutes	43 minutes	6 minutes	VERITAS File System has more than six times faster recovery over UNIX File System with logging.

Because file system availability is critical to system availability, we will focus this summary comparison on UNIX File System with logging, to VERITAS File System in the following chart (report contains details on UNIX File System, UNIX File System with logging, and VERITAS File System results).

Benchmarks	UFS + Logging		VERITAS File System		Advantage
	Ops/s	ORT	Ops/s	ORT	
File Serving:					VERITAS File System has approximately twice the throughput with better overall response time. With a significantly larger server, the UNIX File System with logging performance gain was only 3 percent.
(SFS 2.0) on a 8 CPU / 8 GB server	5,610	6.6	10,942	5.4	
(SFS 2.0) on a 12 CPU / 12 GB server	5,791	6.3	13,851	5.0	
Sequential File I/O: vxbench 24-column stripes	VERITAS File System outperforms UFS + Logging				VERITAS File System performance lead increases as the number of disks in the volume increases. VERITAS File System performance lead increases as you access more files concurrently.
Reads	8K Reads: 26% to 348% 64K Reads: 55% to 323%				
Writes	8K Writes: 73% to 302% 64K Writes: 74% to 377%				
OLTP throughput: (tpm-C)	Concurrent Direct I/O (CDIO)		Quick I/O (QIO)	Cached QIO (CQIO)	VERITAS File System is 8 percent to 20 percent faster on database performance than UNIX File System CDIO. VERITAS File System is 100 percent faster when used with Storage Checkpoints for block-level backup compared to UNIX File System snapshots.
TPC-C	3,963		4,276	4,759	
TPC-C with checkpoints/snapshots	1,687		3,419	*	
Miscellaneous File I/O:	VERITAS File System outperforms UFS + Logging				VERITAS File System shows stronger results as environments become more complex and concurrent commands (mkfile, touch_files, and tar) are run.
mkfile	58% to 117% faster				
touch_files	19% to 338% faster				
tar extract	69% to 84% faster				
cp	19% to 80% faster				

* Cached Quick I/O with Storage Checkpoint™ tpm-C tests were not performed at this time and will be included in the next release of the performance study.

Availability — Why Journaling is Mandatory

fsck – File system consistency check command (Section 1)

UNIX File System fsck performance essentially prohibits its use on high availability servers having large file systems. Fsck time for 70 GB on a Sun StorEdge D1000 JBOD is high at 45 minutes. A 768-GB UNIX File System fsck on three Sun StorEdge A5200 JBODs takes about 4.8 hours. A typical server has many file systems and multiple file systems are commonly striped across several disks. Because of this, even with parallel fsck, a server with non-journaling file systems will take many hours to run file system checks. Thus, the use of a journaling file system, such as UNIX File System with logging or VERITAS File System, should be considered mandatory on any server.

- VERITAS File System full file system consistency checks are six times faster than UNIX File System with logging.
- VERITAS File System has been tuned to ensure maximum availability while delivering maximum performance.

Scalability

SFS 2.0 — A standard file serving benchmark (Section 2)

On this benchmark, VERITAS File System achieved between 34 percent and 36 percent greater throughput than UNIX File System and between 95 percent and 135 percent greater throughput than UNIX File System with logging. VERITAS File System also enabled faster overall response time.

- VERITAS File System produces twice the throughput of UNIX File System with logging while delivering faster response times.
- UNIX File System with logging has a performance penalty and does not scale as well as VERITAS File System as processors are added.

Sequential File I/O — The vxbench utility measures high-bandwidth sequential disk transfers (Section 5)

When transferring large sequential amounts of data to and from a striped volume using vxbench, VERITAS File System scales linearly as more disks are added to a striped array. In addition, when presented with a concurrent load of disjoint sequential data transfers (up to 32 processes), VERITAS File System outperforms UNIX File System with logging, by 300 percent.

- VERITAS File System scales with increasing concurrency, outperforming UNIX File System with logging by 26 percent to 377 percent.
- VERITAS File System scales with increasing RAID-0 column sizes, provided consistent throughput in mixed traffic environments.

TPC-C — A standard online transaction processing throughput performance benchmark (Section 3)

VERITAS File System buffered I/O achieves greater tpm-C throughput than UNIX File System and UNIX File System with logging. The Quick I/O feature of VERITAS Database Edition showed an increase over Sun's new Concurrent Direct I/O, (CDIO, an earlier name was DBIO), with substantial advantages when used with VERITAS File System Storage Checkpoints.

- The Quick I/O and Cached Quick I/O features of VERITAS Database Edition outperforms Solaris 8 Update 4 CDIO by 8 percent to 20 percent.
- When used with Storage Checkpoints (used to provide more granular backup and restore options), VERITAS File System performs at twice the speed.

Note: VERITAS File System Storage Checkpoints are persistent and UNIX File System snapshots are not.

Management

Miscellaneous file benchmarks — mkfile, touch_file, tar extract, cp (Section 4)

VERITAS File System outperformed UNIX File System with logging by as much as 338 percent in these tests and enabled better system scalability.

- VERITAS File System consistently outperforms UNIX File System with logging: mkfile (58 percent to 117 percent), touch_files (19 percent to 338 percent), cp (19 percent to 80 percent) and uncompress/tar extract (69 percent to 84 percent).
- In the touch_files benchmark, UNIX File System with logging performs well on single-process operations, but as the degree of multiprocessing becomes greater, it increasingly lags behind VERITAS File System.

In summary, VERITAS File System gives you the performance you need during operations, as well as the quick recovery you rely on during reboots. Sun's CDIO implementation requires Solaris 8 Update 3 or later and Oracle 8.1.7 or later. VERITAS' implementation works with Oracle 7 and later, as well as Solaris 2.6 and later. VERITAS allows customers to leverage their existing investments as well as provide new implementations — there is no need to upgrade immediately. VERITAS File System 3.4 performance is approximately the same across operating system releases.

1. Performance of File System Recovery (full fsck)

1.1 Introduction

This section evaluates the performance of file system checks (*fsck*) for [VERITAS File System™](#) 3.4 Patch 1 and Sun UNIX File System running on a Solaris 8 Update 4 operating system. File system checking must be fast on high availability servers, which cannot afford long downtimes. Journaling file systems, such as VERITAS File System and UNIX File System with logging, usually can perform a file system check in seconds, through replaying a log of metadata changes that have not yet been committed to disk. Only if the log has become damaged is the more thorough “full fsck” required, where the entire file system’s content is examined for consistency. By contrast, checking a file system that does not have the benefit of logging (such as UNIX File System without logging) always requires the more expensive full fsck.

This section examines the performance of full fsck for UNIX File System, UNIX File System with logging, and VERITAS File System on three machine and disk configurations. We find that due to high full fsck cost, the use of UNIX File System without logging in a high availability server environment is prohibitive. In such environments, *the use of a journaling file system should be considered mandatory, not a luxury*. This conclusion has an important implication for the performance studies presented in the remainder of this paper. Because UNIX File System without logging is not a viable file system in a high availability server due to full fsck time, the primary “baseline” competition to VERITAS File System is UNIX File System with logging. As we will see, UNIX File System with logging performs worse than UNIX File System in most benchmarks.

In the rare case when a journaling file system (such as UNIX with logging and VERITAS File System) is unable to replay its log during a fsck, the more expensive full fsck is required. At such times, VERITAS File System performs a full fsck six to eight times faster than UNIX File System with logging.

1.2 Test Configurations

Full fsck tests were performed on three systems, each running Solaris 8 Update 4. The first is a Sun E4000 computer system, with four 168-MHz UltraSPARC-I CPUs and 832 MB of RAM. Due to hardware bugs in some UltraSPARC-I chips that prevented 64-bit addressing, Solaris was booted in 32-bit mode *for this machine only*. (All other configurations in this paper use 64-bit Solaris). Two Sun StorEdge D1000 JBOD arrays, each with 12 Seagate ST-39173W (Barracuda 9LP) 7,200 RPM 9-GB disks, provided disk space. Using [VERITAS Volume Manager™](#) 3.1.1, these arrays were configured into a 12 column stripe-mirror (RAID-1+0) volume of 100 GB. The arrays were directly connected to fast wide SCSI ports on the E4000.

The second test system is a Sun Blade 1000, with two 750-MHz UltraSPARC-III CPUs and 1 GB of RAM. A single Sun StorEdge T3 hardware RAID-5 array provided disk space for the experiment. The T3 contains nine Seagate ST-318304FC (Cheetah 36 LP) 10,000 RPM 18-GB disks, with eight disks used for data and one for redundancy. Additionally, the T3 was configured to use write-back caching (the default). The array was connected directly to a built-in Fibre Channel-Arbitrated Loop port on the Blade and configured to a 100-GB volume.

The third system used for fsck testing is a Sun E4500, with eight 400-MHz UltraSPARC-II CPUs and 2 GB of RAM. Three Sun StorEdge A5200 JBOD arrays, each with 22 Seagate ST-318304FC (Cheetah 36 LP) 10,000 RPM 18-GB disks, were connected via gigabit fibre. (Two of the disks were non-functioning; this study used the remaining 64 disks.) Two of the arrays were connected to the same Sbus board via JNI 1083 cards, while the third was connected to a PCI board via a Qlogic 2200 card. Using VERITAS Volume Manager 3.1.1, these arrays were configured into a striped (RAID-0) volume of 1 TB.

1.3 File Sets

For the E4000 and Blade 1000 configurations, the same file set was used to populate the file system before running fsck. This file set is a subset of the data produced by a run of the SPECsfs97 benchmark, with 2,870,713 files in 88,772 directories totaling 72,641,776 KB (about 69.3 GB). File sizes range from 0 bytes to 1.35 MB, with a heavy concentration on most of the power of two file sizes. Table 1 shows the file size distribution used in the 100-GB fsck.

File Size Range	Number of Files
Up to 4 K	1,990,612
>4 K to 16 K	473,044
>16 K to 64 K	242,107
>64 K to 256 K	135,901
>256 K to 1 MB	28,063
>1 MB to 1.35 MB	981
About 31 MB (the .tar.gz files)	5

Table 1: File size distribution for 100 GB volume full fsck tests (E4000 and Blade 1000 configurations)

To avoid re-running SPECsfs97 to produce the file set each time, the files were archived into five .tar files which were compressed using gzip (each .tar.gz file representing one of the five top-level directories produced by the prior run of SPECsfs97). The five .tar.gz files were each about 31 MB and are included among the files on which fsck was run.

For the E4500 configuration, a larger (though similar) file set was used. First, the five top-level SPECsfs97 directories were brought into a single .tar file which, when compressed using gzip, is 156 MB. Then, 11 copies of this .tar.gz file were created. When uncompressed and extracted, the file set totals about 768 GB, with a size distribution summarized in Table 2.

File Size Range	Number of Files
Up to 4 K	21,896,732
>4 K to 16 K	5,203,484
>16 K to 64 K	2,663,177
>64 K to 256 K	1,494,911
>256 K to 1 MB	308,693
>1 MB to 1.35 MB	10,791
About 156 MB (the .tar.gz files)	11

Table 2: File size distribution for 1 TB volume full fsck tests (E4500 configuration)

1.4 Experiments

For each machine configuration, `mkfs` was used to create a file system using, UNIX File System or VERITAS File System, across the entire volume. For the E4000 and Blade 1000 configurations, the file systems were 100 GB. For the E4500 configuration, VERITAS File System created a 1 TB file system ($2^{31}-1$ 512 byte sectors). Using default `newfs` options, UNIX File System was unable to create a file system encompassing the entire volume. Through trial and error, UNIX File System eventually succeeded in creating a file system of 1,996,000,000 sectors, or about 951 GB. For VERITAS File System, non-default `mkfs` options used were: 1 K block size, 16 MB log, and large file support. For UNIX File System, the default `mkfs` options were used.

After `mkfs`, a file system (either UNIX File System, UNIX File System with logging, or VERITAS File System) was mounted on the volume. The UNIX File System was mounted either with no options (UNIX File System without logging) or with the `logging` flag (UNIX File System with logging). VERITAS File System was mounted with the delayed logging option to match the semantics of UNIX File System with logging.

After mounting, the file system was populated with the `.tar.gz` files, as described above. These files then were uncompressed and extracted. (Section 4 contains timing information for the `uncompress` and `extract` steps on the Blade 1000 and E4500 configurations.)

After uncompressing and extracting, the file system was unmounted and a full `fsck` was run using the following command:

```
/bin/time fsck -F fstype -n /dev/vx/rdisk/fsckvol
```

Note that the command to `fsck` a UNIX File System volume is the same, whether or not it had been mounted with logging. Nonetheless, because the volume had been populated using different UNIX File System mount options, the full `fsck` times for UNIX File System and UNIX File System with logging differ.

After the `fsck`, the volume was wiped clean by performing a `mkfs` for the next volume type in the experiment. Note that although the `mkfs` options for UNIX File System do not differentiate between logging and non-logging variants (that is a mount-time option), we did not take a shortcut of bypassing `mkfs` when switching between UNIX File System and UNIX File System with logging tests.

1.5 Results

This sub-section presents the results for the experiments previously described.

1.5.1 E4000 Configuration Results

Full fsck times for UNIX File System, UNIX File System with logging, and VERITAS File System on the E4000 configuration are summarized in Figure 1. VERITAS File System fsck runs about 760 percent faster than UNIX File System and about 690 percent faster than UNIX File System with logging. Note that VERITAS File System spends less time on CPU activities than its UNIX counterparts. However, because the VERITAS File System runs completed in much less time than the UNIX runs, CPU time *as a fraction of its respective run time* is much higher for VERITAS File System. Higher relative CPU utilization indicates that VERITAS File System is better positioned than UNIX File System to take advantage of faster CPUs. In contrast, UNIX File System fsck spends most of its time in I/O activity and thus cannot benefit much from faster CPUs.

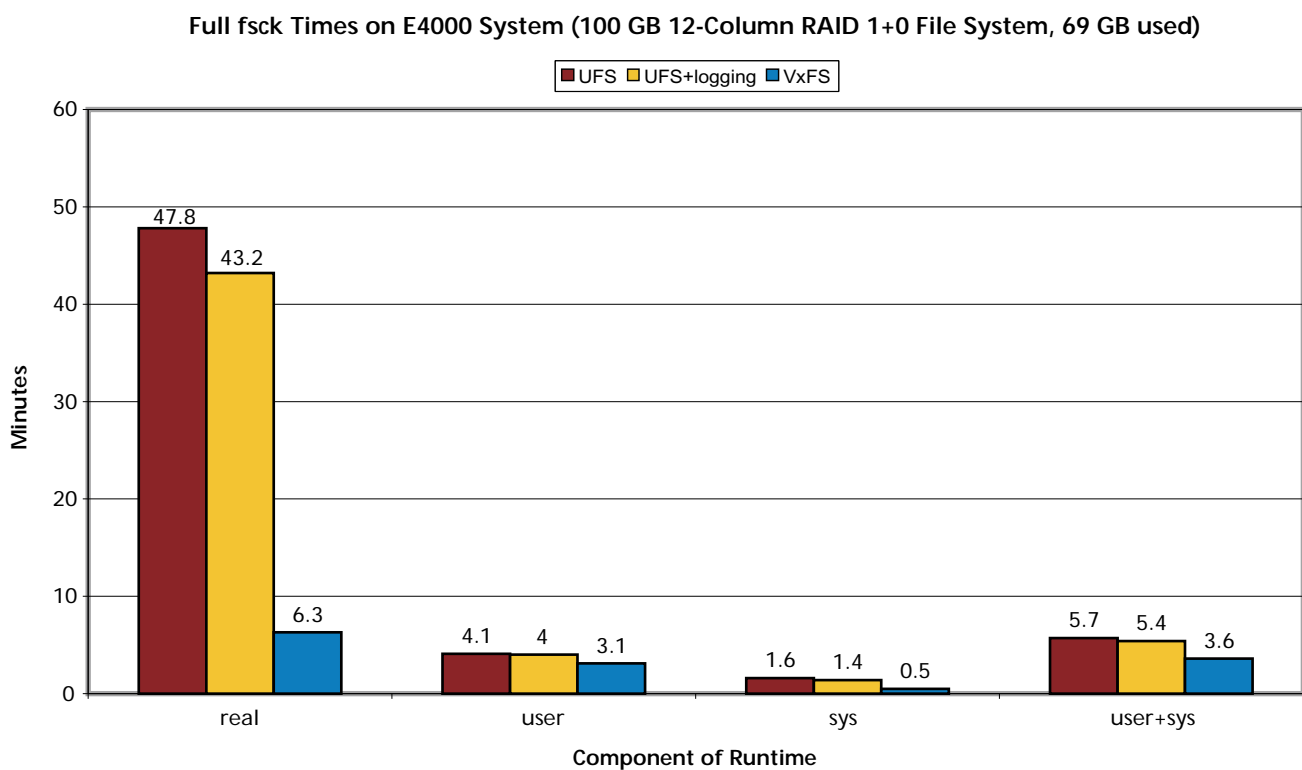


Figure 1: Full fsck on E4000 (69 GB used, on a 100 GB file system). VERITAS File System full fsck is 760 percent faster than UNIX File System and 690 percent faster than UNIX File System with logging. VERITAS File System also spends less total time on CPU activity than UNIX File System and UNIX File System with logging.

1.5.2 Blade 1000 Configuration Results

The time to perform a full fsck on the Blade 1000 configuration is summarized in Figure 2. The most important result arises from the full fsck time of UNIX File System without logging: about 23 minutes. This fsck time is too high to be acceptable and leads us to conclude that UNIX File System (without logging) should not be used in a high availability server.

In the atypical case where journaling file systems cannot replay their logs during a fsck, a full fsck is required. In such times, VERITAS File System is about 450 percent faster than UNIX File System and UNIX File System with logging. Also of interest is the time occupied by CPU activities. VERITAS File System is able to perform its fsck using less CPU activity (0.8 minutes) compared to UNIX File System and UNIX File System with logging (1.7 minutes). VERITAS File System spends about 20 percent of its (shorter) run time on CPU, while UNIX File System and UNIX File System with logging spend about 7 percent of their (longer) run times on CPU activity. Because a greater fraction of its run-time is occupied by CPU activities, VERITAS File System is much better positioned to take advantage of faster CPUs. By contrast, UNIX File System and UNIX File System with logging spend such a high fraction of their run times in I/O, their full fsck performance can improve only incrementally if given a faster CPU.

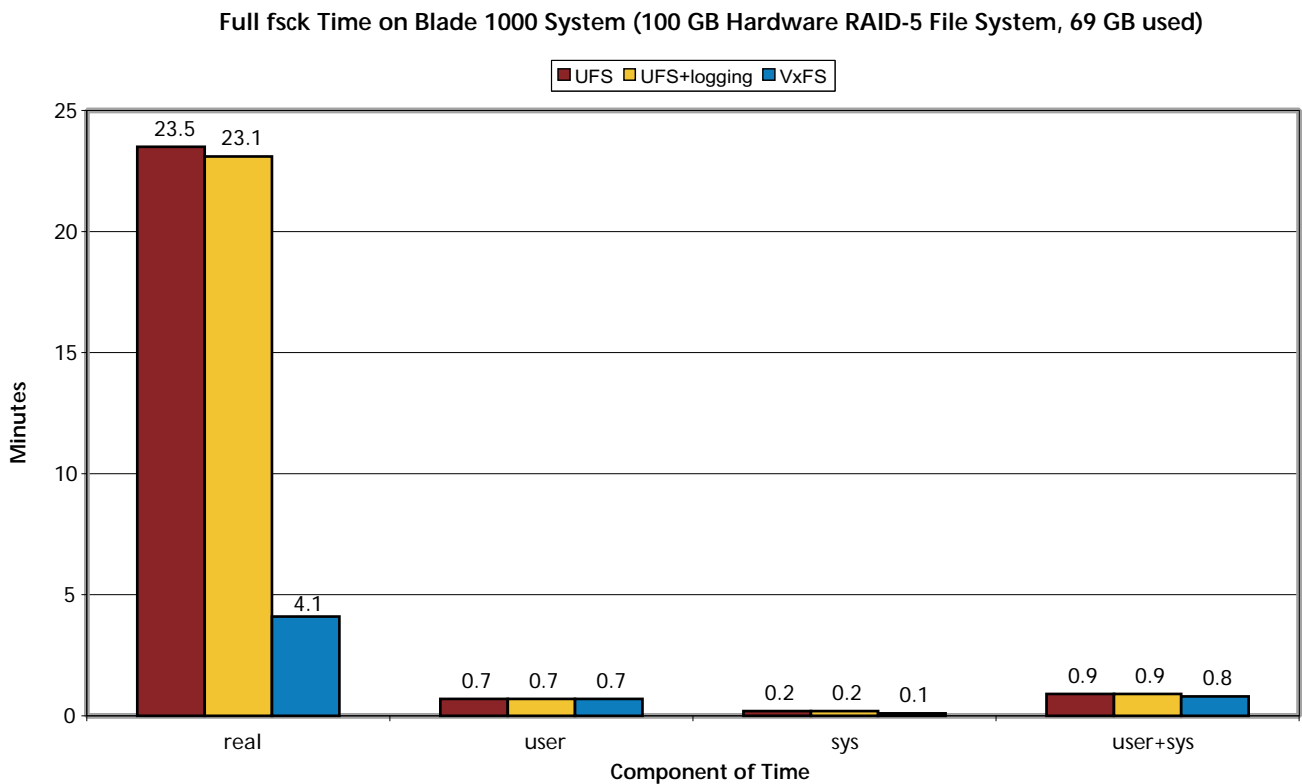


Figure 2: Full fsck on Blade 1000 (100 GB file system, 69 GB used). VERITAS File System full fsck is 450 percent faster than UNIX File System and UNIX File System with logging. VERITAS File System also spends less total time on CPU activity.

1.5.3 E4500 Configuration Results

The time taken to perform a full fsck on the near-terabyte E4500 configuration is summarized in Figure 3. As with the Blade 1000, the near-terabyte file system exhibits prohibitive full fsck performance on UNIX File System: 4.8 hours. Again, we conclude that due to the fsck time, UNIX File System without logging should not be used in a high availability server having this sized file system.

In the unusual case where VERITAS File System and UNIX File System with logging need to perform a full fsck (rather than a log replay), the results show that VERITAS File System is about 770 percent faster than UNIX File System and about 760 percent faster than UNIX with logging. Also of interest is the time occupied by CPU activities. Across the three file systems, CPU activity is roughly the same. Consequently, the percentage of run time that is occupied by CPU time is much higher for VERITAS File System: 56 percent compared to 10 percent for UNIX File System and UNIX File System with logging. Because a greater fraction of its run-time is occupied by CPU activities, VERITAS File System is much better positioned to take advantage of faster CPUs. By contrast, UNIX File System and UNIX File System with logging spend such a high fraction of their run times in I/O, their full fsck performance can improve only incrementally if given a faster CPU.

Full fsck Time on E4500 Configuration (1 TB 64-Stripe RAID-0 file system, 768 GB used)

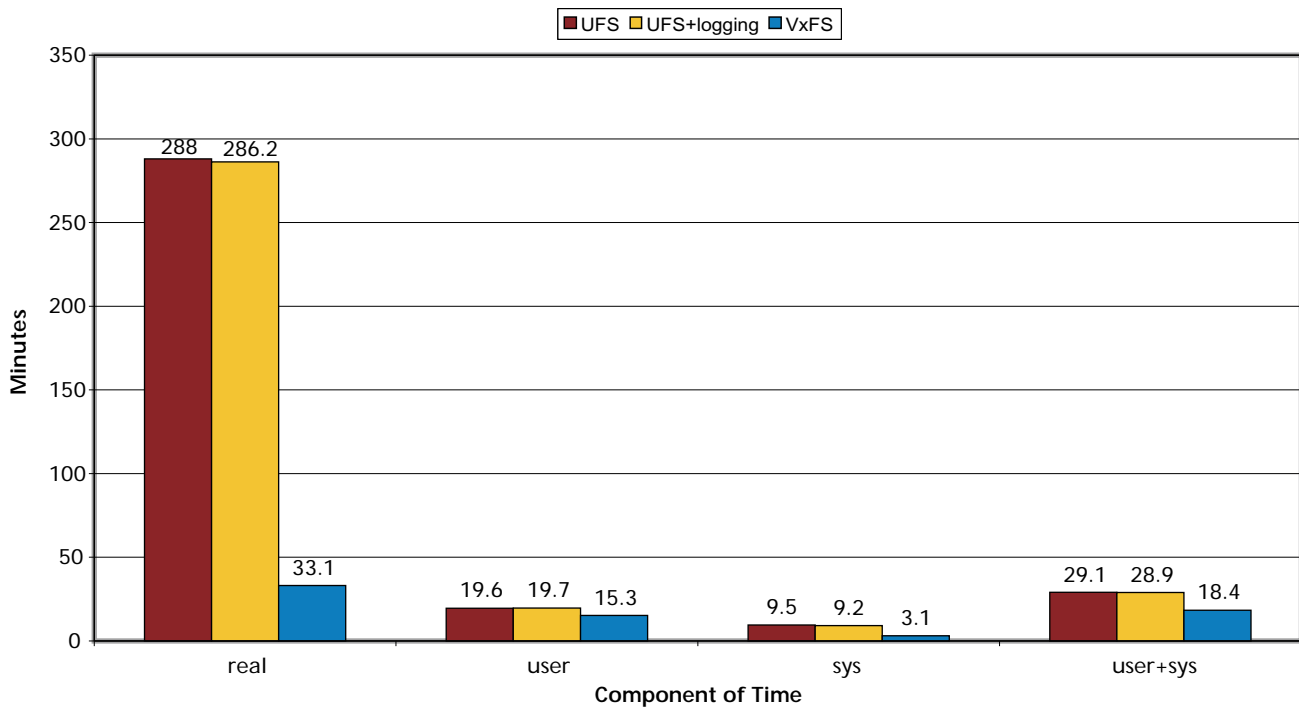


Figure 3: Full fsck on E4500 (1 TB file system, 768 GB used). VERITAS File System full fsck is about 770 percent faster than UNIX File System and about 760 percent faster than UNIX File System with logging. VERITAS File System also spends less total time on CPU activity.

2. Performance as an NFS Server (SPECsfs97 Benchmark)

2.1 Introduction

This section demonstrates the performance benefits of VERITAS File System Version 3.4 Patch 1 over Solaris 8 Update 4 UNIX File System in a network file system (NFS) environment. To evaluate the file system performance, a standardized benchmark suite, the Standard Performance Evaluation Corporation (SPEC) System File Server (SFS) suite sfs97, also known as SFS 2.0, was used. For the purpose of this comparison, only SPECsfs97 TCP and NFSv.3 were benchmarked. VERITAS File System outperformed UNIX File System by as much as 139 percent in SPECsfs97 peak throughput, with significantly faster server response time.

2.2 Test Configurations

An E6500 system was configured as an NFS server with two CPU and memory configurations: eight 400-MHz CPUs with 8 GB of memory (8x8) and 12 400-MHz CPUs with 12 GB of memory (12x12). The file systems that were measured were installed on Unisys Clariion arrays, totaling 198 disks. The disks were 9-GB and 18-GB Seagate Fibre Channel 10,000 RPM drives. The arrays were attached to 12 Sun Sbus Social HBAs. For each test, 11 file systems were created across the 198 disks, with RAID-1+0 volume layout of 9 striping columns. The VERITAS file systems were created with a 1 K block size and a 16-MB log. UNIX file systems were created with the default options.

VERITAS file systems were mounted with large file support and delayed logging (to match the semantics of UFS with logging). UNIX file systems were mounted either with no options or with logging.

The 14 clients used were Sun Microsystems Netra T1 with one 400-Mhz CPU, and 256 MB of memory. A Cisco 100/1000BaseT Network switch (Catalyst 3500XL) was used to network the clients (via 100BaseT interface) and the NFS server (via 1000BaseSX interface/Fiber-optic cable). For NFS services, we started the server with 1600 NFS server threads as recommended by Adrian Cockcroft and Richard Pettit, well-known technical authors in Sun performance tuning.

2.3 Overview of Results

Table 3 shows the improvement in peak throughput that VERITAS File System obtained over UNIX File System and UNIX File System with logging. VERITAS File System provided 34 percent to 36 percent greater peak throughput than UNIX File System and 95 percent to 139 percent greater peak throughput than UNIX File System with logging.

Configuration	VERITAS File System improvement over:	
	UNIX File System	UNIX File System + logging
8x8	36%	95%
12x12	34%	139%

Table 3: Increase in SPECsfs97 peak throughput obtained by VERITAS File System.

2.4 Detailed Results

Table 4 and Table 5 show detailed results of each benchmark run with different UNIX File System and VERITAS File System mount options. The results show that VERITAS File System had lower CPU utilization than UNIX for the 8x8 and 12x12 configurations. Lower CPU utilization enabled better scalability and higher peak throughputs. The disk utilization at peak throughputs was similar for all tests, ranging from 20 percent to 40 percent.

As discussed in Section 2.3, VERITAS File System provides peak bandwidth that is 34 percent to 36 percent greater than UNIX File System and 95 percent to 139 percent greater than UNIX File System with logging. Despite enabling a higher load on the server, VERITAS File System provided for overall response time (ORT) that is 20 percent to 36 percent faster than UNIX File System and 22 percent to 26 percent faster than UNIX File System with logging. (The ORT provides a measurement of how the system responds under an average load.) In other words, not only does VERITAS File System allow the server to do much more work (as measured by throughput), it also provides a faster turnaround time for clients (as measured by ORT).

A comparison of the 12x12 runs for UNIX File System with logging reveals little improvement in peak bandwidth compared to the 8x8 configuration (5791 for the 12x12 configuration and 5610 for the 8x8 configuration). We conclude that UNIX File System with logging does not scale well for SPECsfs97; with an additional four CPUs and 4 GB of RAM, peak throughput improved only 3 percent.

UNIX File System			UNIX File System + Logging			VERITAS File System		
Ops/sec	Msec/op	% CPU	Ops/sec	Msec/op	% CPU	Ops/sec	Msec/op	% CPU
483	3.4	4	482	3.3	5	484	2.2	4
982	3.8	8	982	3.0	9	982	2.7	7
1979	4.3	19	1981	4.7	20	1979	3.1	14
3011	5.1	30	3001	5.5	33	3011	3.7	21
4002	5.8	42	4008	6.8	46	4000	4.0	27
4999	6.8	52	5004	7.9	58	4997	4.6	32
5988	7.5	65	5610	29.2	66	5987	5.0	38
6961	8.9	79				6955	5.4	45
7927	13.9	95				7934	6.1	53
8060	21.9	99				8500	6.8	59
						9032	7.8	66
						9521	8.6	71
						10046	9.7	77
						10595	11.6	86
						10919	13.2	92
						10942	15.5	97
ORT (Msec/op)		6.5	ORT (Msec/op)		6.6	ORT (Msec/op)		5.4

Table 4: SPECsfs97 statistics for 8x8 configuration. In the peak throughput rows (shown in bold), VERITAS File System achieved 36 percent greater throughput than UNIX File System and 95 percent greater throughput than UNIX File System with logging. ORT measurements show that VERITAS File System services requests about 20 percent faster than UNIX File System and about 22 percent faster than UNIX File System with logging.

UNIX File System			UNIX File System + Logging			VERITAS File System		
Ops/sec	Msec/op	% CPU	Ops/sec	Msec/op	% CPU	Ops/sec	Msec/op	% CPU
483	3.5	3	482	4.2	3	483	2.7	3
982	3.4	6	982	3.4	6	981	2.2	5
1977	4.2	13	1980	3.4	14	1978	2.5	9
3007	4.0	20	3002	4.8	22	3000	2.9	14
3989	5.0	27	4003	5.7	29	3991	3.5	18
5002	6.1	33	4994	7.2	37	4994	3.8	22
5987	7.1	41	5791	26.1	45	5984	3.8	26
6954	7.7	48				6948	4.4	31
7947	8.7	58				7914	4.7	35
8508	9.5	66				8484	4.9	38
9032	11.0	74				8994	5.5	41
9553	11.9	80				9499	5.6	46
10037	13.5	89				10047	5.9	48
10353	17.1	95				11000	6.5	56
						12033	7.7	65
						12997	9.3	78
						13851	12.3	87
ORT (Msec/op)		6.8	ORT (Msec/op)		6.3	ORT (Msec/op)		5.0

Table 5: SPECsfs97 statistics for 12x12 configuration. In the peak throughput rows (shown in bold), VERITAS File System achieved 34 percent greater throughput than UNIX File System and 139 percent greater throughput than UNIX File System with logging. ORT measurements show that VERITAS File System services requests about 36 percent faster than UNIX File System and about 26 percent faster than UNIX File System with logging. Also note that the peak bandwidth obtained by UNIX File System with logging is not much higher than in the 8x8 configuration, indicating that systems that use UNIX File System with logging do not scale well.

Figure 4 and Figure 5 illustrate the bandwidth and response time limits for the various file systems, for the 8x8 and 12x12 machine configurations, respectively.

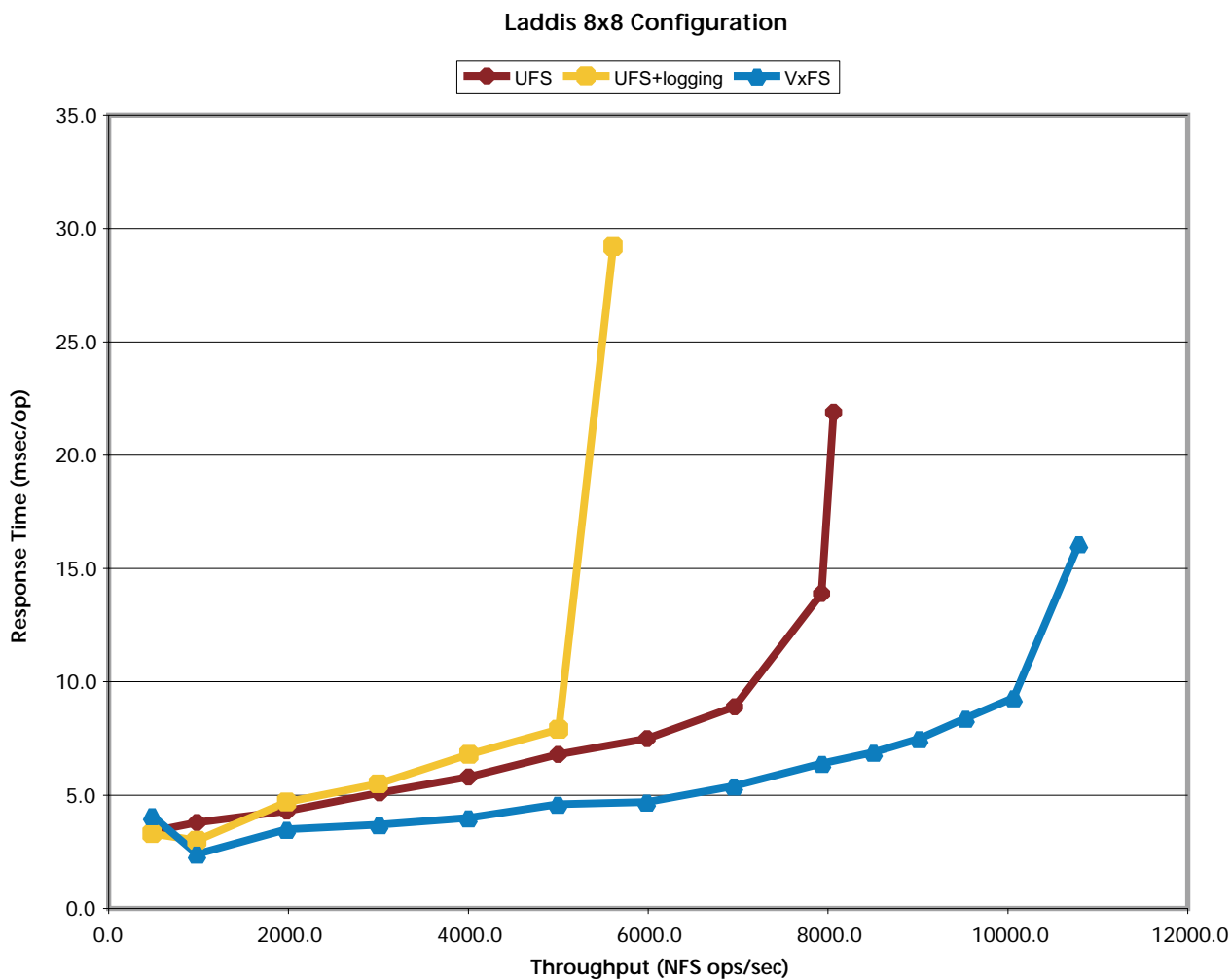


Figure 4: Throughput and response times, 8x8 configuration. UNIX File System with logging is unable to achieve 6,000 NFS ops/sec and sees significant response time degradation after about 5,000 NFS ops/sec. UNIX File System without logging is unable to scale much beyond 8,000 NFS ops/sec and suffers significant response time penalties in throughputs greater than about 7,000 NFS ops/sec. VERITAS File System, in contrast, is able to achieve 10,000 NFS ops/sec before response time begins to increase at a significant rate.

Laddis 12x12 Configuration

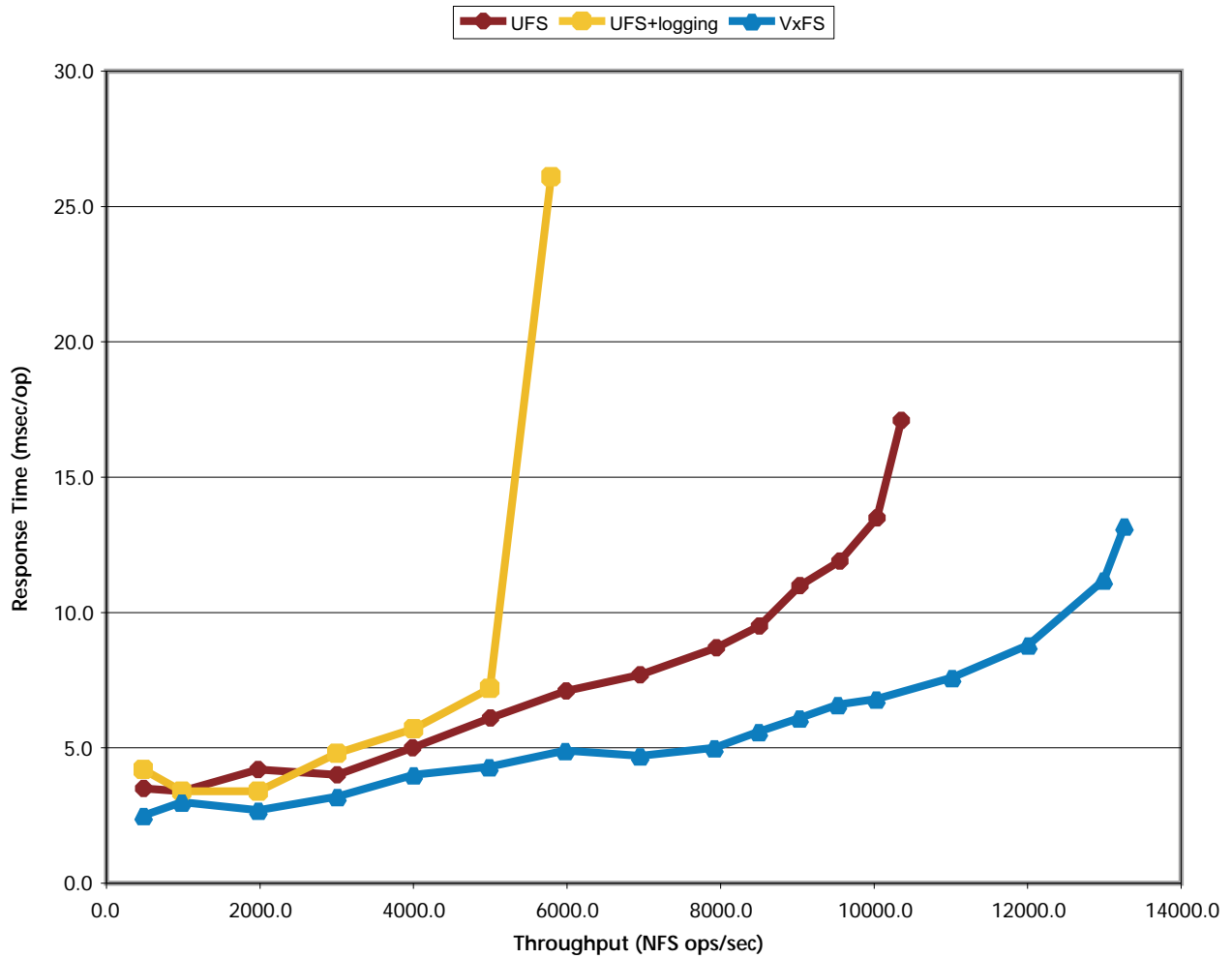


Figure 5: Throughput and response times, 12x12 configuration. As with the 8x8 configuration, UNIX File System with logging is unable to achieve 6,000 NFS ops/sec and sees an explosion of response time after about 5,000 NFS ops/sec. We conclude that UNIX File System with logging is unable to scale to greater numbers of CPUs or greater RAM. UNIX File System without logging is unable to scale much beyond 10,000 NFS ops/sec and suffers significant response time penalties in throughputs greater than that. VERITAS File System, in contrast, is able to achieve about 13,000 NFS ops/sec before response time begins to increase at a significant rate.

For reference, Table 6 shows the workload distribution of a typical SFS benchmark run.

NFS Op	Percent
getattr	11
setattr	1
lookup	27
readlink	7
read	18
write	9
create	1
remove	1
readdir	2
fsstat	1
access	7
commit	5
fsinfo	1
readdirplus	9

Table 6: SFS2 workload distribution

2.5 SPECsfs97 Defects and Our Use of the Benchmark

SPEC has announced that it has identified defects within the SPECsfs97 benchmark and that it is no longer publishing results of that benchmark. As of the time of the writing of this paper, SPEC has not published a replacement benchmark. The majority of the defects in SPECsfs97 revolve around the changes in the file working set with changes in the total number of processes used (clients times processes). In this paper, we used the same number of clients and processes for all benchmarks. We also used the same clients, NFS server and network, with the only changes made to the file system being benchmarked. We feel that with these precautions, the SPECsfs97 benchmark provides a valid means of comparing file systems. Full disclosure of these SPECsfs97 benchmark runs can be found in Appendix A.

3. Performance in an Oracle Environment (TPC-C Benchmark)

3.1 Introduction

This section describes the performance of VERITAS File System™ version 3.4 Patch 1 and Sun UNIX File System while running an online transaction processing (OLTP) workload on Solaris 8 Update 4. Typical OLTP systems involve processing simple to moderately complex transactions with multiple updates to the database. The benchmark used for this performance comparison was derived from the commonly known TPC-C benchmark. This document contrasts the performance of available I/O configurations offered by VERITAS File System and UNIX File System.

The results of this study show:

- For configurations that use the operating system's page cache, VERITAS File System achieves 20 percent greater tpm-C throughput than UNIX File System and 74 percent greater tpm-C throughput than UNIX File System with logging.
- For configurations that bypass the operating system's page cache, the Quick I/O feature of VERITAS Database Edition and UNIX File System Concurrent Direct I/O (CDIO, also known as Database Direct I/O), VERITAS File System achieves 8 percent and 9 percent greater tpm-C throughput than UNIX File System and UNIX File System with logging, respectively. The Cached Quick I/O feature of VERITAS Database Edition increases the tpm-C advantage over UNIX File System and UNIX File System with logging to 20 percent and 21 percent, respectively.
- The Quick I/O feature of VERITAS Database Edition with Data Full Checkpoint realizes a 103 percent tpm-C throughput improvement over UNIX File System Concurrent Direct I/O with Snapshot, and a 110 percent tpm-C improvement over UNIX File System with logging Concurrent Direct I/O with Snapshot.

3.2 Test Configurations

Tests were run on a Sun E6500 computer system, with eight 400MHz CPUs, 8 GB of RAM and six Sbus controllers. Six Unisys Clariion JBOD arrays yielded 48 18-GB Seagate Fibre Channel 10,000 RPM disks used in the TPC-C runs.

The TPC-C data was housed on a 20-way volume, configured to a striped-mirrored (RAID-1+0) layout using VERITAS Volume Manager™ 3.1.1. The volume was connected across two Sun Sbus Social HBAs. The Oracle redo logs were placed on a two-way, striped, mirrored volume on a separate Sun Social controller. The Oracle executables and benchmark code resided on a separate internal disk outside of the volume group under test.

The following software releases were used in testing:

- Oracle 8.1.7 (32-bit)
- Solaris 8 Update 4
- VERITAS File System 3.4 Patch 1
- VERITAS Volume Manager 3.1.1

The following VERITAS File System configurations were tested:

- Buffered I/O
- Quick I/O
- Cached Quick I/O
- Data Full Checkpoint

The following UNIX File System configurations were tested:

- UNIX File System with and without logging
- Concurrent Direct I/O
- Snapshot

For a VERITAS File System Checkpoint and UNIX File System Snapshot comparison, a 25-GB volume was created on the same disks as the TPC-C data disks for consistency with how VERITAS File System creates Data Full Checkpoints. The creation of the UNIX File System Snapshot volume took six minutes. VERITAS File System Data Full Checkpoint creation is practically instantaneous.

The VERITAS file systems were created with 1 K blocks, a 16-MB log and large file support. The UNIX file systems were created with default parameters.

The VERITAS file systems were mounted with delayed logging (to match the semantics of UNIX's logging) and largefile options. The UNIX file systems were mounted with default options, with the following exceptions. UNIX File System with logging runs use the logging mount option, and Concurrent Direct I/O runs use the `forcedirectio` option. In addition, Concurrent Direct I/O runs add the following line to Oracle's initialization file:

```
_filesystemio_options = setall
```

We configured Oracle to use a 2-GB SGA, which is the maximum available in a 32-bit environment. Remaining memory is available for the Quick I/O feature of VERITAS Database Edition.

The database used in the test was 36 GB over a total of 52 Oracle data files, including redo logs, indexes, rollback segments and temporary and user tablespaces. The database was sized as a fully scaled TPC-C database with a scale factor of 200 warehouses.

Additional software configuration information (shared memory and IPC settings as well as the Oracle parameter file) used in this benchmark is contained in Appendix B.

3.3 Results

Table 7 shows the performance improvement realized by VERITAS File System over UNIX File System and UNIX File System with logging for file system configurations that use the operating system's page cache. VERITAS File System improves on UNIX File System and UNIX File System with logging's peak tpm-C throughput by 20 percent and 74 percent, respectively. Figure 6 shows the data in chart form.

VERITAS File System improvement over:	
UNIX File System	UNIX File System + Logging
20%	74%

Table 7: Comparing file system performance when using buffered I/O. VERITAS File System with buffered I/O shows a 20 percent tpm-C improvement compared to UNIX File System with buffered I/O and 74 percent tpm-C improvement compared to UNIX File System with logging and buffered I/O.

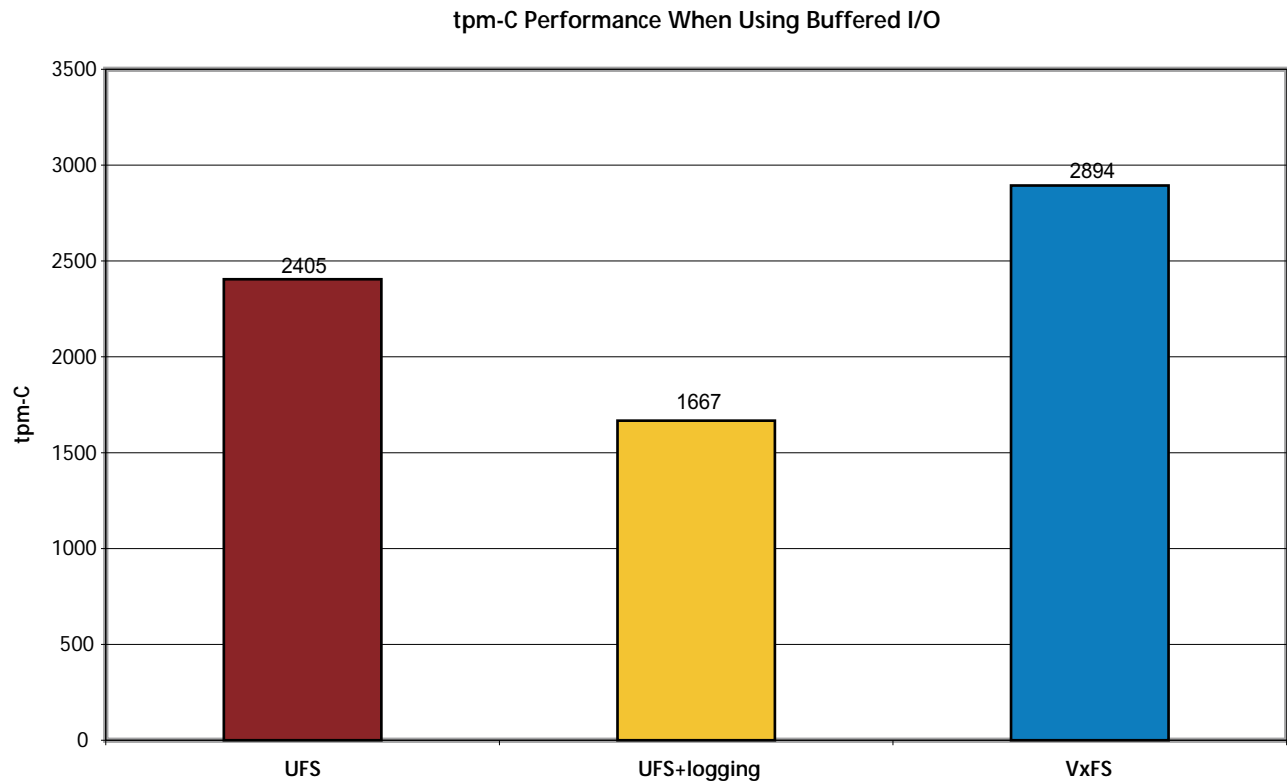


Figure 6: Peak tpm-C for file system configurations that use the operation system's page cache.

Table 8 shows the performance of file system configurations that bypass the operating system's page cache, the Quick I/O feature of VERITAS Database Edition and UNIX File System Concurrent Direct I/O. The Quick I/O feature of VERITAS Database Edition outperforms UNIX File System Concurrent Direct I/O by about 8 percent, and UNIX File System with logging Concurrent Direct I/O by about 9 percent. The Cached Quick I/O feature of VERITAS Database Edition outperforms UNIX File System Concurrent Direct I/O by about 20 percent and UNIX File System with logging Concurrent Direct I/O by about 21 percent.

	Improvements from using VERITAS Database Edition Quick I/O	Improvements from using VERITAS Database Edition Cached Quick I/O
UNIX File System Concurrent Direct I/O	8%	20%
UNIX File System + Logging Concurrent Direct I/O	9%	21%

Table 8: Peak tpm-C improvements gained by using VERITAS File System, for configurations that bypass the operating system page cache (UNIX File System with Concurrent Direct I/O and the Quick I/O feature of VERITAS Database Edition).

Table 9 shows the performance of these file system configurations that bypass the page cache, when augmented with VERITAS File System Data Full Checkpoint and UNIX File System Snapshot technology. In these configurations, the Quick I/O feature of VERITAS Database Edition outperforms UNIX File System Concurrent Direct I/O by 103 percent and UNIX File System with logging Concurrent Direct I/O by 110 percent.

	Improvements from using VERITAS Database Edition Quick I/O
UNIX File System Concurrent Direct I/O Snapshot	103 percent
UNIX File System + Logging Concurrent Direct I/O Snapshot	110 percent

Table 9: Peak tpm-C improvements gained by using VERITAS File System, for configurations that bypass the operating system page cache (UNIX File System with Concurrent Direct I/O and the Quick I/O feature of VERITAS Database Edition) and that use Snapshot/Checkpoint technology. The Quick I/O feature of VERITAS Database Edition with Storage Checkpoints improves on UNIX File System Concurrent Direct I/O and UNIX File System with logging Concurrent Direct I/O with Snapshot by 103 percent and 110 percent, respectively.

The data of Table 8 and Table 9 are illustrated in Figure 7 and Figure 8, respectively.

The data in Table 8 is illustrated in the following figure:

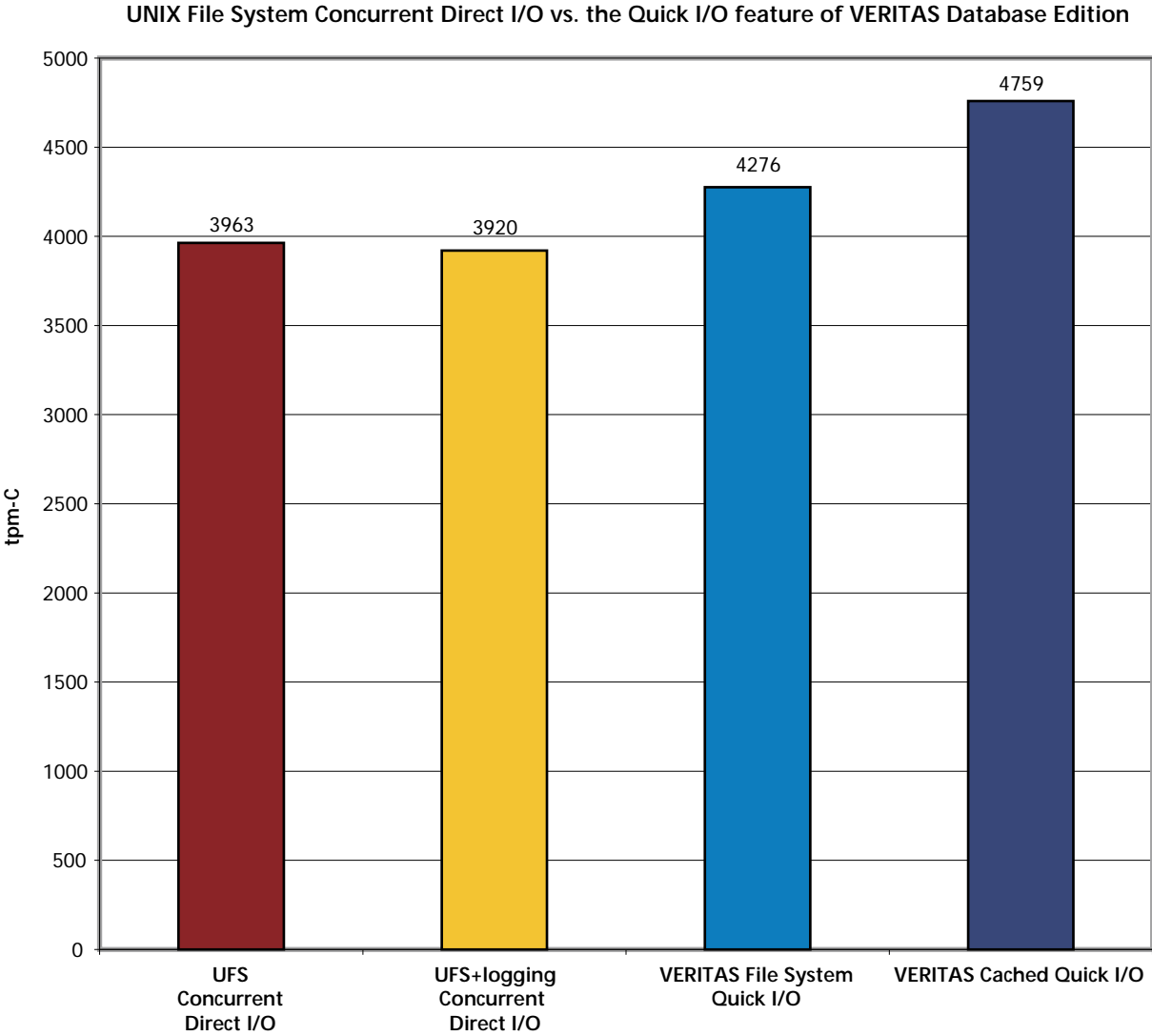


Figure 7: Peak tpm-C, using UFS Concurrent Direct I/O and the Quick I/O feature of VERITAS Database Edition.

The data in Table 9 is illustrated in the following figure:

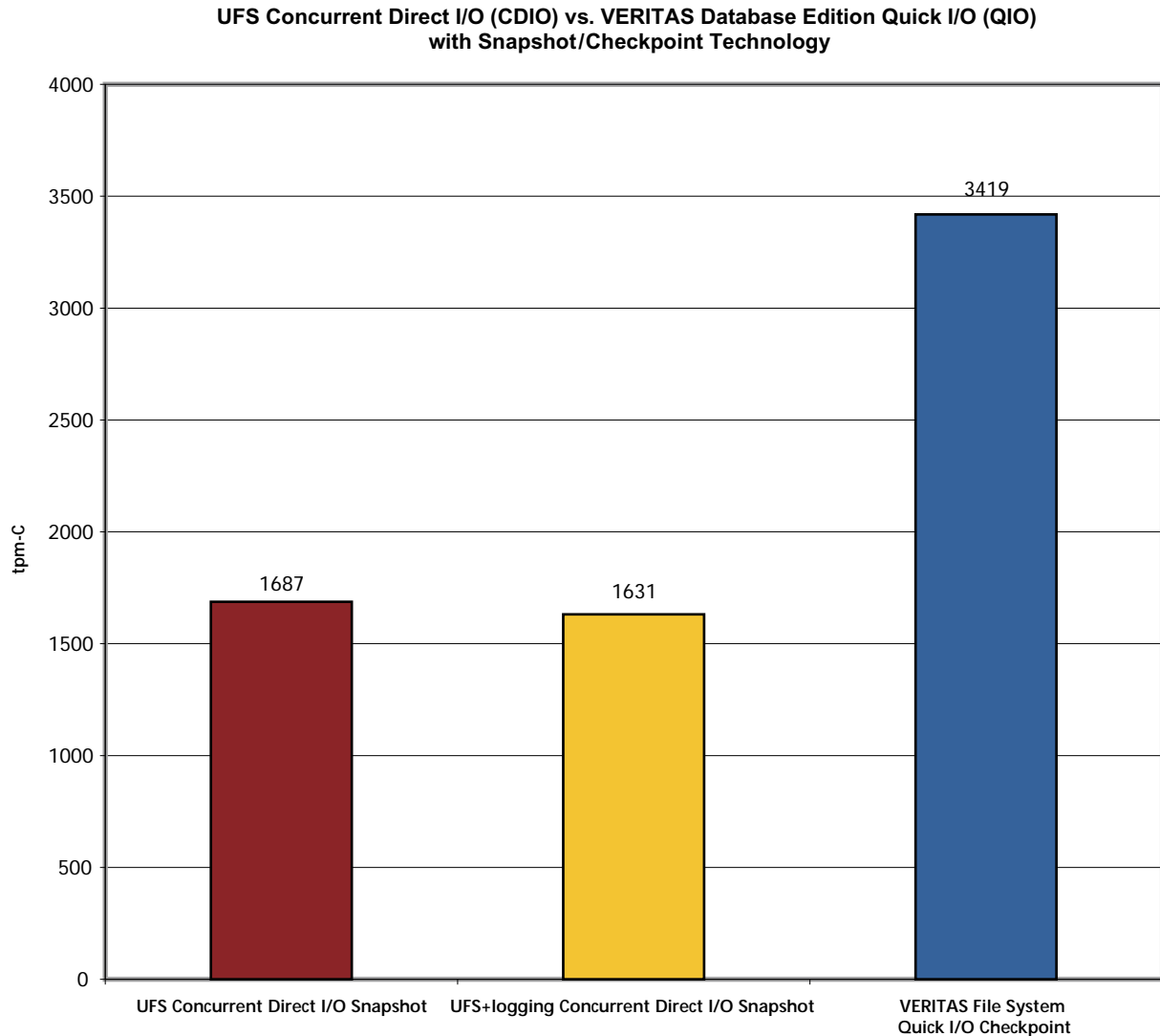


Figure 8: Peak tpm-C, using UNIX File System Concurrent Direct I/O and the Quick I/O feature of VERITAS Database Edition, in conjunction with Snapshot/Checkpoint technology. VERITAS File System with Data Full Checkpoint realizes a performance improvement of over 100 percent compared to UNIX File System with Snapshot.

To summarize, the preceding charts and graphs show that VERITAS File System buffered I/O realizes a 20 percent and 74 percent greater tpm-C throughput than UNIX File System and UNIX File System with logging's buffered I/O, respectively. When bypassing buffered I/O, the Quick I/O feature of VERITAS Database Edition outperforms UNIX File System and UNIX File System with logging Concurrent Direct I/O by about 8 percent and 9 percent in tpm-C throughput, respectively. VERITAS File System Cached Quick I/O, a feature of Database Edition is capable of realizing higher throughput than UNIX File System Concurrent Direct I/O and the Quick I/O feature of Database Edition in a 32-bit environment, if sufficient memory is available for caching. Finally, the Quick I/O feature of Database Edition with Data Full Checkpoint outperformed UNIX File System and UNIX File System with logging Concurrent Direct I/O with Snapshots by more than 100 percent.

4. Performance of Miscellaneous Commands (tar extract, cp, mkfile, touch)

4.1 Introduction

This report demonstrates the performance benefits of VERITAS File System Version 3.4 Patch 1 over UNIX File System on a Sun Microsystems Enterprise server running Solaris 8 Update 4. To evaluate the file system performance, four tests were used. Two are standard UNIX commands (mkfile and cp). The third (touch_files) creates a file (like the UNIX touch command) and writes the first block of the file. (The source for touch_files is contained in Appendix C.) The fourth test extracted large, compressed tar archives.

4.2 Test Configurations

4.2.1 mkfile, touch_files, cp

A Sun E4500 was configured with four 400-MHz CPUs and 4 GB of memory (4x4). The file systems tested were set up on an IBM ESS 2105 F-20 (Shark) with 128 disks. The Shark was configured as two volumes of RAID-5 and two volumes of RAID-0. The disks are 35-GB IBM 10,000 RPM drives connected by fibre to Brocade switches through the two Shark cluster AIX systems. The Shark is connected to one Brocade Silksworm switch that is cascaded to a second Brocade Silksworm switch and then to a JNI card in the E4500. The JNI card is a model FC64-1063-N. Of the 128 disks, 64 are used as Shark RAID-0 and 64 are used as Shark RAID-5 (7 + 1 hot standby). The file systems are created on VERITAS Volume Manager™ stripe-mirror (RAID-1+0) volumes. The stripe-mirror 225 GB volumes over RAID-0 are seven columns (14 disks). The stripe-mirror 225 GB volumes over RAID 5 volumes are two columns. The VERITAS File System file systems were mounted with delayed logging to match the semantics of UNIX File System with logging.

4.2.2 Uncompress and tar Extract

These tests were performed on two systems used for fsck performance comparisons in Section 1. The first system is a Sun Blade 1000, with two 750-MHz UltraSPARC-III CPUs and 1 GB of RAM. A single Sun StorEdge T3 hardware RAID-5 array, directly connected to a built-in Fibre Channel-Arbitrated Loop port on the Blade, was configured to a 100 GB volume. The second system is a Sun E4500, with eight 400-MHz UltraSPARC-II CPUs and 2 GB of RAM. Three Sun StorEdge A5200 JBOD arrays provided 64 18 GB disks, configured for a striped volume (RAID-0) of 1 TB.

Two file sets were used in this study. For the Blade 1000 configuration, the file set consists of five compressed (using gzip) archive (tar) files. Each .tar.gz file represents distinct subsets of the data produced by a run of the SPECsfs97 benchmark. After each file is uncompressed and extracted, the files total 72,641,776 KB of data (about 69.3 GB). The five .tar.gz files were each about 31 MB. For the E4500 configuration, a larger (though similar) file set was used. First, the five top-level SPECsfs97 directories were brought into a single .tar file that, when compressed using gzip, is 164 MB. Then, 11 copies of this .tar.gz file were created. When uncompressed and extracted into different directories on the same file system, the file set totals about 768 GB.

The 69.3 GB and 768 GB file sets are the same that were used in the full fsck study of Section 1. (That section contains a complete description of the file sizes.) In addition, the machine and disk configurations for the Blade 1000 and E4500, as well as the mkfs and mount options that were used to create the file systems are the same as described in Section 1.

After mounting, the file system was populated with the .tar.gz files. The files in the 100-GB file system were extracted using the following script (the 1 TB file system uses a similar script):

```
gzcat laddis5.tar.gz | tar xf - &
gzcat laddis6.tar.gz | tar xf - &
gzcat laddis7.tar.gz | tar xf - &
gzcat laddis8.tar.gz | tar xf - &
gzcat laddis9.tar.gz | tar xf - &
gzcat laddis10.tar.gz | tar xf - &
/bin/time wait
```

After an experiment completed on a given file system, the volume was wiped clean by performing a mkfs for the next volume type in the experiment. Note that although the mkfs options for UNIX File System do not differentiate between logging and non-logging variants (that is a mount time option), we did not take a shortcut of bypassing mkfs when switching between UNIX File System and UNIX File System with logging tests.

4.3 Overview of Results

The performance improvement realized by using VERITAS File System over UNIX File System and UNIX File System with logging varies by command and disk configuration. Table 10 through Table 13 summarize the results.

mkfile file size	RAID-0		RAID-5	
	VERITAS File System improvement over:			
	UNIX File System	UFS + Logging	UNIX File System	UFS + Logging
1 GB	98%	97%	121%	117%
2 GB	76%	81%	106%	105%
4 GB	61%	68%	73%	83%
8 GB	48%	58%	73%	78%

Table 10: VERITAS File System performance improvements for mkfile benchmark.

directories/files per directory	SINGLE PROCESS		MULTIPROCESS (100 parallel processes/directory)	
	VERITAS File System improvement over:			
	UNIX File System	UFS + Logging	UNIX File System	UFS + Logging
100/1000 RAID-0	79%	19%	45%	338%
100/1000 RAID-5	67%	21%	57%	310%

Table 11: VERITAS File System performance improvements for touch_files benchmark. Of special interest is the performance of UNIX File System with logging on the multiprocess benchmark. In this case, VERITAS File System ran more than 300 percent faster than UNIX File System with logging.

source, destination	1:1		1:5 parallel		1:10 parallel	
	VERITAS File System improvement over:					
	UFS	UFS + Logging	UFS	UFS + Logging	UFS	UFS + Logging
RAID-0, RAID-5	32%	34%	23%	30%	18%	26%
RAID-0, RAID-5	33%	52%	15%	19%	15%	21%
RAID-5, RAID-0	63%	80%	23%	38%	11%	19%
RAID-5, RAID-5	51%	79%	10%	24%	10%	23%

Table 12: cp Benchmark performance improvements for VERITAS File System

Sun Blade/T3 RAID-5, 69 GB on a 100-GB Volume Five Concurrent Uncompress/Extract Pipelines		Sun E4500/A5200 RAID-0, 768 GB on a 1-TB Volume 11 Concurrent Uncompress/Extract Pipelines	
VERITAS File System improvement over:		VERITAS File System improvement over:	
UNIX File System	UFS + Logging	UNIX File System	UFS + Logging
50%	69%	70%	84%

Table 13: Uncompress and tar extract performance improvements for VERITAS File System

4.4 Detailed Results

Detailed elapsed time results are shown in Figure 9 through Figure 14.

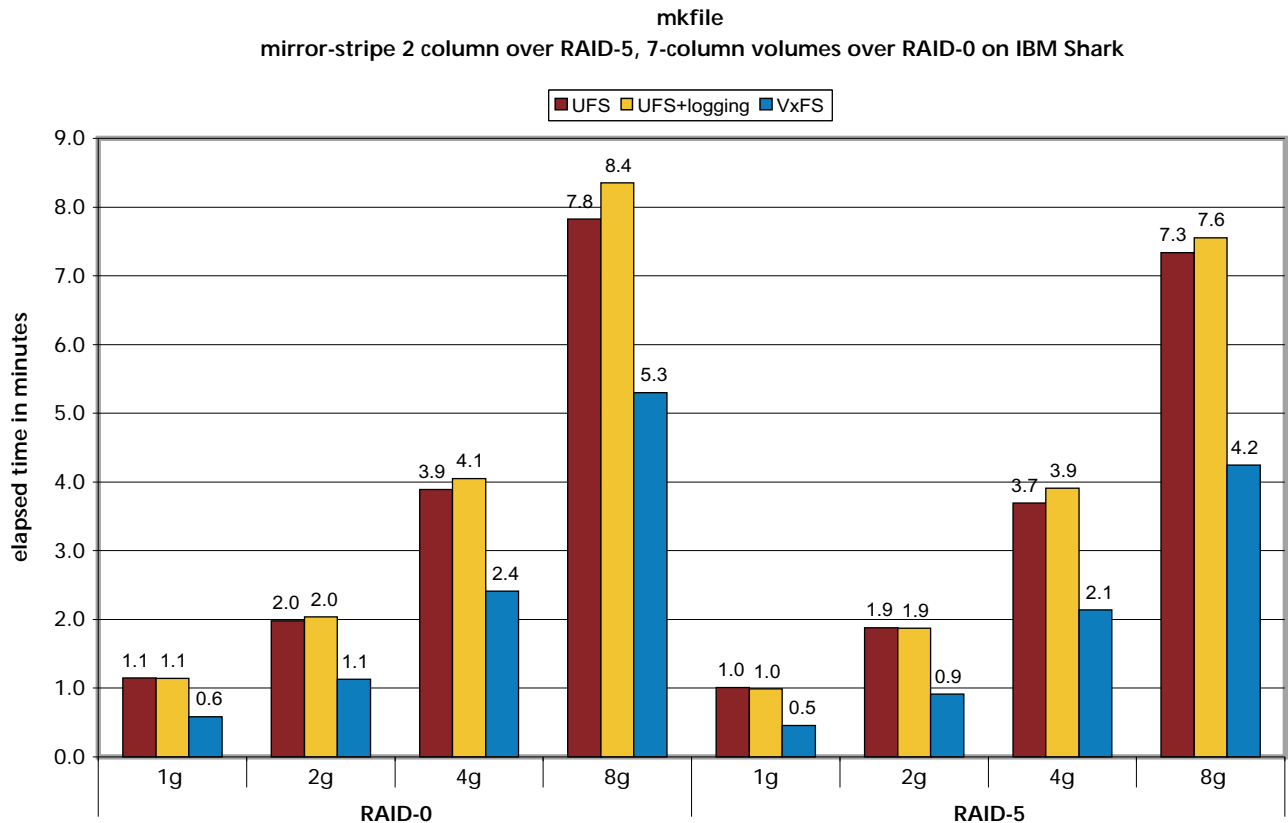


Figure 9: mkfile. VERITAS File System outperforms UNIX File System and UNIX File System with logging by 48 percent to 98 percent for RAID-0 configurations, and by 73 percent to 121 percent for RAID-5 configurations.

touch_files 100 1000 6144
 VERITAS Volume Manager stripe-mirror 7 columns over IBM Shark RAID-0 (2x7 disks)
 VERITAS Volume Manager stripe-mirror 2 columns over IBM Shark RAID-5 (2x[7+1 hot standby] disks)

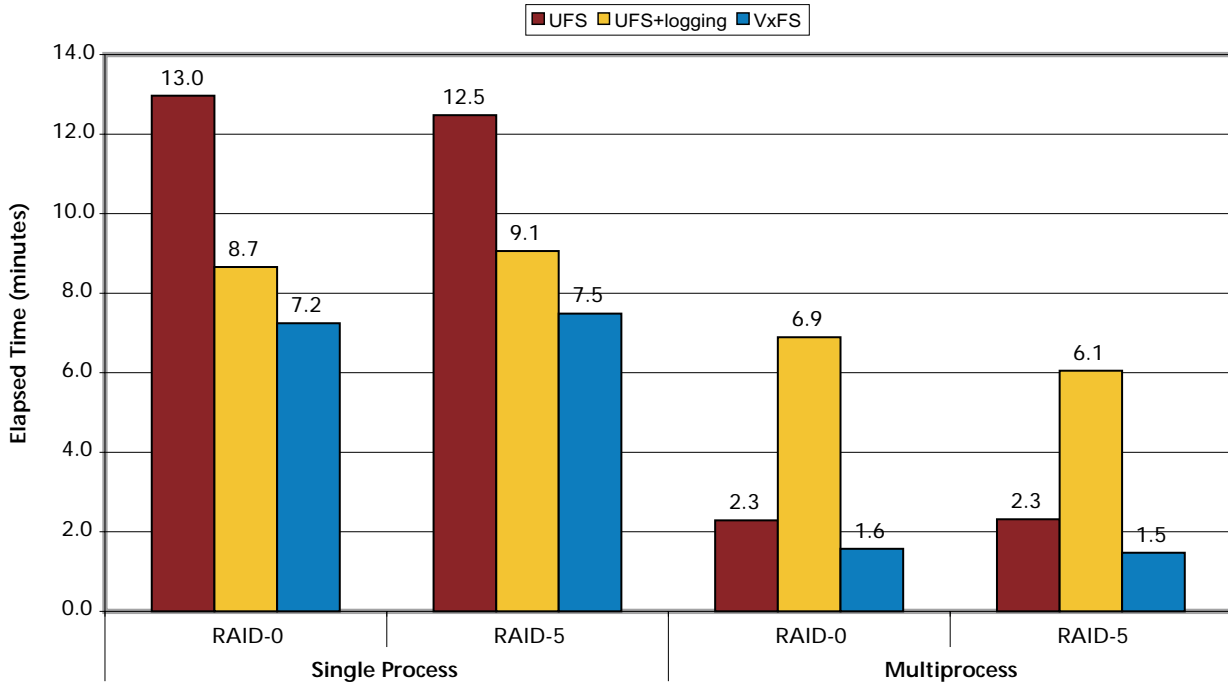


Figure 10: touch_files. Notice the poor performance of UNIX File System with logging on the multiprocess runs.

cp 4.5G file (1:1 copy, 1:5 copies, 1:10 copies)
 VERITAS Volume Manager stripe-mirror 7 columns over IBM Shark RAID-0 (2x7 disks)
 VERITAS Volume Manager stripe-mirror 2 columns over IBM Shark RAID-5 (2x[7+1 hot standby] disks)

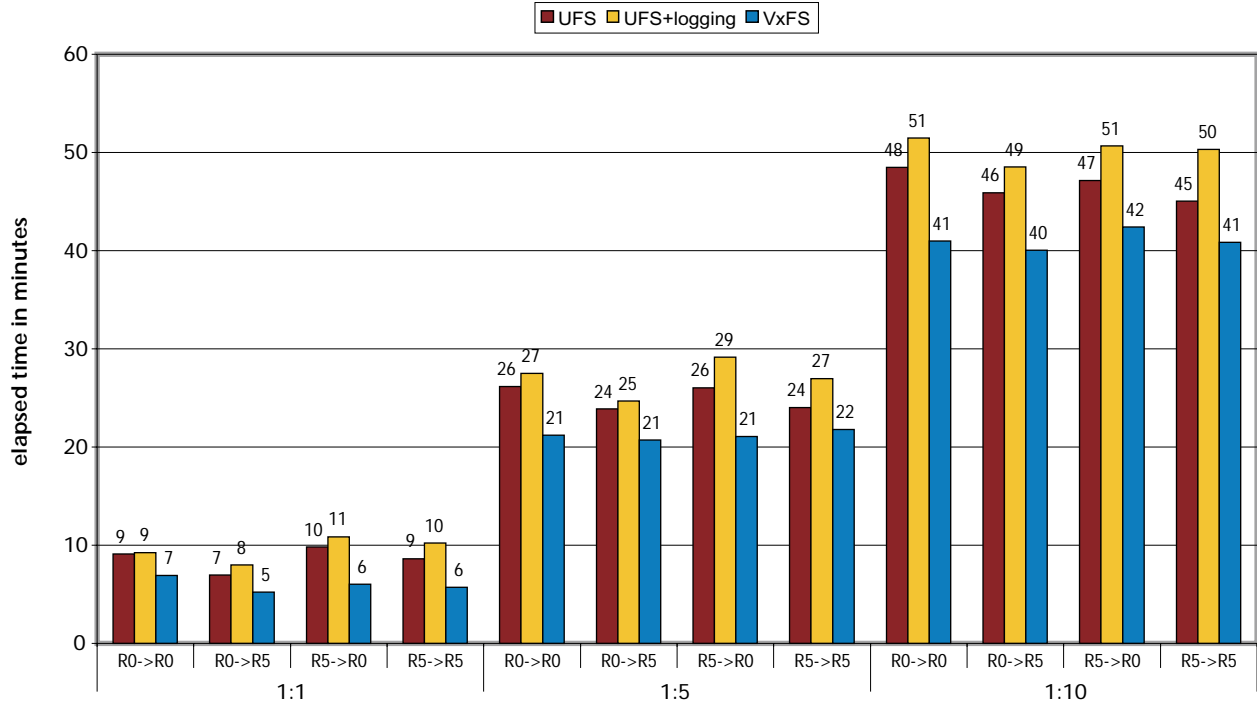


Figure 11: cp. VERITAS File System outperforms UNIX File System and UNIX File System with logging by 32 percent to 80 percent for 1-to-1 copy, by 10 percent to 38 percent for 1-to-5 parallel copy and by 10 percent to 26 percent for 1-to-10 parallel copy.

The time taken to uncompress and extract on the Blade 1000 system (about 69.3 GB of data on a 100 GB volume) is shown in Figure 12. Although the uncompress component (`gunzip`) involves significant CPU time, which is likely the same for each of the three file systems, the extract (`un-tar`) component differs enough to show a significant overall win for VERITAS File System. It is about 50 percent faster than UNIX File System and about 70 percent faster than UNIX File System with logging.

Time to Uncompress and Un-tar on Sun Blade 1000 (About 69 GB on a 100-GB Volume)

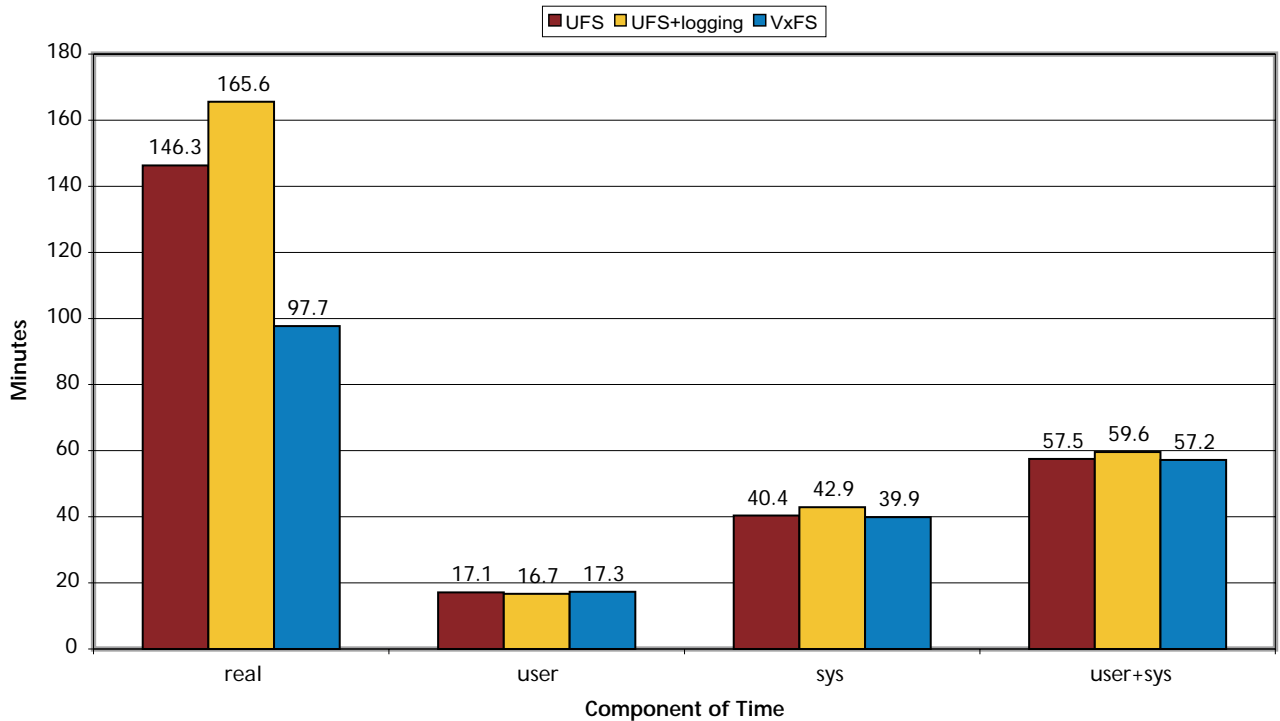


Figure 12: Uncompress and extract on Blade 1000. Note that the user and sys times are per-CPU. The user and system times as reported by `/bin/time`, in units of CPU-minutes, is twice the displayed value, because `/bin/time` reports the sum of CPU times across all processes.

The time to uncompress and extract on the E4500 configuration (about 768 GB used on a 1 TB file system) is summarized in Figure 13. The uncompress component involves significant CPU time which is likely the same for each file system. However, the extract component differs enough between the file systems to show a significant win for VERITAS File System. It is about 69 percent faster than UNIX File System and about 84 percent faster than UNIX File System with logging.

Time to Uncompress and Un-tar on E4500 (1 TB 64-Stripe RAID-0 File System, 768 GB Used)

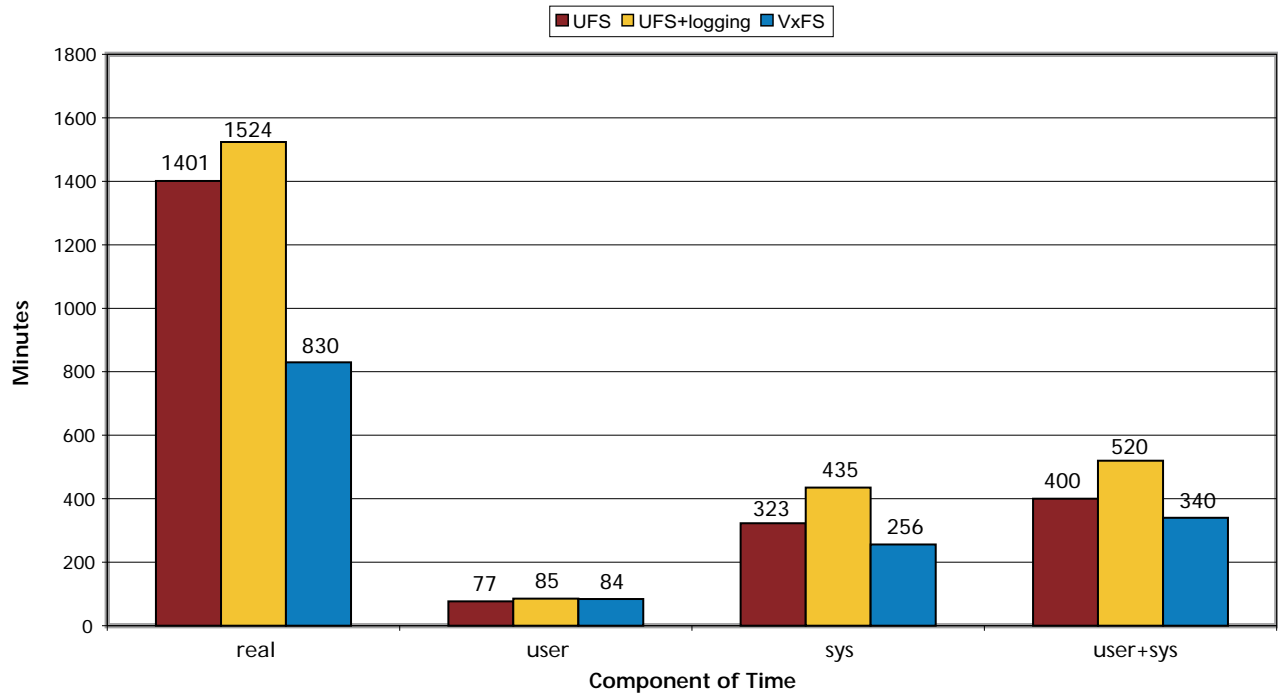


Figure 13: Uncompress and extract on E4500. Note that the user and sys times are per-CPU. The user and system times as reported by /bin/time, in units of CPU-minutes, is eight times the displayed values because /bin/time reports the sum of CPU times across all processors.

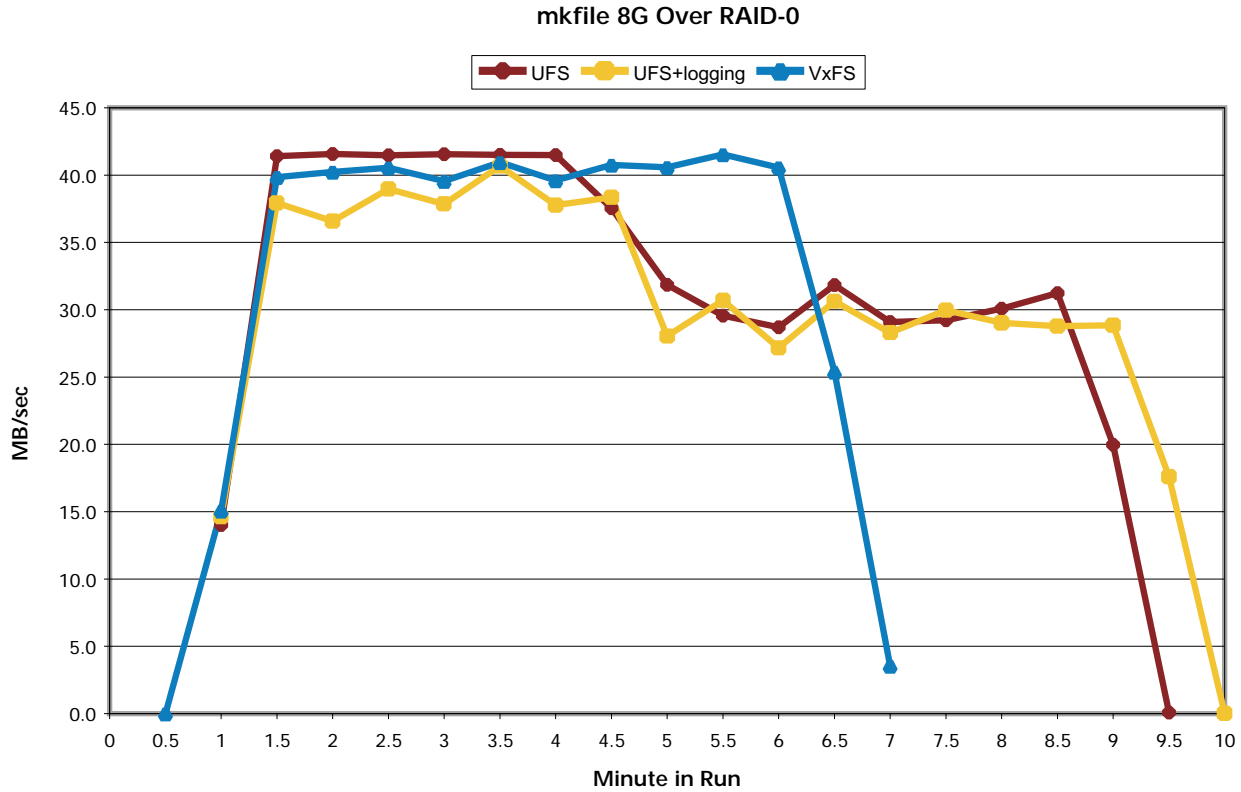


Figure 14: File System performance of mkfile over time. VERITAS File System obtains its peak bandwidth quickly and continues working at that rate until it completes the benchmark, in about seven minutes. UNIX File System and UNIX File System with logging initially reach approximately the same bandwidth as VERITAS File System, but fail to maintain that rate after four minutes. Consequently, the UNIX File System and UNIX File System with logging runs take longer to complete than VERITAS File System.

4.5 Summary

This study shows the performance advantages of VERITAS File System over Solaris 8 Update 4 UNIX File System. VERITAS File System outperformed UNIX File System by as much as 338 percent in these tests and enabled better system scalability. The commands examined and improvements realized by using VERITAS File System over UNIX File System and UNIX File System with logging, are:

- mkfile: 48 percent to 121 percent
- touch_files: 19 percent to 338 percent (VERITAS File System has a large advantage over UNIX with logging when performing multiprocess concurrent I/O.)
- cp: 10 percent to 80 percent
- uncompress/tar extract: 50 percent to 84 percent

5. Performance of Sequential File I/O (vxbench)

5.1 Introduction

This section evaluates the performance of high-bandwidth disk reads and writes for VERITAS File System™ Version 3.4 Patch 1 and Sun UNIX File System running on the Solaris 8 Update 4 operating system. The ability of a server to process large disk operations quickly, under high loads, can be a telling factor in the server's scalability.

This section compares read and write bandwidth that can be attained with UNIX File System, UNIX File System with logging and VERITAS File System file systems. By using a program called *vxbench*, we are able to measure the bandwidth of sequential disk operations under varying conditions:

- Reads versus writes
- I/O units (8 K vs. 64 K)
- Varying concurrency. Note that when several processes concurrently perform sequential reads and writes on different files, the workload presented to the file system as a whole is non-sequential.

We were able to draw several conclusions from these experiments:

- VERITAS File System scales very well under concurrent loads; UNIX File System and UNIX File System with logging do not. In particular, with 16 or 32 concurrent read or write operations, VERITAS File System often outperforms the UNIX File System variants by 300 percent or more. With increasing concurrency, UNIX File System and UNIX File System with logging performance improves little or none for reads and generally degrades for writes.
- While UNIX File System and VERITAS File System bandwidth improves as RAID-0 striping units increase from 1 to 24, VERITAS File System scales better. VERITAS File System read performance scales linearly and write performance scales almost linearly.
- UNIX File System with logging performance consistently lagged behind that of UNIX File System, even though our tests used large files that presumably do not stress a file system's metadata.

5.2 Test Configurations

The vxbench tests were performed on a Sun E6500 computer system, with 12 400-MHz UltraSPARC-II CPUs and 12 MB of RAM. The machine has eight Sbus boards, with a total of 12 Fibre Channel Sbus controllers. Unisys Clariion arrays, each with 18 GB Seagate 10,000 RPM Fibre Channel disks, were attached to the Sbus controllers using Sun Social HBAs. The arrays were configured into a striped (RAID 0) volume using VERITAS Volume Manager™ 3.1.1. However, due to the high bandwidth nature of the benchmark, it was important to configure the machine so that no bandwidth bottleneck existed in the disk arrays, controllers, or boards. Such bottlenecks might prevent the file system software from being stressed.

To determine the maximum I/O bandwidth of the hardware configuration, we performed large sequential reads from the raw disks. An example of a vxbench command to perform such a read is:

```
vxbench -w read -i iosize=64k,iocount=3000 -P raw-disk-device(s)
```

This test showed that a single disk of a single Clariion array supported about 25 MB/sec. When additional raw disks were added (appended additional `/dev/rdisk` arguments to the vxbench command), scaling stopped after about 75 MB/sec, or three disks per array. The limitation was either the fibre connection out of the array (rated at 120 MB/sec), or the Sbus controller (rated at 200 MB/sec). (Obtaining bandwidth below the rated rate is expected, due to protocol overheads.) Regardless of the cause, we determined to use only three disks per Clariion array in this study.

When the `vxbench` command was expanded to read from raw disks residing on different arrays, the bandwidth obtained depended on whether the arrays shared an Sbus board. As long as the arrays were on different Sbus boards, the rate of 75 MB/sec per Clariion array scaled almost linearly. The E6500 had eight Sbus boards and we connected only one Clariion array to each such board, yielding a total of 24 disks (given the three disk per brick limit). In this way, we were able to configure RAID 0 volumes of between 1 and 24 columns, confident that in the presence of sequential reads and writes, scalability limits are due to software, not hardware.

5.3 Experiments

The experiments consisted of `vxbench` runs to write and read 1 GB of data. Several `vxbench` options altered the way in which the data was transferred:

- The transfer is either a read or a write. Note that before a transfer, we unmount and then remount the file system to ensure that its cache is cleared. Additionally, after a file system is mounted (at `/mnt`), and just before running `vxbench`, the mounted file system has its directory metadata primed with `ls -l /mnt`, so `vxbench` will not incur this cost.
- `Vxbench` is configured to read or write with I/O units of either 8K or 64K.
- `Vxbench` can read (write) the 1 GB of data using a variable number of files, though the grand total of all files was always kept at 1 GB. Note that *one process is assigned to each file* (using the `vxbench -P` flag). Our tests include runs that use 1, 2, 4, 8, 16 and 32 processes (and files).

In addition to `vxbench` variants, other variables in this experiment were:

- Choice of file system: either UNIX File System, UNIX File System with logging, or VERITAS File System. No options for UNIX File System mkfs were specified beyond the default. For VERITAS File System, non-default mkfs options were: 1 K block size, 16-MB log and largefile support. The only mount-time option for UNIX File System was the `logging` flag (UNIX File System with logging). VERITAS File System was mounted with the delayed logging option to match the semantics of UNIX File System with logging.
- Varying stripe units. VERITAS Volume Manager 3.1.1 was used to configure a RAID-0 volume using from between one and 24 disks (columns). Our tests include runs that use 1, 2, 3, 4, 5, 6, 8, 10, 12, 16, 20 and 24 columns. A volume is created using the command:

```
vxassist -g testdg make voll 15g layout=stripe ncol=numcols disks
```

5.4 Results

Because the set of variables over which the benchmark was run is a five-dimensional space (file system type, number of RAID-0 columns, reads vs. writes, 8K vs. 64K unit size and number of processes and files), no single graph can show all results. The full results are listed in Appendix D; this sub-section illustrates the most important points.

5.4.1 Scaling With Increasing Concurrency

This sub-section discusses how well UNIX File System, UNIX File System with logging and VERITAS File System read performance scale with increasing processes (and files). We focus on the case of a 24-way RAID-0 stripe, because that configuration provides the maximum opportunity for high-bandwidth transfers. In one sense, multiple processes and files may be expected to decrease the I/O rate, because having multiple files implies that the (grand total of) 1 GB is no longer read or written sequentially; some disk seeking is necessary. On the other hand, multiple processes allow latency to be hidden, tending to increase the I/O rate.

Figure 15 and Figure 16 show the performance of 8K reads and writes, respectively, on this 24-way stripe. For 8K reads, all file systems usually perform better under higher concurrencies. VERITAS File System improves its read transfer rate quickly as the number of processes increases, obtaining its peak bandwidth of about 220 MB/sec with only eight processes. (This rate is about half of what vxbench is able to obtain when run against the raw disk. The difference is likely due to the overhead of the operating system page cache and consequent copies.)

UNIX File System improves its read performance only incrementally up to 16 processes and begins to degrade beyond that. VERITAS File System is consistently 180 percent to 320 percent faster than UNIX File System with four or more processes. Only with one process, where VERITAS File System is 33 percent faster, is the gap narrower.

UNIX File System with logging has similar read performance compared to UNIX File System for the lower concurrencies and inferior performance for 16 processes (18 percent slower) and 32 processes (20 percent slower). Compared to UNIX File System with logging, VERITAS File System obtains read performance improvement of 27 percent with one process and between about 250 percent and 350 percent for the higher concurrencies.

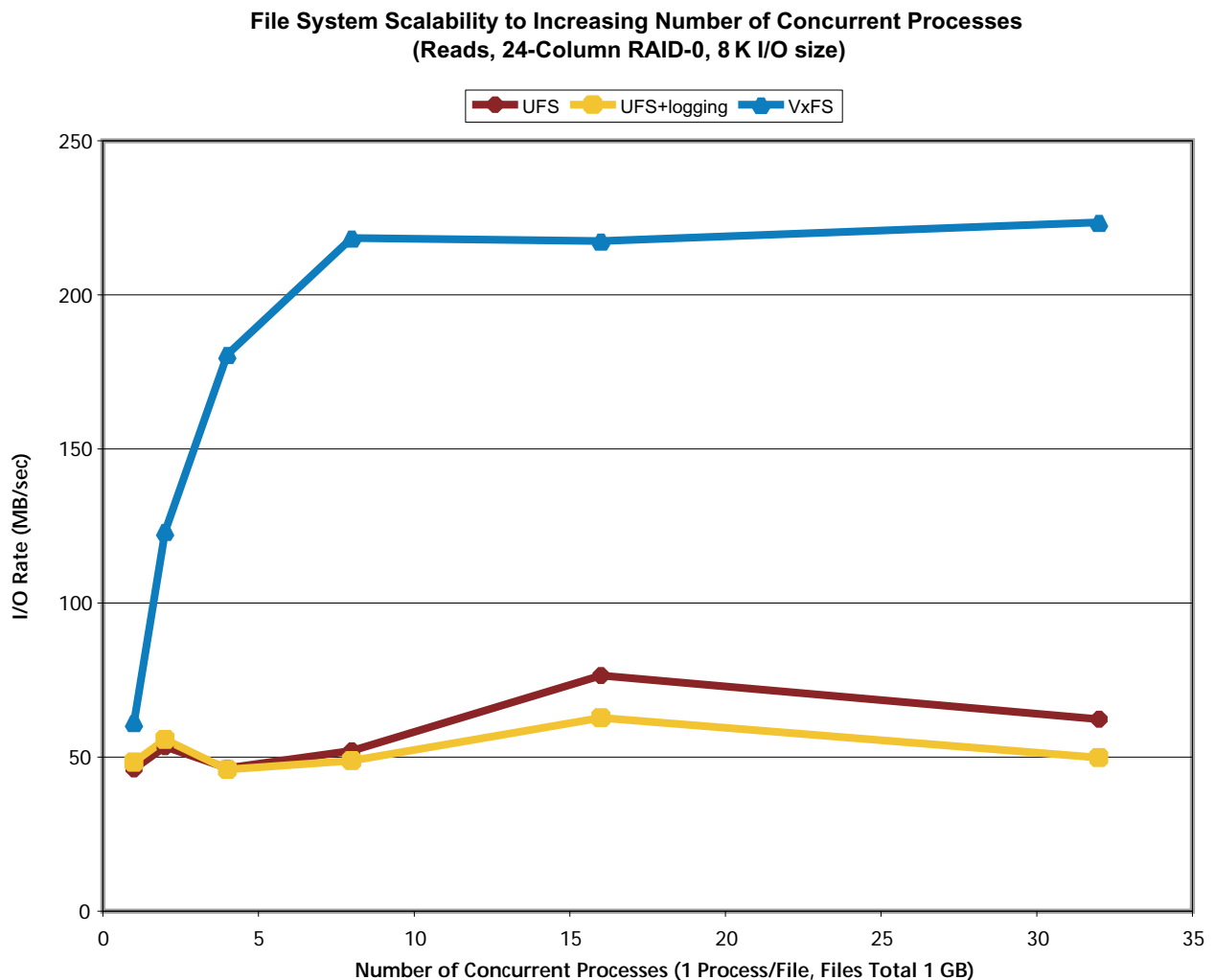


Figure 15: File system scalability to multiple processes and files, (8K reads). UNIX File System and UNIX File System with logging read performance improves only marginally with multiple processes and begin to dip again beyond 16 processes. VERITAS File System quickly obtains its peak read performance of over 200 MB/sec with eight processes, and does not degrade beyond that.

The performance of 8K writes, shown in Figure 16, tells a similar story. Compared to UNIX File System, VERITAS File System obtains performance increases from 57 percent (1 process) to 208 percent (32 processes). The performance increases are an indication of how VERITAS File System write performance scales with concurrency, while UNIX File System generally does not (except from one to four processes). The performance increases compared to UNIX File System with logging are even more dramatic, where VERITAS File System bests UNIX File System with logging from between 75 percent (1 process) to 310 percent (16 processes). UNIX File System with logging runs between 10 percent and 45 percent slower than UNIX File System without logging.

**File System Scalability To Increasing Number of Concurrent Processes
(Writes, 24-Column RAID-0, 8 K I/O size)**

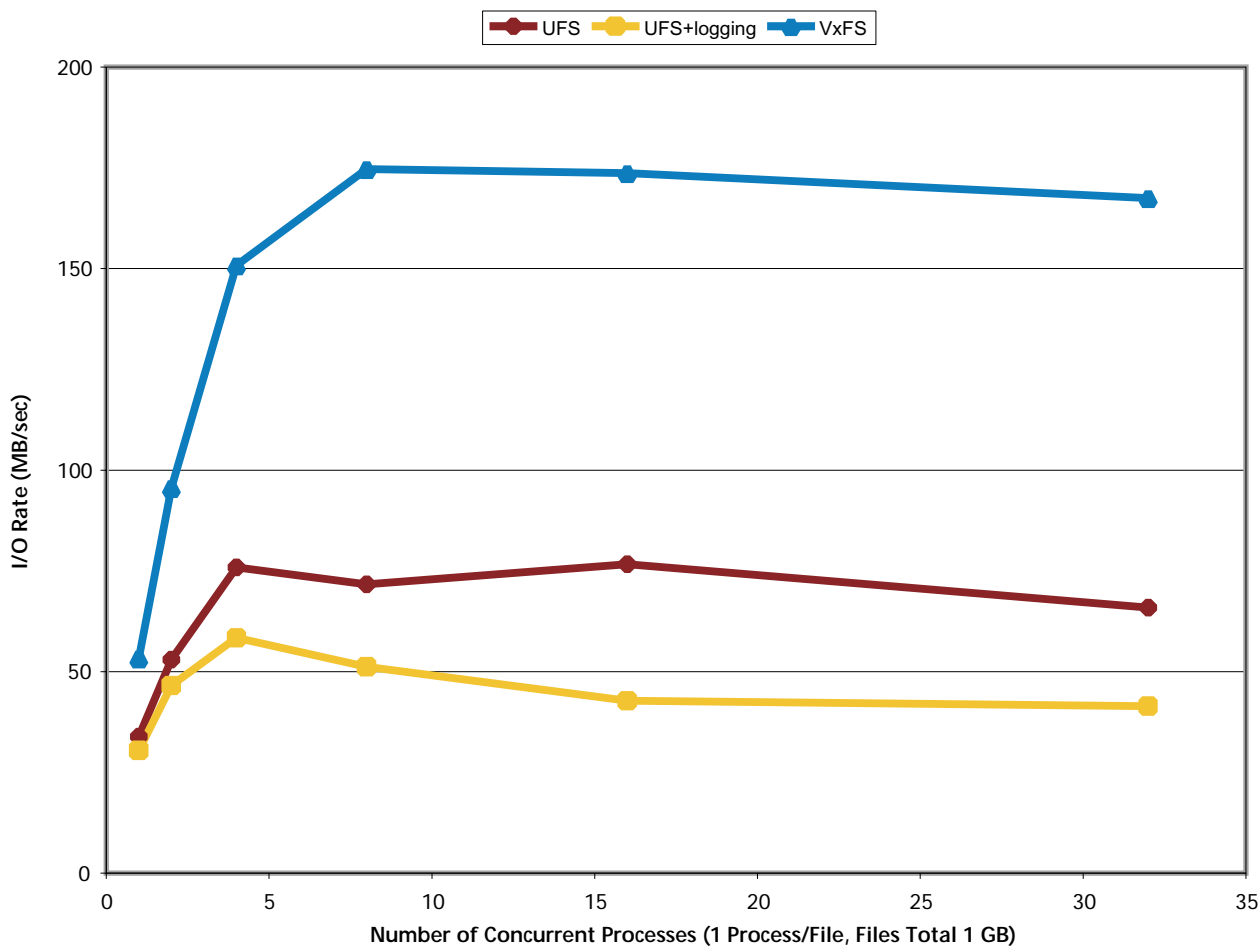


Figure 16: File system scalability to multiple processes and files, (8K writes). UNIX File System shows flat or slightly declining performance with eight or more processes. UNIX File System with logging performs even worse, with performance declining steadily, to about 42 MB/sec, as concurrency increases past four processes. VERITAS File System, by contrast, quickly achieves a peak write throughput that is about 80 percent of its read throughput, and does not degrade significantly as concurrency increases.

Figure 17 and Figure 18 compare UNIX File System, UNIX File System with logging and VERITAS File System reads and writes (respectively) on a 24-way stripe with a larger I/O size, 64K. Figure 17 shows that for reads, all file systems usually perform better with higher concurrencies. VERITAS File System improves its read transfer rate quickly as the number of processes increases, obtaining its peak bandwidth of about 220 MB/sec with only eight processes. UNIX File System improves its read performance only incrementally with multiple processes; VERITAS File System is consistently 200 percent to 300 percent faster than UNIX File System with four or more processes. Only with one process, where VERITAS File System is 47 percent faster than UNIX File System, is the gap narrower. UNIX File System with logging has similar read performance compared to UNIX File System for the lower concurrencies and inferior performance for 16 processes (18 percent slower) and 32 processes (25 percent slower). Compared to UNIX File System with logging, VERITAS File System obtains read performance increase of 56 percent with one process, and between about 270 percent and 350 percent for the higher concurrencies.

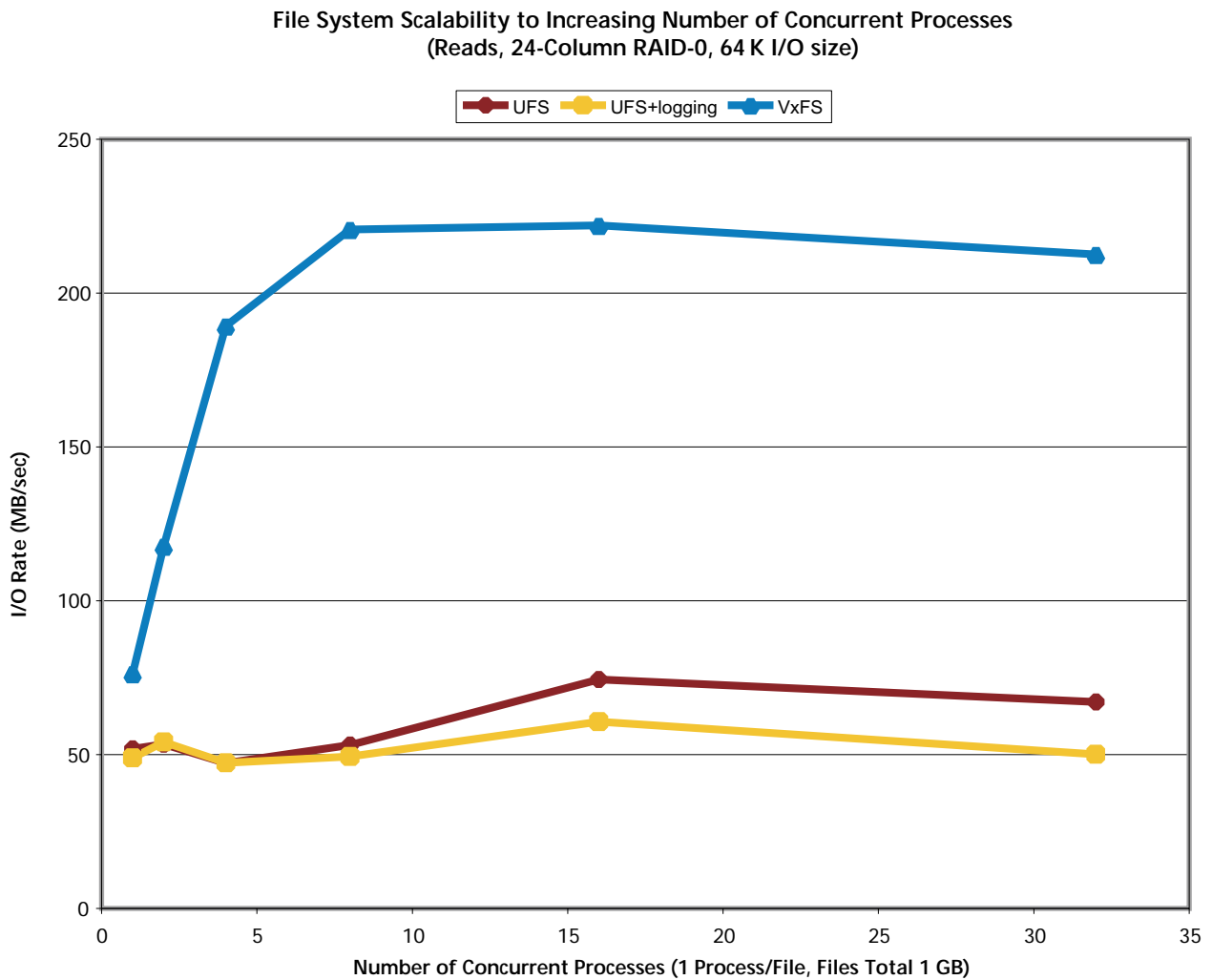


Figure 17: File system scalability to multiple processes and files (64K reads). The performance of UNIX File System and UNIX File System with logging improves only marginally with multiple processes and begins to dip again after 16 processes. VERITAS File System quickly obtains its peak read performance of over 200 MB/sec.

Write performance, shown in Figure 18, tells a similar story. Compared to UNIX File System, VERITAS File System obtains performance increases from 67 percent (1 process) to 208 percent (32 processes). The performance increases are an indication of how VERITAS File System write performance scales with concurrency, while UNIX File System generally does not (except from one to four processes, where UNIX File System obtained its only improvement). The performance increases compared to UNIX File System with logging are even more dramatic, where VERITAS File System bests UNIX File System with logging from between 77 percent (1 process) to 377 percent (32 processes).

**File System Scalability To Increasing Number of Concurrent Processes
(Writes, 24-Column RAID-0, 64 K I/O size)**

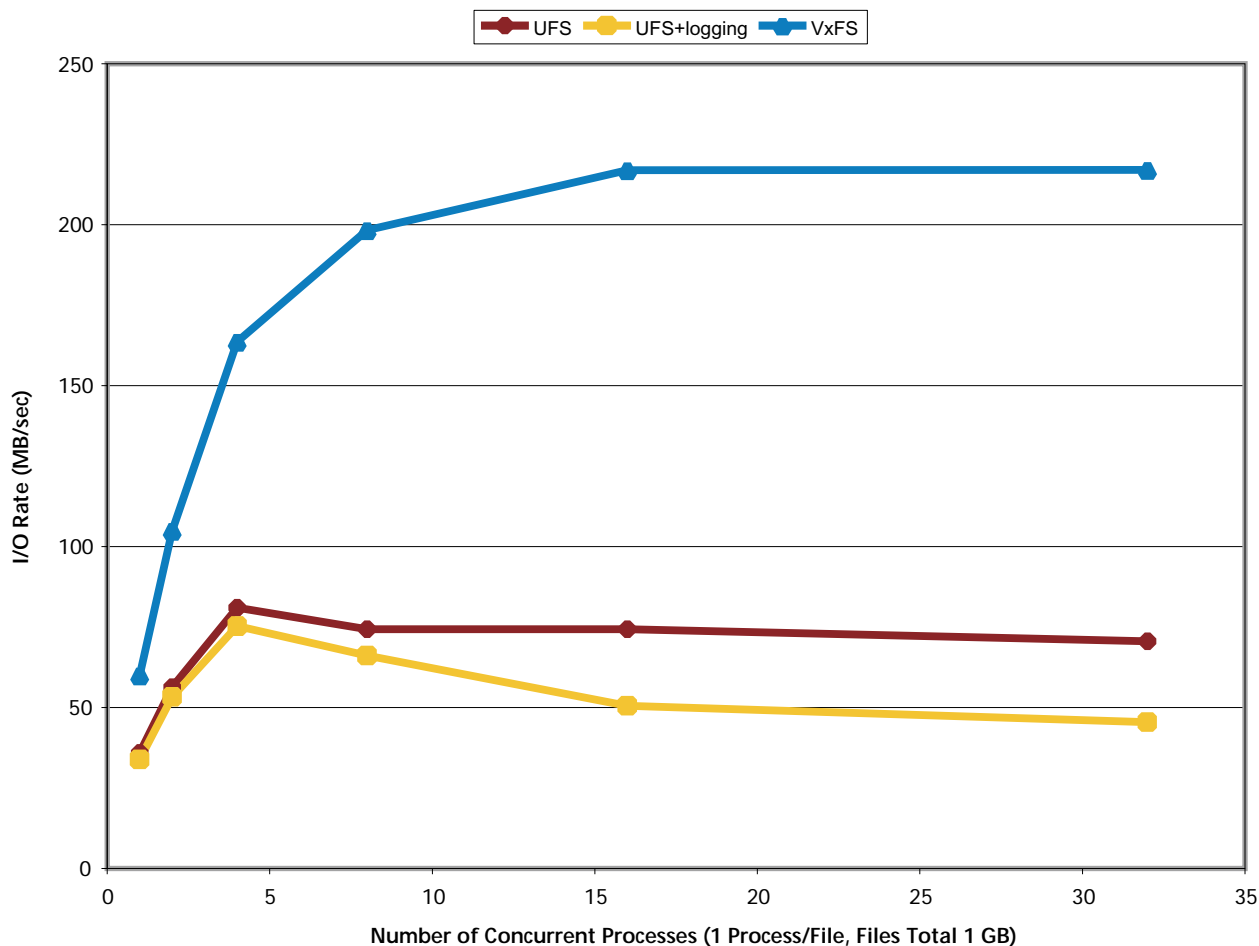


Figure 18: File system scalability to multiple processes and files (64K writes). UNIX File System shows flat or slightly declining performance with eight or more processes. UNIX File System with logging performs even worse, with performance declining steadily, to about 45 MB/sec, as concurrency increases past four processes. VERITAS File System, by contrast, quickly achieves a peak write throughput that nearly equals its read throughput, and does not degrade as concurrency increases.

5.4.2 Scaling With Increasing Number of Columns in a RAID-0 Stripe

Figure 19 through Figure 22 give an indication of how the file systems scale with increasing numbers of columns in a RAID-0 stripe. To keep the graphs simple, the number of concurrent processes is fixed at eight.

Figure 19 and Figure 20 shows the scaling for 8K and 64K reads, respectively. The figures illustrate that VERITAS File System is much better able to benefit from the additional bandwidth of larger stripe sizes. For reads, VERITAS File System scales linearly to higher stripe sizes. UNIX File System and UNIX File System with logging also scale, though at a much slower pace, meaning that VERITAS File System increases its performance over these file systems with increasing number of columns. Compared to UNIX File System, the VERITAS File System 64 K read performance increases at 1, 8, 16 and 24 columns are 59 percent, 98 percent, 209 percent and 320 percent, respectively. Compared to UNIX File System with logging, the VERITAS File System 64K read performance increases for 1, 8, 16 and 24 columns are 127 percent, 260 percent, 288 percent and 347 percent, respectively. UNIX File System is from 7 percent to 82 percent faster than UNIX File System with logging for 64 K reads.

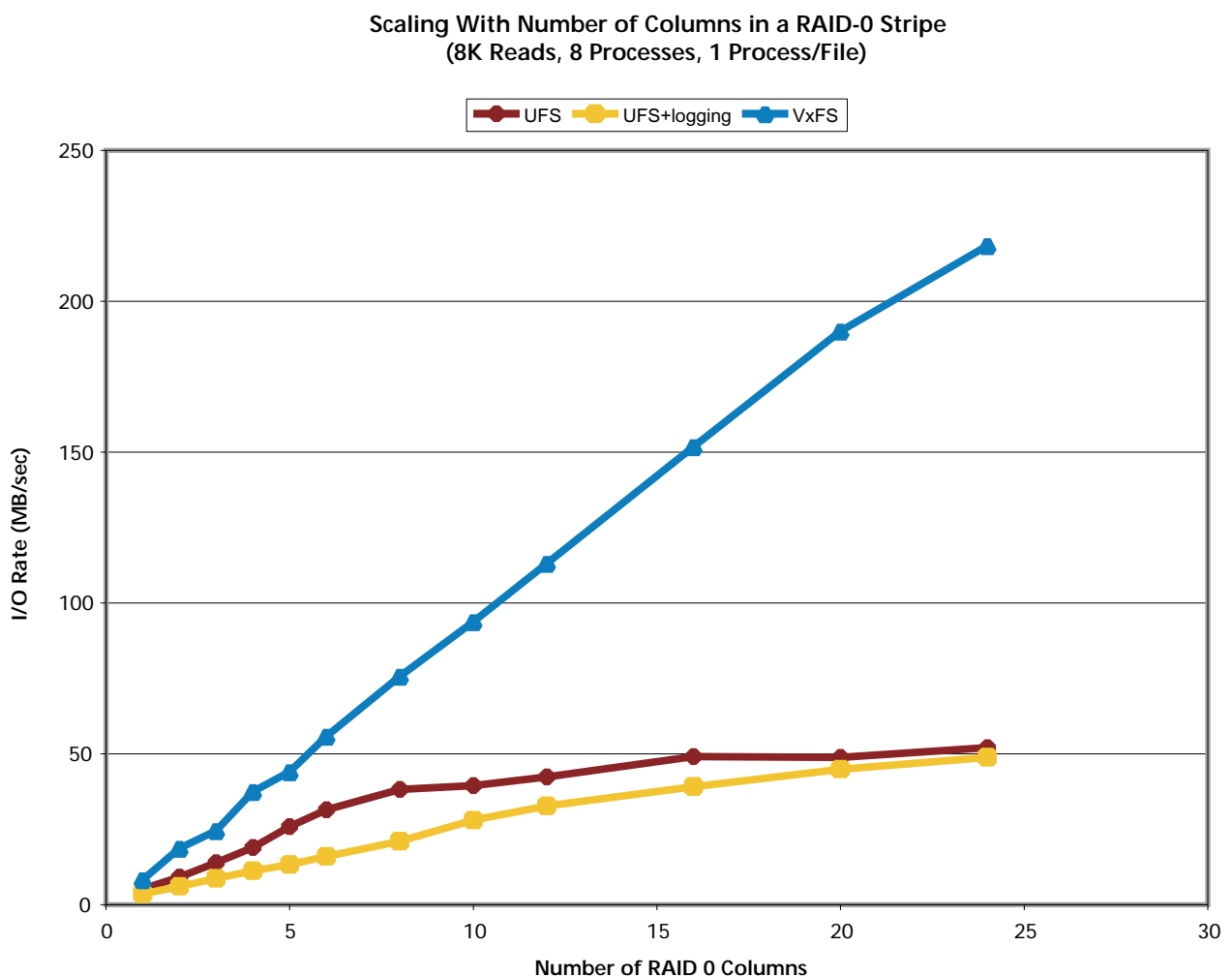


Figure 19: Scalability to increasing number of RAID-0 columns (8K reads). VERITAS File System scales its read performance linearly. UNIX File System and UNIX File System with logging read performance scales at a much slower rate compared to VERITAS File System. In addition, they scale little beyond 16 columns. Even with the highest numbers of columns, UNIX File System and UNIX File System with logging are unable to read much more than 50 MB/sec, a data rate that was surpassed by VERITAS File System with only six columns.

Scaling With Number of Columns in a RAID-0 Stripe
(64K Reads, 8 Processes, 1 Process/File)

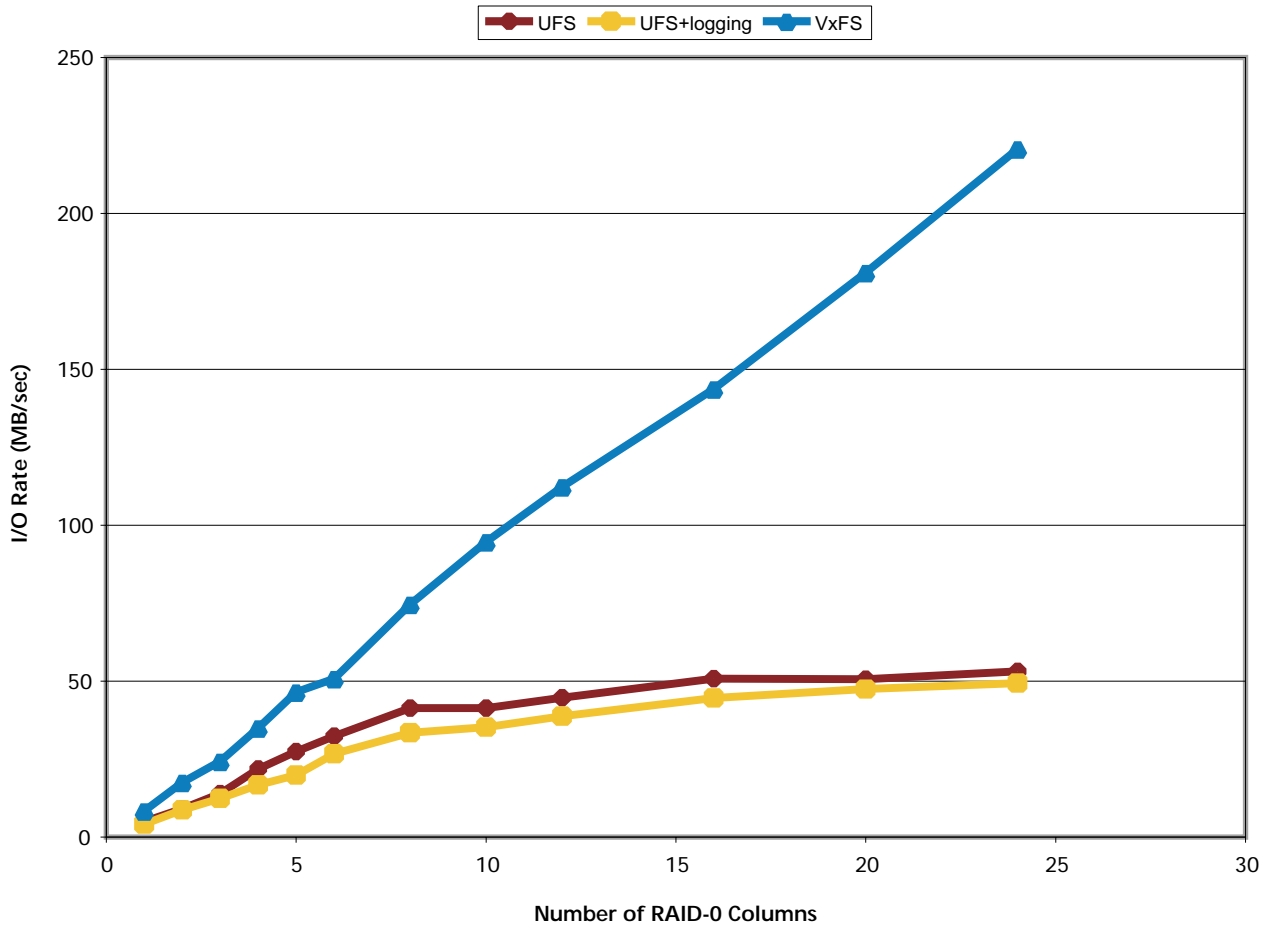


Figure 20: Scalability to increasing number of RAID-0 columns (64K reads). VERITAS File System scales its read performance linearly. UNIX File System and UNIX File System with logging read performance scales at a much slower rate than VERITAS File System. In addition, they scale little beyond eight columns. Even with the highest numbers of columns, UNIX File System and UNIX File System with logging are unable to read much more than 50 MB/sec, a data rate that was achieved by VERITAS File System with only six columns. At the higher stripe sizes, VERITAS File System surpasses UNIX File System and UNIX File System with logging by more than 300 percent.

The VERITAS File System performance increases are slightly less for writes than for reads because UNIX File System and UNIX File System with logging do a reasonable job of scaling for writes (though the rate of increase is always less than that of VERITAS File System). Compared to UNIX File System, VERITAS File System 64K write performance increases at 1, 8, 16 and 24 columns are 150 percent, 102 percent, 162 percent and 167 percent, respectively. Compared to UNIX File System with logging, VERITAS File System 64 K write performance increases for 1, 8, 16 and 24 columns are 193 percent, 171 percent, 241 percent and 200 percent, respectively. Although UNIX File System and UNIX File System with logging scale better for writing than reading, there still is a large performance win for VERITAS File System. As usual, UNIX File System performed better than UNIX File System with logging for writes by 12 percent to 34 percent.

**Scaling With Number of Columns in a RAID-0 Stripe
(8K Writes, 8 Processes, 1 Process/File)**

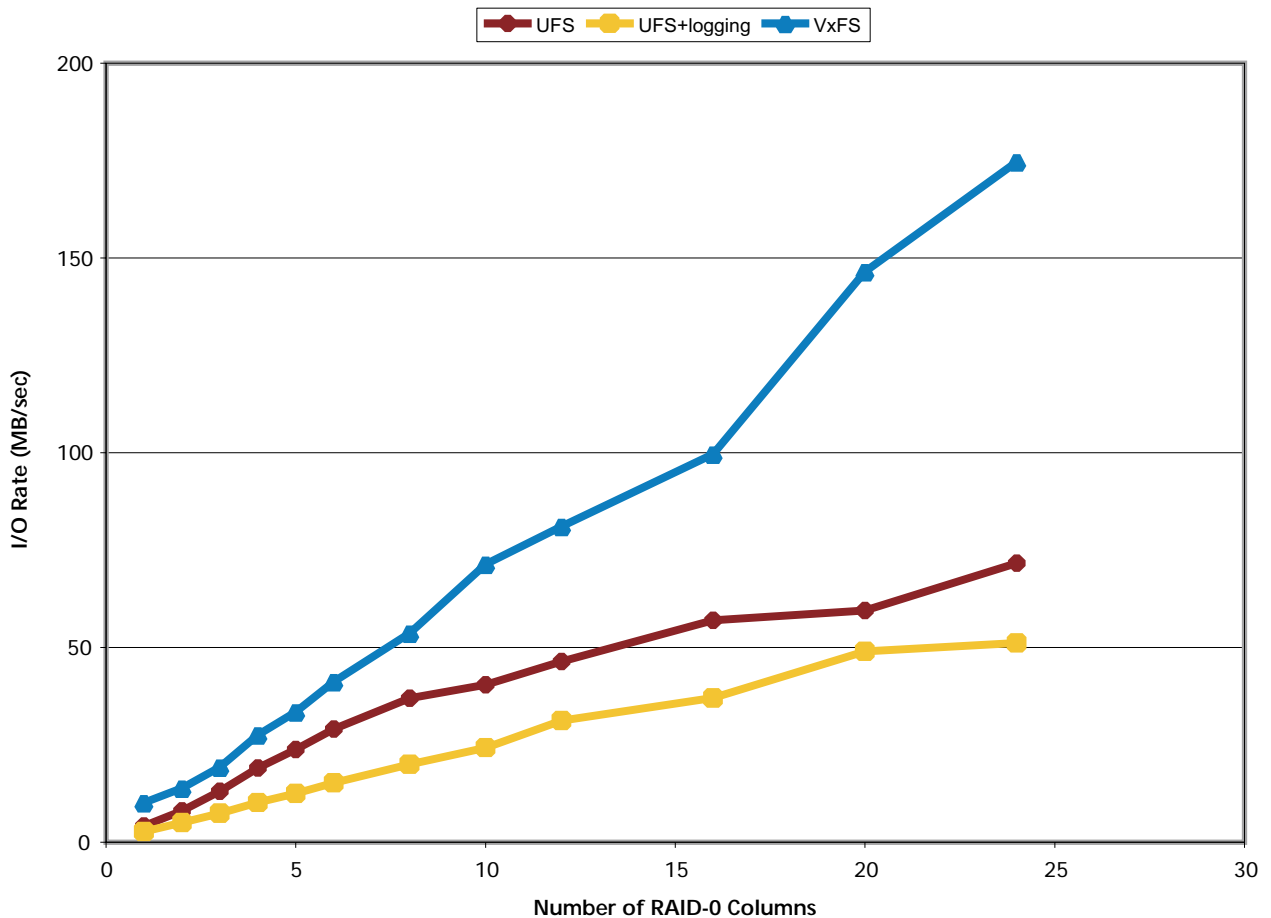


Figure 21: Scalability to increasing number of RAID-0 columns (8 K writes). VERITAS File System scales its write performance about linearly. UNIX File System and UNIX File System with logging also scale, but to a much lesser degree.

**Scaling With Number of Columns in a RAID-0 Stripe
(64K Writes, 8 Processes, 1 Process/File)**

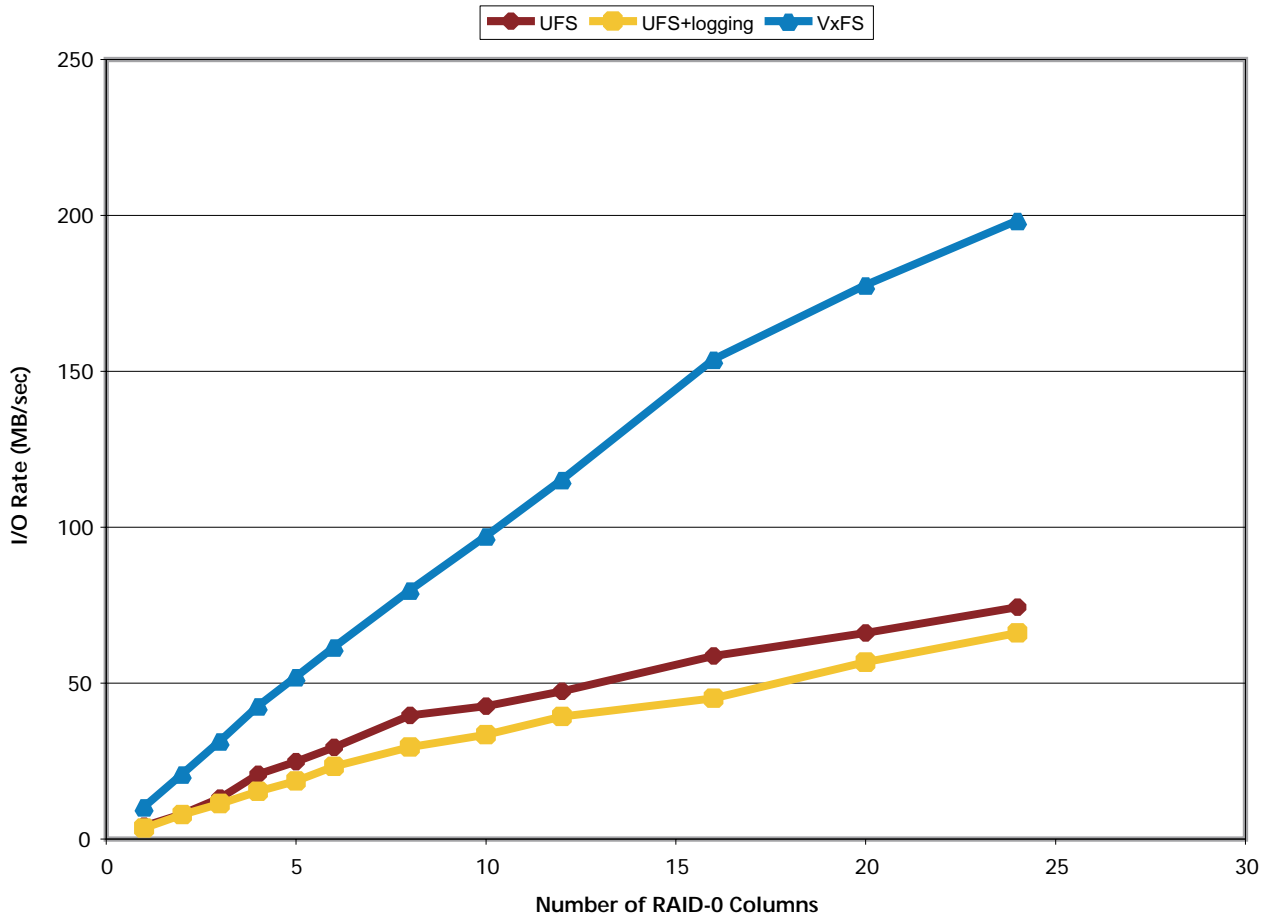


Figure 22: Scalability to increasing number of RAID-0 columns (64K writes). VERITAS File System scales its write performance almost linearly for all stripe sizes. UNIX File System and UNIX File System with logging also scale, but to a much lesser degree. Note that even when using all 24 columns, UNIX File System and UNIX File System with logging are unable to match the write performance that VERITAS File System achieves with only eight columns.

The VERITAS File System performance increases are slightly less for writes than for reads because UNIX File System and UNIX File System with logging do a reasonable job of scaling for writes. Compared to UNIX File System, VERITAS File System write performance increases at 1, 8, 16 and 24 columns are 144 percent, 45 percent, 75 percent and 144 percent, respectively. Compared to UNIX File System with logging, VERITAS File System write performance increases for 1, 8, 16 and 24 columns are 275 percent, 168 percent, 169 percent and 241 percent, respectively. Although UNIX File System and UNIX File System with logging scale better for writing than for reading, there still is a large performance win for VERITAS File System. As usual, UNIX File System performed better than UNIX File System with logging for writes, by 40 percent to 85 percent.

Appendix A: Configuration Information for SPECsfs97 Benchmarks

1. E6500 (8X8)

E6500 JBOD SPECsfs97.v3 Result

UNIX File System

SPECsfs97.v3 = 8060 Ops/Sec (Overall Response Time = 6.5)

UNIX File System + Logging

SPECsfs97.v3 = 5610 Ops/Sec (Overall Response Time = 6.6)

VERITAS File System

SPECsfs97.v3 = 10942 Ops/Sec (Overall Response Time = 5.4)

UNIX File System		UNIX File System + Logging		VERITAS File System	
Throughput ops/sec	Response Msec	Throughput ops/sec	Response Msec	Throughput ops/sec	Response Msec2.5
483	3.4	482	3.3	484	2.2
982	3.8	982	3.0	982	2.7
1979	4.3	1981	4.7	1979	3.1
3011	5.1	3001	5.5	3011	3.7
4002	5.8	4008	6.8	4000	4.0
4999	6.8	5004	7.9	4997	4.6
5988	7.5	5610	29.2	5987	5.0
6961	8.9			6955	5.4
7927	13.9			7934	6.1
8060	21.9			8500	6.8
				9032	7.8
				9521	8.6
				10046	9.7
				10595	11.6
				10919	13.2
				10942	15.5

CPU, Memory and Power

Model NameSun Microsystems Enterprise 6500
 Processor400-MHz UltraSPARC
 # of Processors8
 Primary Cache16 KBI + 16 KBD on chip
 Secondary Cache4096 KB (I+D) off chip
 Other CacheN/A
 UPSN/A
 Other HardwareN/A

Memory Size8192 MB
 NVRAM SizeN/A
 NVRAM TypeN/A
 NVRAM DescriptionN/A

Server Software

OS Name and Version Solaris 8 Update 4
Other Software VERITAS File System 3.4 Patch1, VERITAS Volume Manager 3.1.1
File System VERITAS File System, UNIX File System
NFS version 3

Server Tuning

Buffer Cache Size default
NFS Processes 1,600
Fileset Size 126 GB (VERITAS File System), 82 GB (UNIX File System), 58 GB (UNIX File System + Logging)

Network Subsystem

Network Type 1000Mbit Ethernet
Network Controller Desc. 1000Mbit Sun Gigabit Ethernet
Number Networks 1 (N1)
Number Network Controllers 1
Protocol Type TCP

Switch Type Cisco Catalyst 3500XL
Bridge Type N/A
Hub Type N/A
Other Network Hardware N/A

Disk Subsystem and Filesystem

Number Disk Controllers 11
Number of Disks 198
Number of Filesystems 11 (F1-F11)
File System Creation Ops bsize = 1024, logsize = 16384, delayed logging (VERITAS File System), default (UFS, UFS + Logging)
File System Config default (See Notes)

Disk Controller Sun internal UltraSCSI-3 controller
of Controller Type 1
Number of Disks 4
Disk Type Fujitsu 18-GB Enterprise 10 K Series MAG3182L, 10,000 RPM
File Systems on Disks OS, swap, misc.

Disk Controller SUN Socal SCSI-3 (501-3060)
of Controller Type 11
Number of Disks 198
Disk Type Seagate 9-GB Cheetah ST39103FC, 10,000 RPM (158)
Disk Type Seagate 18-GB Cheetah ST318203FC, 10,000 RPM (40)
File Systems on Disks F1-F11
Special Config Notes Unisys ESM700 (Clariion) each enclosure with 10 drives

Load Generator (LG) Configuration

Number of Load Generators14
Number of Processes per LG11
Biod Max Read Setting5
Biod Max Write Setting5

LG TypeLG1
LG ModelSun Microsystems Netra T1
Number and Type Processors1 400-MHz UltraSPARC
Memory Size256 MB
Operating SystemSolaris 2.8
CompilerSUNWspro4
Compiler Optionsdefault
Network TypeOnboard 100baseT

Testbed Configuration

LG#	LG Type	Network	Target File System	Notes
1-14	LG1	N1	F1, F2, F3... F11	

Notes and Tuning

Vxtunefs parameters are dynamically set at the time of making the file system; no effort was made to change the default chosen by VERITAS File System.

```
read_pref_io = 65536  
read_nstream = 9  
read_unit_io = 65536  
write_pref_io = 65536  
write_nstream = 9  
write_unit_io = 65536  
pref_strength = 20  
buf_breakup_size = 262144  
discovered_direct_iosz = 262144  
max_direct_iosz = 9437184  
default_indir_size = 8192  
qio_cache_enable = 0  
write_throttle = 492160  
max_diskq = 9437184  
initial_extent_size = 8  
max_seqio_extent_size = 2048  
max_buf_data_size = 8192  
hsm_write_prealloc = 0
```

2. E6500 (12X12)

E6500 JBOD SPECsfs97.v3 Result

UNIX File System

SPECsfs97.v3 = 10353 Ops/Sec (Overall Response Time = 6.8)

UNIX File System + Logging

SPECsfs97.v2 = 5791 Ops/Sec (Overall Response Time = 6.3)

VERITAS File System

SPECsfs97.v3 = 13851 Ops/Sec (Overall Response Time = 5.0)

UNIX File System		UNIX File System + Logging		VERITAS File System	
Throughput ops/sec	Response Msec	Throughput ops/sec	Response Msec	Throughput ops/sec	Response Msec2.5
483	3.5	482	4.2	483	2.7
982	3.4	982	3.4	981	2.2
1977	4.2	1980	3.4	1978	2.5
3007	4.0	3002	4.8	3000	2.9
3989	5.0	4003	5.7	3991	3.5
5002	6.1	4994	7.2	4994	3.8
5987	7.1	5791	26.1	5984	3.8
6954	7.7			6948	4.4
7947	8.7			7914	4.7
8508	9.5			8484	4.9
9032	11.0			8994	5.5
9553	11.9			9499	5.6
10037	13.5			10047	5.9
10353	17.1			11000	6.5
				12033	7.7
				12997	9.3
				13851	12.3

CPU, Memory and Power

Model NameSun Microsystems Enterprise 6500

Processor400-MHz UltraSPARC

of Processors12

Primary Cache16 KBI + 16 KBD on chip

Secondary Cache4096 KB (I+D) off chip

Other CacheN/A

UPSN/A

Other HardwareN/A

Memory Size12288 MB

NVRAM SizeN/A

NVRAM TypeN/A

NVRAM DescriptionN/A

Server Software

OS Name and Version Solaris 8 Update 4
Other Software VERITAS File System 3.4 Patch1, VERITAS Volume Manager 3.1.1
File System VERITAS File System, UNIX File System
NFS version 3

Server Tuning

Buffer Cache Size default
NFS Processes 1600
Fileset Size 135 GB (VERITAS File System), 106 GB (UFS) 58 GB (UFS + Logging)

Network Subsystem

Network Type 1000Mbit Ethernet
Network Controller Desc. 1000Mbit Sun Gigabit Ethernet
Number Networks 1 (N1)
Number Network Controllers 1
Protocol Type TCP

Switch Type Cisco Catalyst 3500XL
Bridge Type N/A
Hub Type N/A
Other Network Hardware N/A

Disk Subsystem and Filesystem

Number Disk Controllers 11
Number of Disks 198
Number of Filesystems 11 (F1-F11)
File System Creation Ops bsize = 1024, logsize = 16384 delayed logging (VERITAS File System), default (UFS, UFS+Logging)
File System Config default (See Notes)

Disk Controller Sun internal UltraSCSI-3 controller
of Controller Type 1
Number of Disks 4
Disk Type Fujitsu 18-GB Enterprise 10K Series MAG3182L, 10,000 RPM
File Systems on Disks OS, swap, misc.

Disk Controller SUN Socal SCSI-3 (501-3060)
of Controller Type 11
Number of Disks 198
Disk Type Seagate 9-GB Cheetah ST39103FC, 10,000 RPM(158)
Disk Type Seagate 18-GB Cheetah ST318203FC, 10,000 RPM(40)
File Systems on Disks F1-F11
Special Config Notes Unisys ESM700 (Clariion) each enclosure with 10 drives

Load Generator (LG) Configuration

Number of Load Generators14
Number of Processes per LG11
Biod Max Read Setting5
Biod Max Write Setting5

LG TypeLG1
LG ModelSun Microsystems Netra T1
Number and Type Processors1 400-MHz UltraSPARC
Memory Size256 MB
Operating SystemSolaris 2.8
CompilerSUNWspro4
Compiler Optionsdefault
Network TypeOnboard 100baseT

Testbed Configuration

LG#	LG Type	Network	Target File System	Notes
1-14	LG1	N1	F1, F2, F3... F11	

Notes and Tuning

Vxtunefs parameters are dynamically set at the time of making the file system; no effort was made to change the default chosen by VERITAS File System.

```
read_pref_io = 65536  
read_nstream = 9  
read_unit_io = 65536  
write_pref_io = 65536  
write_nstream = 9  
write_unit_io = 65536  
pref_strength = 20  
buf_breakup_size = 262144  
discovered_direct_iosz = 262144  
max_direct_iosz = 9437184  
default_indir_size = 8192  
qio_cache_enable = 0  
write_throttle = 492160  
max_diskq = 9437184  
initial_extent_size = 8  
max_seqio_extent_size = 2048  
max_buf_data_size = 8192  
hsm_write_prealloc = 0
```

Appendix B: Additional Software Configuration Information for TPC-C Benchmark

The following shared memory and IPC settings were enabled in the `/etc/system` file:

* Shared memory

*

```
set shmsys:shminfo_shmmin=1
set shmsys:shminfo_shmmax=6442450000
set shmsys:shminfo_shmmni=256
set shmsys:shminfo_shmseg=100
```

*

* Semaphores

```
set semsys:seminfo_semmap=256
set semsys:seminfo_semmni=4096
set semsys:seminfo_semmns=4096
set semsys:seminfo_semmnu=4096
set semsys:seminfo_semume=64
set semsys:seminfo_semmsl=75
set semsys:seminfo_semopm=50
```

Below is the Oracle parameter file used for the benchmark tests:

```
control_files                = (/TPC-C_disks/control_001)

#dbwr_io_slaves              = 20
#dbwr_io_slaves              = 10
#disk_asynch_io              =FALSE
disk_asynch_io               =TRUE

_filesystemio_options        =setall

parallel_max_servers         = 30
recovery_parallelism         = 20
# checkpoint_process         = TRUE           # obsolete
compatible                   = 8.1.5.0.0
db_name                       = TPC-C
db_files                      = 200
db_file_multiblock_read_count = 32
#db_block_buffers            = 512000
#db_block_buffers            = 1024000
db_block_buffers              = 900000
# _db_block_write_batch      = 1024           # obsolete
# db_block_checkpoint_batch   = 512           # obsolete
dml_locks                     = 500
hash_join_enabled             = FALSE
log_archive_start             = FALSE
# log_archive_start           = TRUE
#log_archive_buffer_size     = 32             # obsolete
#log_checkpoint_timeout      = 600
log_checkpoint_interval       = 100000000
log_checkpoints_to_alert      = TRUE
log_buffer                    = 1048576
#log_archive_dest            = /archlog
# gc_rollback_segments       = 220           # obsolete
# gc_db_locks                 = 100          # obsolete
gc_releasable_locks          = 0
max_rollback_segments        = 220
open_cursors                  = 200
#processes                   = 200
processes                     = 150
sessions                      = 600
transactions                  = 400
distributed_transactions      = 0
transactions_per_rollback_segment = 1
#rollback_segments           =
(t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t14,t15,t16,t17,t18,t19,t20,t21,t22,t23,t24,t
25,t26,t27,t28,t29)
rollback_segments             =
(t_0_1,t_0_2,t_0_3,t_0_4,t_0_5,t_0_6,t_0_7,t_0_8,t_0_9,t_0_10,t_0_11,t_0_12,t_0_13,t_0_14
,t_0_15,t_0_16,t_0_17,t_0_18,t_0_19,t_0_20,t_0_21,t_0_21,t_0_22,t_0_23,t_0_24,t_0_25,t_0_
26,t_0_27,t_0_28,t_0_29,t_0_30,t_0_31,t_0_32,t_0_33,t_0_34,t_0_35,t_0_36,t_0_37,t_0_38,t_
0_39,t_0_40,t_0_41,t_0_42,t_0_43,t_0_44,t_0_45,t_0_46,t_0_47,t_0_48,t_0_49,t_0_50,t_0_51,
t_0_52,t_0_53,t_0_54,t_0_55,t_0_56,t_0_57,t_0_58,t_0_59,t_0_60)

shared_pool_size              = 75000000
# discrete_transactions_enabled = FALSE       # obsolete
cursor_space_for_time         = TRUE
```

Appendix C: Source Code for touch_files Program

```
/* $Id: touch_files.c,v 1.7 2001/05/14 15:05:26 oswa Exp $ */
/*
 * Touch/Create file benchmark.
 *
 * Compile with: gcc -O2 -o touch_files touch_files.c
 * or for debug: gcc -O2 -Wall -g -o touch_files touch_files.c
 */

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <errno.h>
#include <fcntl.h>
#include <unistd.h>
#include <time.h>

static char ver[] = "$Id: touch_files.c,v 1.7 2001/05/14 15:05:26 oswa Exp $";

int
main(int argc, char **argv)
{
    unsigned int    ndirs = 0;
    unsigned int    nfiles = 0;
    unsigned int    bsize = 0;
    unsigned int    i, j;
    char            path[1024];
    void            *filler;
    int             multithreaded = 0, pid, parent = 1;
    int             fh;
    time_t          stime, ftime;

    if (argc < 3) {
        printf("usage: touch_files [-m] num-dirs num-files");
        printf(" [size-in-bytes]\n\n");
        return (1);
    }

    while ((i = getopt(argc, argv, "m")) != -1) {
        switch (i) {
            case 'm':
                multithreaded = 1;
                break;
        }
    }

    ndirs = atoi(argv[optind++]);
    nfiles = atoi(argv[optind++]);
    if (multithreaded && argc > 4 || !multithreaded && argc > 3 ) {
        bsize = atoi(argv[optind]);
        filler = malloc(bsize);
        if (filler == NULL) {
            printf("Unable to allocate memory\n");
            exit(1);
        }
    }

    printf("Creating %d top level directories ", ndirs);
    printf("with %d files in each\n", nfiles);
    printf("Using %s\n",
           (multithreaded ? "multiple processes" : "single thread"));
}
```

```

/* start timer */
stime = time(NULL);
printf("Starting at: %s", ctime(&stime));
for (i = 0; i < ndirs; i++) {
    /* top dirs */
    sprintf(path, "./%d", i);
    if (((mkdir(path, 0777)) == -1) && errno != EEXIST) {
        perror("mkdir() failed");
        exit(1);
    }
    if (multithreaded) {
        pid = fork();
        if (pid == -1) {
            fprintf(stderr, "Cannot fork\n");
            exit(1);
        } else if (pid != 0) {
            continue;
        } else {
            parent = 0;
        }
    }
    for (j = 0; j < nfiles; j++) {
        /* files per dir */
        sprintf(path, "./%d/%d", i, j);
        fh = open(path,
            O_WRONLY|O_CREAT|O_TRUNC|O_EXCL|O_DSYNC|O_SYNC,
            0666);
        if (fh == -1) {
            perror("open() failed");
            exit(1);
        }
        if (write(fh, filler, bsize) == -1) {
            perror("write() failed");
            exit(1);
        }
        close(fh);
    }

    if (!parent) {
        return 0;
    }
}

if (multithreaded && parent) {
    for (i = 0 ; i < ndirs ; i++) {
        wait(NULL);
    }
}

/* stop timer + print results */

ftime = time(NULL);
printf("Finished at: %s", ctime(&ftime));
printf("The run took %d seconds\n", (ftime - stime));

free(filler);
return (0);
}

```

Appendix D: Complete vxbench Results

No single graph could summarize the entire set of vxbench runs from Section 5 because the set of variables that were modified from run to run form a five-dimensional space: file system, RAID-0 striping unit, reads vs. writes, 8 K vs. 64 K I/O size and concurrency. This appendix lists all such results for reference.

All write and read operations total 1GB. The variations are the I/O size (8 K vs. 64 K) and how many concurrent files are accessed (there is always process one per file). The total size read or written is always 1 GB, no matter how many files are used. Results are I/O rates, in MB/sec.

ncol=1

op	iosize	concurrent files	UFS (MB/sec)	UFS+log (MB/sec)	VERITAS FS (MB/sec)
rd	8K	1	23.58	23.61	25.35
rd	8K	2	9.98	13.90	19.49
rd	8K	4	7.79	3.77	13.86
rd	8K	8	5.18	3.62	8.23
rd	8K	16	3.04	2.85	13.45
rd	8K	32	2.62	2.59	10.75
rd	64K	1	23.61	23.53	26.13
rd	64K	2	11.92	12.33	19.15
rd	64K	4	8.81	7.00	13.54
rd	64K	8	4.91	4.17	8.34
rd	64K	16	2.99	2.50	12.87
rd	64K	32	2.71	2.29	10.69
wr	8K	1	19.53	14.58	21.46
wr	8K	2	14.23	13.91	10.75
wr	8K	4	9.42	4.87	10.11
wr	8K	8	4.14	2.70	10.12
wr	8K	16	2.53	2.39	10.03
wr	8K	32	2.23	2.15	9.82
wr	64K	1	19.61	17.85	21.20
wr	64K	2	15.04	13.91	10.10
wr	64K	4	9.70	7.74	10.29
wr	64K	8	4.12	3.52	10.31
wr	64K	16	2.59	2.16	10.31
wr	64K	32	2.44	2.07	10.07

ncol=2

op	iosize	concurrent files	UFS (MB/sec)	UFS+log (MB/sec)	VERITAS FS (MB/sec)
rd	8K	1	43.01	43.32	50.84
rd	8K	2	14.60	17.59	37.17
rd	8K	4	11.98	7.46	20.13
rd	8K	8	9.21	5.99	18.70
rd	8K	16	5.75	5.17	20.11
rd	8K	32	4.93	4.61	22.64
rd	64K	1	44.56	44.38	52.28
rd	64K	2	18.97	18.05	38.17
rd	64K	4	13.45	11.95	21.58
rd	64K	8	9.05	8.69	17.56
rd	64K	16	5.56	5.04	17.96
rd	64K	32	4.89	4.55	22.23
wr	8K	1	24.78	21.43	20.40
wr	8K	2	23.01	21.22	12.70
wr	8K	4	15.91	9.13	12.59
wr	8K	8	8.07	5.07	13.91
wr	8K	16	4.95	4.30	13.27
wr	8K	32	4.31	4.04	12.68
wr	64K	1	26.49	24.84	35.47
wr	64K	2	23.25	22.54	19.03
wr	64K	4	15.98	13.67	18.73
wr	64K	8	8.01	7.86	20.90
wr	64K	16	4.83	4.38	20.97
wr	64K	32	4.39	4.07	20.52

ncol=3

op	iosize	concurrent files	UFS (MB/sec)	UFS+log (MB/sec)	VERITAS FS(MB/sec)
rd	8K	1	45.58	45.96	61.19
rd	8K	2	17.48	21.90	49.04
rd	8K	4	18.81	11.23	35.19
rd	8K	8	13.97	8.68	24.58
rd	8K	16	8.53	7.66	28.40
rd	8K	32	7.29	7.00	34.35
rd	64K	1	45.58	47.32	73.02
rd	64K	2	21.46	21.07	48.66
rd	64K	4	19.75	18.64	34.84
rd	64K	8	13.97	12.37	24.27
rd	64K	16	7.81	7.48	27.30
rd	64K	32	7.42	6.63	34.18
wr	8K	1	26.99	23.40	49.17
wr	8K	2	27.79	26.30	19.71
wr	8K	4	23.26	14.37	19.55
wr	8K	8	13.12	7.44	19.26
wr	8K	16	7.46	6.39	19.73
wr	8K	32	6.35	6.01	19.25
wr	64K	1	28.88	26.97	58.11
wr	64K	2	29.24	27.83	29.48
wr	64K	4	23.46	21.33	30.34
wr	64K	8	13.16	11.37	31.49
wr	64K	16	6.83	6.41	32.07
wr	64K	32	6.59	5.86	31.92

ncol=4

op	iosize	concurrent files	UFS (MB/sec)	UFS+log (MB/sec)	VERITAS FS (MB/sec)
rd	8K	1	48.67	50.07	62.04
rd	8K	2	20.72	22.93	60.36
rd	8K	4	20.52	14.73	44.97
rd	8K	8	18.98	11.22	37.53
rd	8K	16	11.86	10.06	36.97
rd	8K	32	9.85	9.37	47.35
rd	64K	1	51.20	48.90	77.95
rd	64K	2	23.59	24.46	60.71
rd	64K	4	22.34	19.93	46.92
rd	64K	8	21.85	16.69	34.95
rd	64K	16	11.38	10.65	36.48
rd	64K	32	10.08	8.89	46.24
wr	8K	1	29.14	24.84	51.70
wr	8K	2	32.23	29.79	26.01
wr	8K	4	29.05	19.91	26.49
wr	8K	8	19.04	10.25	27.55
wr	8K	16	10.53	8.44	27.09
wr	8K	32	8.74	7.89	27.36
wr	64K	1	31.09	30.33	59.45
wr	64K	2	32.96	33.01	41.00
wr	64K	4	30.90	27.58	40.29
wr	64K	8	20.73	15.29	42.68
wr	64K	16	10.14	9.28	42.55
wr	64K	32	9.07	7.87	42.07

ncol=5

op	iosize	concurrent files	UFS (MB/sec)	UFS+log (MB/sec)	VERITAS FS (MB/sec)
rd	8K	1	51.11	52.20	61.79
rd	8K	2	23.78	28.55	70.98
rd	8K	4	22.45	17.66	63.55
rd	8K	8	25.91	13.31	44.05
rd	8K	16	17.03	12.27	47.31
rd	8K	32	11.83	11.34	58.14
rd	64K	1	50.23	53.67	76.75
rd	64K	2	27.80	27.14	71.40
rd	64K	4	23.90	22.89	56.54
rd	64K	8	27.43	19.86	46.53
rd	64K	16	16.08	12.79	47.13
rd	64K	32	12.08	11.07	55.50
wr	8K	1	30.34	26.36	51.97
wr	8K	2	36.14	33.05	32.08
wr	8K	4	35.26	25.07	34.11
wr	8K	8	23.81	12.52	33.47
wr	8K	16	16.19	10.61	34.64
wr	8K	32	10.52	9.79	33.34
wr	64K	1	32.42	30.27	59.69
wr	64K	2	38.02	36.14	54.12
wr	64K	4	35.62	34.07	51.56
wr	64K	8	24.83	18.66	52.04
wr	64K	16	14.95	11.04	50.26
wr	64K	32	10.99	9.77	50.72

ncol=6

op	iosize	concurrent files	UFS (MB/sec)	UFS+log (MB/sec)	VERITAS FS (MB/sec)
rd	8K	1	51.42	52.43	62.61
rd	8K	2	27.03	32.45	81.65
rd	8K	4	25.12	18.52	69.14
rd	8K	8	31.49	15.98	55.82
rd	8K	16	21.26	14.64	56.35
rd	8K	32	13.91	13.53	68.97
rd	64K	1	51.46	51.75	77.01
rd	64K	2	30.05	28.78	81.87
rd	64K	4	26.46	24.23	68.11
rd	64K	8	32.34	26.69	50.80
rd	64K	16	19.36	15.85	55.24
rd	64K	32	14.82	13.03	69.10
wr	8K	1	30.84	26.85	52.42
wr	8K	2	38.39	36.18	39.10
wr	8K	4	40.97	28.48	40.87
wr	8K	8	29.03	15.23	41.22
wr	8K	16	20.41	12.50	40.55
wr	8K	32	12.39	11.58	40.86
wr	64K	1	32.91	31.44	58.35
wr	64K	2	40.42	38.91	63.63
wr	64K	4	40.63	38.40	61.94
wr	64K	8	29.39	23.33	61.61
wr	64K	16	18.42	13.83	59.28
wr	64K	32	13.68	11.56	60.11

ncol=8

op	iosize	concurrent files	UFS (MB/sec)	UFS+log (MB/sec)	VERITAS FS (MB/sec)
rd	8K	1	51.93	53.03	60.92
rd	8K	2	32.17	34.48	98.49
rd	8K	4	27.69	22.94	100.28
rd	8K	8	38.19	21.02	75.69
rd	8K	16	35.55	19.03	73.59
rd	8K	32	25.19	17.78	92.98
rd	64K	1	54.07	54.75	76.13
rd	64K	2	33.97	32.47	100.18
rd	64K	4	29.97	28.58	96.44
rd	64K	8	41.40	33.48	74.60
rd	64K	16	38.09	22.16	74.04
rd	64K	32	23.16	17.14	88.53
wr	8K	1	31.58	28.64	52.17
wr	8K	2	43.19	38.34	54.61
wr	8K	4	49.98	36.20	51.14
wr	8K	8	37.00	20.03	53.67
wr	8K	16	37.90	17.18	53.98
wr	8K	32	23.73	15.70	53.79
wr	64K	1	33.96	33.78	58.33
wr	64K	2	45.47	42.60	92.12
wr	64K	4	50.97	46.89	80.84
wr	64K	8	39.62	29.49	79.85
wr	64K	16	40.07	19.63	77.28
wr	64K	32	21.71	15.01	75.89

ncol=10

op	iosize	concurrent files	UFS (MB/sec)	UFS+log (MB/sec)	VERITAS FS (MB/sec)
rd	8K	1	51.24	51.02	63.19
rd	8K	2	39.04	40.47	105.49
rd	8K	4	29.84	26.80	117.31
rd	8K	8	39.50	28.03	93.81
rd	8K	16	35.51	23.11	94.15
rd	8K	32	22.94	22.82	113.60
rd	64K	1	54.12	50.97	76.93
rd	64K	2	39.78	40.06	107.50
rd	64K	4	32.56	31.88	116.63
rd	64K	8	41.41	35.27	94.70
rd	64K	16	32.59	26.42	94.48
rd	64K	32	26.06	21.68	105.10
wr	8K	1	33.12	29.57	52.46
wr	8K	2	46.68	40.29	90.15
wr	8K	4	55.20	41.25	70.03
wr	8K	8	40.45	24.22	71.34
wr	8K	16	35.66	21.94	67.73
wr	8K	32	20.89	20.84	69.39
wr	64K	1	35.80	33.87	60.46
wr	64K	2	48.83	46.45	99.66
wr	64K	4	58.56	53.40	96.48
wr	64K	8	42.59	33.46	97.19
wr	64K	16	31.84	23.44	93.19
wr	64K	32	25.34	19.27	94.13

ncol=12

op	iosize	concurrent files	UFS (MB/sec)	UFS+log (MB/sec)	VERITAS FS (MB/sec)
rd	8K	1	49.83	51.23	62.50
rd	8K	2	40.85	43.44	112.11
rd	8K	4	33.01	30.22	134.69
rd	8K	8	42.35	32.73	113.11
rd	8K	16	38.81	28.27	112.23
rd	8K	32	28.23	26.38	123.57
rd	64K	1	53.20	53.04	77.95
rd	64K	2	44.27	42.34	110.61
rd	64K	4	33.43	33.84	138.75
rd	64K	8	44.65	38.73	112.33
rd	64K	16	38.83	30.61	112.03
rd	64K	32	29.01	26.77	127.98
wr	8K	1	33.19	29.88	51.60
wr	8K	2	47.56	42.07	94.39
wr	8K	4	60.77	45.82	81.17
wr	8K	8	46.43	31.19	81.10
wr	8K	16	38.35	26.18	81.57
wr	8K	32	25.29	23.29	82.70
wr	64K	1	35.17	34.06	59.36
wr	64K	2	50.33	48.09	100.28
wr	64K	4	62.72	58.33	118.18
wr	64K	8	47.41	39.28	115.29
wr	64K	16	37.65	27.79	112.11
wr	64K	32	26.74	23.21	110.54

ncol=16

op	iosize	concurrent files	UFS (MB/sec)	UFS+log (MB/sec)	VERITAS FS (MB/sec)
rd	8K	1	49.68	51.67	62.27
rd	8K	2	47.73	48.90	121.63
rd	8K	4	37.56	35.95	172.32
rd	8K	8	49.06	39.12	151.83
rd	8K	16	71.72	34.00	148.11
rd	8K	32	60.78	33.67	166.04
rd	64K	1	52.99	50.41	75.33
rd	64K	2	48.35	47.15	123.90
rd	64K	4	38.15	38.69	172.13
rd	64K	8	50.85	44.61	143.69
rd	64K	16	65.73	42.84	148.58
rd	64K	32	59.43	39.06	167.37
wr	8K	1	33.53	30.42	52.79
wr	8K	2	51.08	44.75	91.45
wr	8K	4	66.26	51.88	102.10
wr	8K	8	56.95	37.08	99.63
wr	8K	16	65.82	30.94	101.91
wr	8K	32	62.73	29.41	108.79
wr	64K	1	34.81	35.12	61.37
wr	64K	2	53.07	51.39	99.97
wr	64K	4	70.26	64.44	151.86
wr	64K	8	58.68	45.16	153.86
wr	64K	16	63.70	39.87	146.07
wr	64K	32	60.41	33.94	144.93

ncol=20

op	iosize	concurrent files	UFS (MB/sec)	UFS+log (MB/sec)	VERITAS FS (MB/sec)
rd	8K	1	49.25	51.00	62.99
rd	8K	2	50.25	53.02	114.54
rd	8K	4	43.70	41.41	179.27
rd	8K	8	48.87	44.88	190.08
rd	8K	16	62.57	47.29	184.88
rd	8K	32	47.72	43.04	197.90
rd	64K	1	51.94	52.29	77.84
rd	64K	2	52.68	50.69	118.22
rd	64K	4	46.79	42.28	195.37
rd	64K	8	50.66	47.47	181.06
rd	64K	16	61.89	50.43	182.07
rd	64K	32	44.49	41.91	197.45
wr	8K	1	34.18	30.37	51.61
wr	8K	2	52.37	45.91	93.16
wr	8K	4	72.84	54.75	145.66
wr	8K	8	59.51	49.00	146.50
wr	8K	16	56.05	38.63	148.35
wr	8K	32	45.38	35.20	143.19
wr	64K	1	36.20	34.51	59.72
wr	64K	2	56.74	52.82	98.83
wr	64K	4	76.03	70.06	161.24
wr	64K	8	66.11	56.71	177.70
wr	64K	16	59.56	42.01	183.83
wr	64K	32	42.66	38.79	182.48

ncol=24

op	iosize	concurrent files	UFS (MB/sec)	UFS+log (MB/sec)	VERITAS FS (MB/sec)
rd	8K	1	46.19	48.40	61.21
rd	8K	2	53.35	55.67	123.23
rd	8K	4	46.46	46.03	180.63
rd	8K	8	52.06	48.81	218.42
rd	8K	16	76.44	62.74	217.46
rd	8K	32	62.33	49.86	223.51
rd	64K	1	51.88	48.89	76.25
rd	64K	2	53.42	54.07	117.76
rd	64K	4	47.24	47.33	189.29
rd	64K	8	53.15	49.37	220.71
rd	64K	16	74.39	60.70	222.03
rd	64K	32	67.08	50.10	212.57
wr	8K	1	33.84	30.47	53.25
wr	8K	2	52.96	46.53	95.52
wr	8K	4	75.84	58.40	150.78
wr	8K	8	71.64	51.16	174.68
wr	8K	16	76.62	42.76	173.72
wr	8K	32	65.89	41.46	167.47
wr	64K	1	35.88	33.90	59.98
wr	64K	2	56.33	53.28	104.89
wr	64K	4	81.02	75.30	163.65
wr	64K	8	74.33	66.14	198.34
wr	64K	16	74.36	50.53	216.93
wr	64K	32	70.58	45.49	217.05



V
E
R
I
T
A
S

W
H
I
T
E

P
A
P
E
R

VERITAS Software Corporation
Corporate Headquarters
350 Ellis Street
Mountain View, CA 94043
650-527-8000 or 800-327-2232

For additional information about VERITAS Software, its products, or the location of an office near you, please call our corporate headquarters or visit our Web site at www.veritas.com

sales@veritas.com

Copyright © 2001 VERITAS Software Corporation. All Rights Reserved. VERITAS, VERITAS SOFTWARE, the VERITAS logo, *Business Without Interruption*, VERITAS The Data Availability Company, VERITAS File System, VERITAS Quick I/O, Storage Checkpoint and VERITAS Volume Manager are trademarks or registered trademarks of VERITAS Software Corporation in the U.S. and/or other countries. Other product names mentioned herein may be trademarks or registered trademarks of their respective companies. Specifications and product offerings subject to change without notice. August 2001.

90-01554-399