

File System Performance: The Solaris™ OS, UFS, Linux ext3, and ReiserFS

A Technical White Paper
August 2004



Table of Contents

Overview	1
Introduction	1
Summary Data	2
File Systems Overview: UFS, ext3, and ReiserFS	3
Introduction	3
UFS	3
ext3	5
ReiserFS	6
Systems and Benchmark Tests	8
The PostMark Benchmark	8
Results	11
ext3	11
ReiserFS	12
UFS	12
Comparative Performance	13
Conclusion	14
Further Work	15
Acknowledgments	15
References	16

Chapter 1

Overview

Introduction

During the economic boom of the late 1990s, companies expanded their information technology (IT) infrastructures to meet the demands of increasing numbers of network users and devices. Times have since changed, and businesses are under increasing pressure to deliver IT services with greater functionality at reduced costs. Pressure to deliver results to the bottom line is forcing IT managers to find new ways to drive costs out of their operations, and they are increasingly turning to commodity, x86-based hardware for its renowned price-performance.

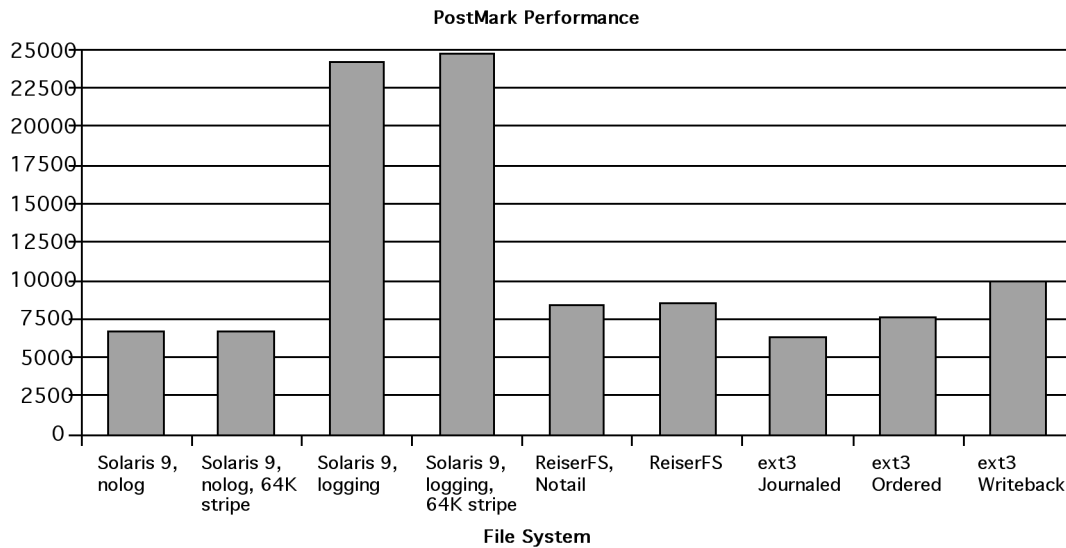
With an unwavering commitment to its long-term vision — *The Network is the Computer*™ — Sun Microsystems continues to lead the industry toward simpler, faster, open, and more cost-effective ways of using network computing for business benefit. As such, Sun offers a comprehensive range of servers and workstations based on AMD Opteron and Intel Xeon processors, running either the Solaris™ Operating System (OS) or Linux. Continually aiming to meet changing customer demands, Sun has integrated new technologies and features into the Solaris OS, designed to deliver significant performance enhancements across systems based on UltraSPARC®, AMD Opteron, and Intel Xeon technologies.

This paper focuses on a specific benchmark example to highlight how these improvements have enhanced the performance of Solaris systems. It presents initial comparisons of the Solaris OS and Linux performance using the PostMark file system benchmark. PostMark has become an industry-standard benchmark, designed to emulate Internet applications such as e-mail, netnews, and e-commerce. As such, it provides relevant insights into how the Solaris OS and Linux can be expected to perform in Web, e-mail, and related environments, where x86-based hardware is typically deployed.

This paper compares the performance of three file systems: UFS running on the Solaris OS, ext3 on Linux, and ReiserFS on Linux. The results demonstrate that for applications such as e-mail, news, e-commerce, and others, where the profile of the application is one of small I/Os, Solaris UFS provides superior performance over Linux.

Summary Data

The following graph shows an aggregate of the transaction capability per second for the file systems tested. These tests were carried out by accumulating measurements for the number of transactions per second across a range of loads (25,000 to 500,000 transactions). This graph presents a simplified view of the results. Detailed results are provided later in the paper.



The following broad conclusions can be drawn:

- Solaris UFS performs better on the PostMark benchmark than any of the other file systems tested.
- The use of logging in the Solaris UFS is the key factor in delivering high performance for this benchmark.
- The trade-off between data integrity and performance in the various mount options to ext3 is confirmed. This has implications where data accessibility is critical.
- The slight performance overhead of tail packing in ReiserFS is demonstrated. This has implications for users for whom both disk space and performance are critical factors.

Chapter 2

File Systems Overview: UFS, ext3, and ReiserFS

Introduction

This chapter provides a brief technical introduction to the file systems used during these tests, namely Solaris UFS, Linux ext3, and Linux ReiserFS, highlighting those features that impact performance the most. This outline is for the benefit of those readers not familiar with these file systems. Readers familiar with this area may safely skip this material. Further details can be found in the materials and URLs listed in the *References* section of this paper.

Both the Solaris OS and Linux provide a choice of file systems. For example, Solaris applications requiring high-performance bulk I/O or a storage area network (SAN-aware) file system may utilize Sun StorEdge™ QFS¹. Hierarchical storage management is made available in the Solaris OS via Sun StorEdge SAM-FS software.

In addition to ext3 and ReiserFS, Linux users can choose JFS, which was made available to the community by IBM², and XFS, which was ported as the result of work done at SGI³. A good general primer on the file systems available for Linux is provided in *Linux Filesystems*⁴. This paper does not provide additional details on alternative file systems, but concentrates on the three tested: UFS, ext3, and ReiserFS.

UFS

UFS is the primary file system for the Solaris OS. UFS is extremely mature, very stable, and for most applications, is the file system of choice. It is also bundled for free with the base operating system.

Solaris UFS has its roots in the Berkeley Fast File System (FFS) of the 1980s, although today's file system is the result of more than 20 years of enhancement, evolution, and stabilization. FFS was designed to overcome the problems inherent in the original UNIX® file system; principally poor performance due to a small fixed block size and a separation of metadata and data. These factors led to a mode of operation that required a great number of lengthy disk seeks. As a UNIX file system aged on disk, this problem got worse. There was only one superblock in the original UNIX file system, which meant that its corruption led to irretrievable damage, necessitating a full file system re-creation and data restoration. However, FFS was simple and lightweight, and the source code for it is still instructive today⁵.

1. Reference 9 (*Performance Benchmark Report: UNIX File System and VERITAS File System 3.5 on Solaris 9 Operating System 12/02 Release*) and 10 (*SAS Intelligence Storage Integration with Sun Data Management*)
2. Reference 1 (*JFS Overview: How the Journalled File System Cuts System Restart Times to the Quick*)
3. Reference 15 (*Porting the SGI XFS File System to Linux*)
4. Reference 6 (*Linux Filesystems*)
5. Reference 12 (*Lions' Commentary on UNIX 6th Edition*)

Work at Berkeley aimed to improve both reliability and throughput of the file system with the development of variable block sizes, a file system check program (fsck(1M)), and multiple superblocks. Details can be found in *The Design and Implementation of the 4.3 Bsd Unix Operating System: Answer Book*⁶ and *The Design and Implementation of the 4.4BSD Operating System*⁷.

The fundamental core of the Solaris OS is the original SunOS™ product. SunOS 4.x software was based on the 4.3 BSD UNIX distribution; SunOS 5.x software, which is the heart of the Solaris OS today, was first released in 1992, the result of a merge of BSD UNIX and AT&T UNIX System V by AT&T and Sun. UFS is a fundamental BSD technology that was introduced into System V UNIX as a result of this work.

Since then, work has continued on the development of UFS. At the same time, the development of the virtual file cache has replaced the traditional UNIX buffer cache. The principal difference between the two is the ability of the latter to act as a cache for files rather than disk blocks. This results in a performance gain, as the retrieval of cached data does not require the entire file system code path to be traversed. These developments are covered in detail in *Solaris Internals: Core Kernel Architecture*⁸.

Logging in UFS

A major enhancement to UFS was implemented in 1994: File system metadata logging to provide better reliability and faster reboot times following a system crash or outage⁹. Logging enables the file system to be mounted and used immediately without lengthy file system checking. Originally, logging was provided as part of a volume management add-on product, Online: DiskSuite™ (later Solstice DiskSuite™) software. In 1998, logging functionality and volume management were incorporated into the base Solaris Operating System. Solstice DiskSuite logging used a separate disk partition to store the log. Now UFS logging embeds the log in the file system.

Since its inclusion in the Solaris OS, UFS logging has undergone continuous improvement. Performance often exceeds nonlogging performance. For example, under the metadata-intensive PostMark benchmark used for these tests, logging provides a 300-percent improvement on nonlogging transaction rates¹⁰. Users with any sizable file systems use logging regardless of performance considerations. Thus, it makes sense to enable it as the default option. It is enabled by default for file systems of less than one terabyte commencing with the 9/04 release of the Solaris 9 OS.

Because logging provides advantages for the PostMark benchmark, it will be covered in more detail.

UFS Logging Details

UFS logging does not log the contents of user files; it logs *metadata* — the structure of the file system. Metadata is data that *describes* user file data, such as inodes, cylinder groups, block bitmaps, directories, and so on. After a crash, the file system remains intact, but the contents of individual files may be impacted. In common with most journaling file systems, UFS does not have the facility to log file data because of the negative effect this would have on performance. (Note that the ext3 file system discussed later in this paper does have this capability as a mount option, and benchmarking results bear out its negative effect on performance.)

The UFS log is a persistent, circular, append-only data store, occupying disk blocks within the file system. It is not visible to the user. The space used by the log shows up in *df(1M)* output but not in *du(1)* output. Typically the log consumes one megabyte per gigabyte of file system space, up to a maximum of 64 megabytes.

6. Reference 11 (*The Design and Implementation of the 4.3 Bsd Unix Operating System: Answer Book*)

7. Reference 14 (*The Design and Implementation of the 4.4BSD Operating System*)

8. Reference 13 (*Solaris Internals: Core Kernel Architecture*)

9. The term crash is used in this paper as shorthand for the situation where a file system has not been cleanly unmounted. This may happen for a number of reasons, for example, power outages, panics, or hardware failures. The net effect is a file system of dubious consistency.

10. Reference 9 (*Performance Benchmark Report: UNIX File System and VERITAS File System 3.5 on Solaris 9 Operating System 12/02 Release*)

UFS logging, in common with other logging file systems, borrows from database technology and adds the concept of transactions and two-phase commit to the file system. Changes to metadata — the transactions — are first journaled to the intent log. Then, and only then, they are committed to the file system. Upon reboot following a crash, the log is replayed and the file system rolls back (that is, ignores) incomplete transactions and applies complete transactions.

The advantage is that a file system that has previously suffered a crash can be made available for use far sooner than waiting for the traditional *fsck* to complete, because *fsck* must scan the entire file system to check its consistency. The time for *fsck* to complete is proportional to the size of the file system. In contrast, the time to replay the log is proportional to the size of the log. Generally this amounts to a few seconds.

Effect of UFS Logging on Performance

Logging was implemented in UFS to provide faster file system recovery times. A by-product of its implementation is faster processing of small files and metadata operations. This situation is somewhat counterintuitive; surely writing to a log and then writing to the file system should take longer than a single file system write?

Not necessarily. Performance can be positively impacted via the cancellation of some physical metadata operations. This occurs when these updates are issued very rapidly, such as when expanding a tar file or other archive. Another example is recursively deleting directories and their contents.

“Without logging, the system is required to force the directory to disk after every file is processed (this is the definition of the phrase writing metadata synchronously); the effect is to write 512 or 2048 bytes every time 14 bytes is changed. When the file system is logging, the log record is pushed to disk when the log record fills, often when the 512-byte block is completed. This results in a $512/14 = 35$ times reduction.”¹¹

Essentially, logging not only provides faster file system recovery times, it also delivers significant performance gains in the specific areas of small file and metadata-intensive operations.

Space Management in UFS

UFS uses block allocation sizes of 4 KB and 8 KB, which provide significantly higher performance than the 512-byte blocks used in the System V file system. To overcome the potential disadvantage of wasting space in unused blocks, UFS uses the notion of file system fragments. Fragments allow a single block to be broken up into two, four, or eight fragments when necessary. The choice is made through the *fragsize* parameter to *mkfs* (see *mkfs_ufs(1M)*). If the block size is 4 KB, possible values are 512 bytes, 1 KB, 2 KB, and 4 KB. When the block size is 8 KB, legal values are 1 KB, 2 KB, 4 KB, and 8 KB. The default value is 1 KB.

ext3

The original Linux file system was a Minix file system. Minix was a small operating system designed for educational purposes, not for production¹². It had a 64-MB file system and filenames were limited to 14 characters. The ext file system, which replaced it in Linux in 1992, increased these limits to 2 GB and 255 characters, but file access, modification, and creation times were missing from the inode and performance was low. ext2, modelled on the Berkeley FFS, had better on-disk layout and improved performance, and became the de facto standard file system for Linux. It extended the size limit to 4 TB and the filename size to 255 bytes.

11. Reference 24 (*Design, Features, and Applicability of Solaris File Systems*)

12. Reference 22 (*Operating Systems Design and Implementation*)

ext3¹³, an evolution of ext2, adds logging for precisely the same reasons as UFS and other file systems — to facilitate fast reboots following system crashes. Another feature of ext3 is that it is entirely forwards and backwards compatible with ext2. An ext3 file system may be remounted as an ext2 file system and vice versa. This compatibility had a significant impact on the adoption of the new file system.

In ext3, logging batches individual transactions into compound transactions in memory prior to committing to disk to aid performance, a process known as checkpointing. While checkpointing is in progress, a new compound transaction is started. While one compound transaction is being written out, another is accumulating. Furthermore, it is possible to specify an alternative location for the log, which can also enhance performance via increased disk bandwidth.

Logging in ext3

Different levels of journaling can be specified as mount options for the file system, which effects both data integrity and performance. This section describes these mount options.

data=journal

Originally, ext3 was designed to perform full *data and metadata* journaling. In this mode, the system journals all changes to the file system, whether they are made to data or metadata. Hence, both can be brought back to a consistent state. The drawback of full data journaling is that it can be slow, although this performance penalty can be mitigated by setting up a relatively large journal.

data=ordered

Recently, a new journaling mode has been added to ext3 that provides the benefits of full journaling but without introducing the severe performance penalty of *data=journal*. This journals metadata only (like UFS). By using this mode, ext3 can provide data and metadata consistency, even though only metadata changes are recorded in the journal. ext3 uses this mode by default.

data=writeback

Some file system workloads show very significant speed improvement with the *data=writeback* option, which provides lower data consistency guarantees than the *data=journal* or *data=ordered* modes. One of these cases involves heavy synchronous writes. Another is creating and deleting large numbers of small files heavily, such as delivering a very large flow of small e-mail messages. This is born out in the results presented in this paper. Using the *data=writeback* option means data consistency guarantees are essentially the same as the ext2 file system; the difference is that file system integrity is maintained continuously during normal operation.

ReiserFS

The ReiserFS journaling file system was created by Hans Reiser and a team of developers at namesys.com. Its development has been assisted by significant investments from hardware and Linux software companies. It was the first journaling file system to be made available for Linux. An interesting aspect of the ReiserFS approach is that namesys.com considers that if a file system is sufficiently feature-rich and meets users performance requirements, users can utilize the file system directly rather than having to layer other data management abstractions, such as databases, upon it. With this goal in mind, the development team made small file performance a principal objective.

13. Reference 23 (*EXT3, Journaling Filesystem*)

Logging in ReiserFS

ReiserFS journals metadata updates. It does not journal changes to file data in the same way that ext3 can. As in the case of UFS, this guarantees the consistency of the file system following a crash, although the data in the files may be inconsistent.

Space Management in ReiserFS

ReiserFS is designed to support variable block sizes ranging from 512 bytes to 64 KB. Only the 4-KB block size is implemented at present.

Directories and other file system data structures make use of B*Trees, which enhance directory lookups, insertions of names into large directories, and high-speed file deletion. As in the case of UFS, ReiserFS can store data in block fragments to minimize space wastage. ReiserFS uses the term *tail* to refer to the end portion of a file that is smaller than a block, or an entire file that is smaller than a block. ReiserFS uses a mechanism called tail packing to combine small files and file fragments and store the combined data directly in the B*Tree leafnode¹⁴. Because tail packing has some performance overhead, it is implemented as a mount option.

ReiserFS is the default file system for SUSE LINUX, and in some ways, is a competitor to ext3.

14. Reference 4 (*Journal File Systems*)

Chapter 3

Systems and Benchmark Tests

This chapter provides the configuration details of the systems tested, including hardware, operating systems, and file systems, as well as an overview of the PostMark benchmark. The level of detail presented is sufficient to re-create the benchmark. Please note that both the Solaris OS and SUSE LINUX were run out of the box for this test, without any tuning.

Hardware Configuration

A Sun Fire™ V65x server equipped with two 3.06-GHz Intel Xeon CPUs, each with 1-MB cache, was used for the tests. The main memory consisted of 1 GB of registered DDR-266 ECC SDRAM. The server was connected to a Sun StorEdge 3310 SCSI array containing 12 Hitachi DK32EJ-36NC 36G 10000 rpm disks.

The PostMark Benchmark

Introduction to PostMark

Network Appliance's PostMark has become an industry-standard benchmark for small file and metadata-intensive workloads. As outlined above, it was designed to emulate Internet applications such as e-mail, netnews, and e-commerce. More detail on the benchmark can be found in *PostMark: A New File System Benchmark*¹⁵.

PostMark works by creating a configurable sized pool of random text files ranging in size between configurable high and low bounds. These files are continually modified. The building of the file pool allows for the production of statistics on small file creation performance. Transactions are then performed on the pool. Each transaction consists of a pair of create/delete or read/append operations. The use of randomization and the ability to parameterize the proportion of create/delete to read/append operations is used to overcome the effects of caching in various parts of the system.

15. Reference 8 (*PostMark: A New File System Benchmark*)

The Scripted Load

The PostMark program interprets a script describing the workload to be applied. This benchmark used the following parameters.

```

set size 10000 15000      (Sets low and high bounds of files)
set number 20000         (Sets number of simultaneous files)
set transactions RUN_LOAD (Sets number of transactions)
set report verbose
run
quit
RUN_LOAD={25000, 50000, 75000, 100000, 125000, 250000, 300000,
400000, 500000}

```

Table 1. PostMark Parameter File

Software Versions

Software	Version
Solaris 9 OS	Solaris 9 4/04
SUSE LINUX kernel	2.6.5-7.5-smp
LVM2	2.00.09-17.3
ReiserFS	3.6.13-24.1
PostMark	v 1.5 (3/37/01)
ext3	Shipped as core SUSE LINUX kernel

Table 2. Linux and Solaris Software Versions

Solaris Volume Striping and Mount Options

Simple stripes were used of the form:

```

d0 1 8 c0t0d0s2 c0t0d1s2 c0t0d2s2 c0t0d3s2 c0t0d4s2 c0t0d5s2 c0t0d6s2 c0t0d7s2
-i 128b

```

Linux Volume Striping and Mount Options

Default mount option parameters from `mkfs.reiserfs` and `mkfs.ext3` were used with the following exceptions:

- For SUSE LINUX examples, the results posted are with 4 KB specified in the size for the volume created. This was used because it matches perfectly the page size and file system block sizes.
- Different data ordering parameters for the ext3 file system were used as appropriate, such as `order=data`, `order=journal`, `order=writeback`.
- For ReiserFS, the `notail` option was used where appropriate to disable tail packing. The stripes were created with the following commands:

```
pvcreate /dev/sda; pvcreate /dev/sdb; pvcreate /dev/sdc
pvcreate /dev/sdd; pvcreate /dev/sde; pvcreate /dev/sdf
pvcreate /dev/sdg; pvcreate /dev/sdh

vgcreate v0 /dev/sda /dev/sdb /dev/sdc /dev/sdd /dev/sde /dev/sdf \
/dev/sdg /dev/sdh

lvcreate -v -i8 -I4 -L800G -nd0 v0
```

Table 3. Linux Volume Creation Parameters

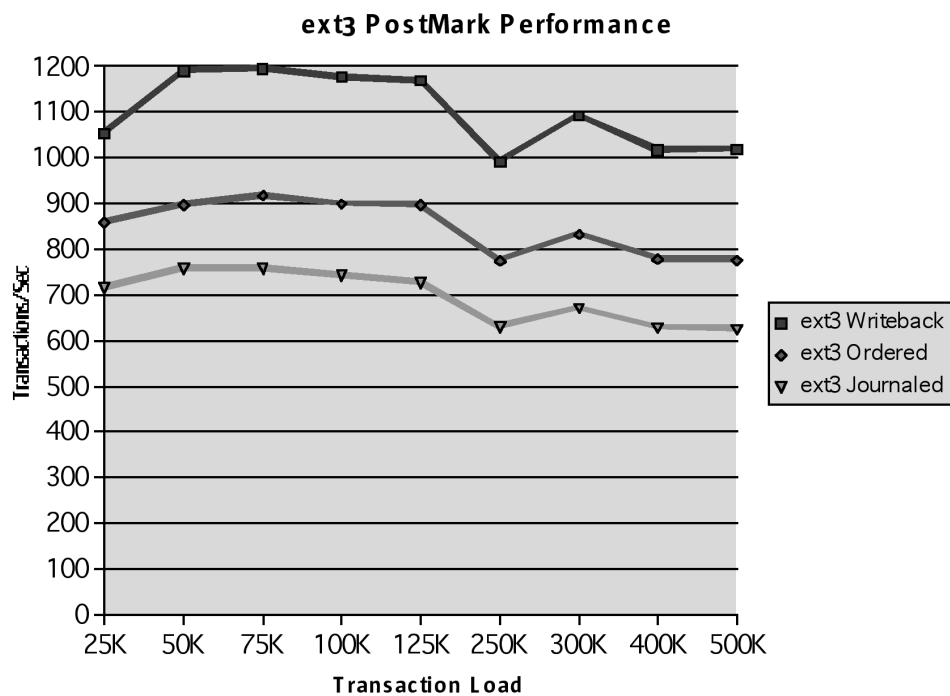
Chapter 4

Results

The following graphs show the effects of applying increasing transaction loads to the various file systems. As this happens, performance naturally degrades. Larger figures indicate higher performance.

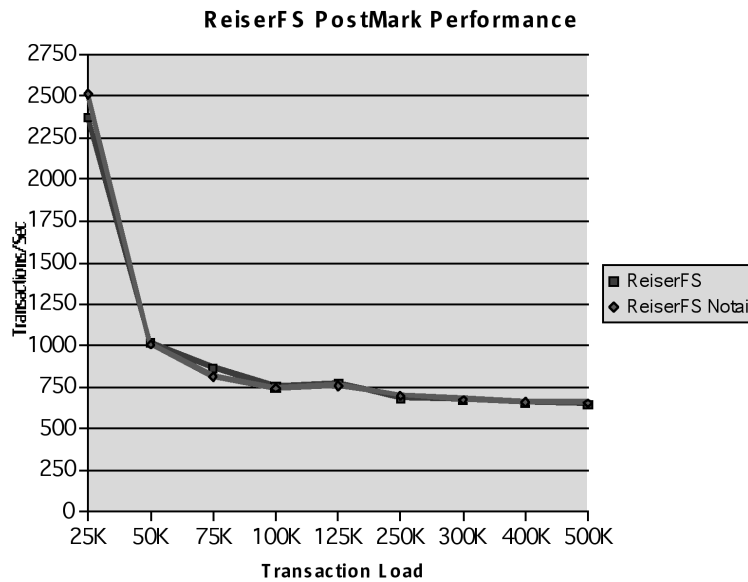
ext3

As discussed in Chapter 2, *Logging in ext3*, each of the journaling options to the mount(1M) command has a concomitant overhead. The following graph clearly illustrates this data integrity/performance trade-off.



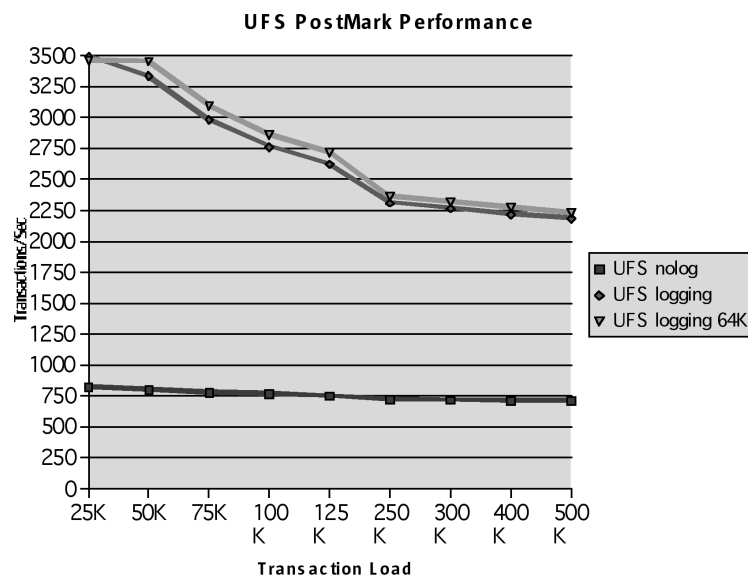
ReiserFS

The following graph shows that there is only marginal performance degradation in using the tail-packing option to the mount command.



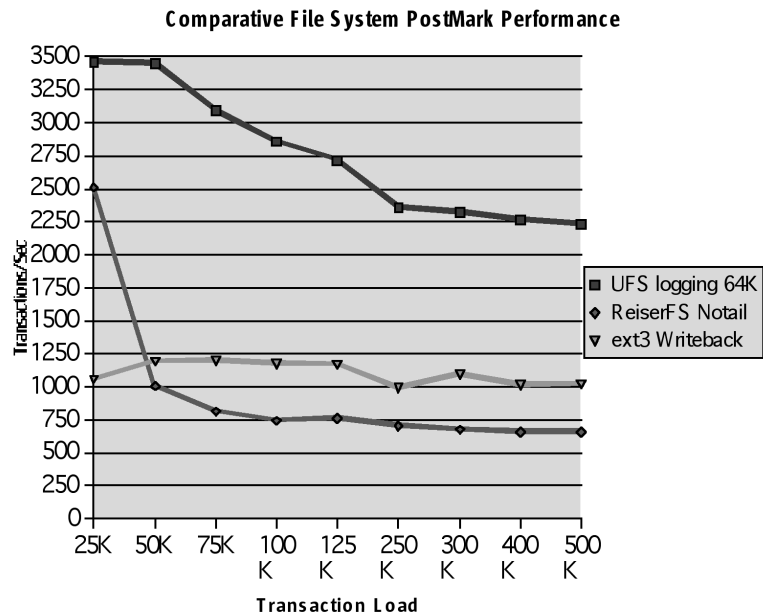
UFS

Within the available options, UFS shows the most variation in possible performance outcomes; the effect of mounting the file system with the logging option is dramatic. Further performance gains can be achieved by tuning the stripe size of the volume manager using metainit(1M).



Comparative Performance

The final graph in this chapter compares the results achieved using the best-performing option for each file system against each other.



Chapter 5

Conclusion

This paper presents initial testing of metadata-intensive file system operations in the context of the PostMark benchmark. This has shown that, for the conditions tested, UFS in the Solaris OS can significantly outperform ext3 and ReiserFS in SUSE LINUX for certain workloads.

The following points should also be taken into consideration:

- These tests were performed on a Sun Fire server with powerful processors, a large memory configuration, and a very wide interface to an array of high-speed disks to ensure that the fewest possible factors would inhibit file system performance. It is possible that the differences between file systems would be less pronounced on a less powerful system simply because all file systems would run into bottlenecks in moving data to the disks.
- A file system and its performance do not exist in a vacuum. The file system performs only as well as the hardware and operating system infrastructure surrounding it, such as the virtual memory subsystem, kernel threading implementation, and device drivers. As such, Sun's overall enhancements in the Solaris OS, combined with high-powered Sun Fire servers, will provide customers with high levels of performance for applications and network services. Additionally, proof-of-concept (POC) implementations are invaluable in supporting purchasing decisions for specific configurations and applications.
- Benchmarks provide general guidance to performance. The conclusion that can be drawn from these tests is that in application areas such as e-mail, netnews, and e-commerce, Solaris UFS performs best in an "apples-to-apples" comparison with Linux ext3 and ReiserFS benchmarks. Again, POCs and real-world customer testing help evaluate performance for specific applications and services.
- This paper has demonstrated the superior performance of the Solaris OS over Linux, as measured by the PostMark benchmark. Combined with low acquisition and ownership costs, integrated Sun and open source applications, and enterprise-class security, availability, and scalability, the Solaris OS provides x86 customers with a price/performance ratio that is unmatched by comparable competitive offerings.

Further Work

It is hoped that further work will enable this paper to be expanded in the future;

- Reiser4. When it is available, the new release of ReiserFS can be evaluated.
- Statistical metrics of scalability. Additional information on file system scalability can be derived from the available data³.
- Performance of file systems under parallel PostMark loads. More extensive testing of the effects of parallel application loads¹⁶ would yield a richer picture of comparative performance.
- Testing of Solaris 10 UFS and Solaris ZFS (zettabyte file system). When it is generally available, the new release of the Solaris OS, which includes many refinements to the core kernel as well as a new file system, can be evaluated.
- Incorporation of feedback to this paper.

Acknowledgments

The author, Dominic Kay, gratefully acknowledges the work of Gleb Reys, Damien Farnham, and the team at the Solaris Performance PIT in Dublin, Ireland in furnishing the test data for this study. Thanks to those who took the time to review the paper: Maureen Chew, Richard Croucher, Paul Eggleton, Gillian Gover, Stephen Harpster, Bill Moffitt, Neil Perrin, and Larry Wake. The author would be grateful for any comments and suggestions on how it might be improved; comments can be sent to dakay@sun.com.

16. Reference 9 (*Performance Benchmark Report: UNIX File System and VERITAS File System 3.5 on Solaris 9 Operating System 12/02 Release*)

Chapter 6

References

1. *JFS Overview: How the Journalled File System Cuts System Restart Times to the Quick*, Steve Best (2000), www-106.ibm.com/developerworks/library/l-jfs.html
2. *JFS Layout: How the Journalled File System Handles the On-Disk Layout*, Steve Best and Dave Kleikamp (2000), www-106.ibm.com/developerworks/library/l-jfslayout
3. *Design and Implementation of the Second Extended File System*, Remy Card, Theodore Ts'o, and Stephen Tweedie, web.mit.edu/tytso/www/linux/ext2intro.html
4. *Journal File Systems*, Juan I. Santos Florido (2000), Linux Gazette, www.linuxgazette.com/issue55/florido.html
5. *EXT3 File System mini-howto*, Rajesh Fowkar (2001), puggy.symonds.net/~rajesh/howto/ext3/toc.html
6. *Linux Filesystems*, William von Hagen, SAMS Press (2002)
7. *Red Hat's New Journaling File System: ext3*, Michael K. Johnson (2001), www.redhat.com/support/wpapers/redhat/ext3
8. *PostMark: A New File System Benchmark*, Jeffrey Katcher, www.netapp.com/tech_library/3022.html
9. *Performance Benchmark Report: UNIX File System and VERITAS File System 3.5 on Solaris 9 Operating System 12/02 Release*, Dominic Kay (2003), www.sun.com/software/whitepapers/solaris9/filesystem_benchmark.pdf
10. *SAS Intelligence Storage Integration with Sun Data Management*, Dominic Kay and Maureen Chew (2004)
11. *The Design and Implementation of the 4.3 Bsd Unix Operating System: Answer Book*, Samuel J. Leffler and Marshall Kirk McKusick, Addison-Wesley (1991)
12. *Lions' Commentary on UNIX 6th Edition*, John Lions, Annabooks (1996)
13. *Solaris Internals: Core Kernel Architecture*, Jim Mauro and Richard McDougall, Sun Microsystems Press, Prentice Hall (2000)
14. *The Design and Implementation of the 4.4BSD Operating System*, Marshall Kirk McKusick, Keith Bostic, Michael J. Karels, and John S. Quarterman, Addison-Wesley Longman (1996)
15. *Porting the SGI XFS File System to Linux*, Jim Mostek, William Earl, and Dan Koren (SGI) and Russell Cattelan, Kenneth Preslan, and Matthew O Keefe (Sistina Software) (1999), oss.sgi.com/projects/xfspapers/als/als.pdf
16. *UNIX Filesystems: Evolution, Design, and Implementation*, Steve D. Pate, Wiley (2003)
17. *Introducing ext3*, Daniel Robbins (2001), www-106.ibm.com/developerworks/library/l-fs7
18. *Surprises in ext3*, Daniel Robbins (2002), www-106.ibm.com/developerworks/linux/library/l-fs8
19. *Journaling and ReiserFS*, Daniel Robbins (2001), www-106.ibm.com/developerworks/library/l-fs.html

20. *Using ReiserFS and Linux 2.4*, Daniel Robbins (2001), www-106.ibm.com/developerworks/library/l-fs2.html
21. *Sun QFS, Sun SAM-FS, and Sun SAM-QFS File System Administrator's Guide*, Sun Microsystems (2002), docs.sun.com/db/doc/816-2542-10?q=QFS
22. *Operating Systems Design and Implementation*, Andrew S. Tanenbaum and Albert S. Woodhull, Prentice Hall (1987)
23. *EXT3, Journaling Filesystem*, Dr. Stephen Tweedie (2000), <http://olstrans.sourceforge.net/release/OLS2000-ext3/OLS2000-ext3.html>
24. *Design, Features, and Applicability of Solaris File Systems*, Brian Wong (Sun Microsystems, 2004), www.sun.com/blueprints/0104/817-4971.pdf

© 2004 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, CA 95054 USA

All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California.

Sun, Sun Microsystems, the Sun logo, Online: DiskSuite, Solaris, Solstice DiskSuite, Sun Fire, SunOS, Sun StorEdge, and The Network is the Computer are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015(b)(6/95) and DFAR 227.7202-3(a). DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS HELD TO BE LEGALLY INVALID.



Please
Recycle



Adobe PostScript

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA Phone 1-650-960-1300 or 1-800-555-9SUN Web sun.com



Sun Worldwide Sales Offices: Argentina +5411-4317-5600, Australia +61-2-9844-5000, Austria +43-1-60563-0, Belgium +32-2-704-8000, Brazil +55-11-5187-2100, Canada +905-477-6745, Chile +56-2-3724500, Colombia +571-629-2323, Commonwealth of Independent States +7-502-935-8411, Czech Republic +420-2-3300-9311, Denmark +45 4556 5000, Egypt +202-570-9442, Estonia +372-6-308-900, Finland +358-9-525-561, France +33-134-03-00-00, Germany +49-89-46008-0, Greece +30-1-618-8111, Hungary +36-1-489-8900, Iceland +354-563-3010, India-Bangalore +91-80-2298989/2295454; New Delhi +91-11-6106000; Mumbai +91-22-697-8111, Ireland +353-1-8055-666, Israel +972-9-9710500, Italy +39-02-641511, Japan +81-3-5717-5000, Kazakhstan +7-3272-466774, Korea +82-2193-5114, Latvia +371-750-3700, Lithuania +370-729-8468, Luxembourg +352-49 11 33 1, Malaysia +603-21161888, Mexico +52-5-258-6100, The Netherlands +00-31-33-45-15-000, New Zealand-Auckland +64-9-976-6800; Wellington +64-4-462-0780, Norway +47 23 36 96 00, People's Republic of China-Beijing +86-10-6803-5588; Chengdu +86-28-619-9333, Guangzhou +86-20-8755-5900; Shanghai +86-21-6466-1228; Hong Kong +852-2202-6688, Poland +48-22-8747800, Portugal +351-21-4134000, Russia +7-502-935-8411, Saudi Arabia +9661 273 4567, Singapore +65-6438-1888, Slovak Republic +421-2-4342-94-85, South Africa +27 11 256-6300, Spain +34-91-767-6000, Sweden +46-8-631-10-00, Switzerland-German 41-1-908-90-00; French 41-22-999-0444, Taiwan +886-2-8732-9933, Thailand +662-344-6888, Turkey +90-212-335-22-00, United Arab Emirates +9714-3366333, United Kingdom +44 0 1252 420000, United States +1-800-555-9SUN or +1-650-960-1300, Venezuela +58-2-905-3800, or online at sun.com/store

SUN © 2004 Sun Microsystems, Inc. All rights reserved. Sun, Sun Microsystems, the Sun logo, Online: DiskSuite, Solaris, Solstice DiskSuite, Sun Fire, SunOS, Sun StorEdge, and The Network is the Computer are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc. UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd. Information subject to change without notice. 08/04 R1.0