SOLARIS™ ZFS AND MICROSOFT SERVER 2003 NTFS
FILE SYSTEM PERFORMANCE

White Paper
June 2007

Table of Contents

Chapter 1
Executive Overview

The Solaris™ 10 06/06 Operating System (OS) introduces a new data management technology — the Solaris ZFS file system. Replacing the traditionally separate file system and volume manager functionality found in most operating environments, Solaris ZFS provides immense capacity and performance coupled with a proven data integrity model and simplified administrative interface. This white paper explores the performance characteristics and differences of Solaris ZFS and the Microsoft Windows Server 2003 NTFS file system through a series of publicly available benchmarks, including BenchW, Postmark, and others. Testing results reveal:

- Solaris ZFS outperforms the Microsoft Windows Server 2003 NTFS file system by a factor of 1.99 to 15, depending on the test.
- A 2 TB Solaris ZFS file system can be created in 17.5 seconds using two commands. In contrast, creating a similarly sized Microsoft Windows Server 2003 NTFS file system involves extensive command line or graphical user interface interaction and takes nearly four hours to complete.
- Testing of multiple disk arrays was limited due to the inability to create volumes containing more than 32 disks in the Microsoft Windows Server 2003 environment.

Figure 1-1 illustrates the differences between Solaris ZFS and the Microsoft Windows Server 2003 NTFS file system in a number of tests. The results indicate that Solaris ZFS outperforms the Microsoft Windows Server 2003 NTFS file system for a variety of data-intensive applications and can satisfy most data management needs.

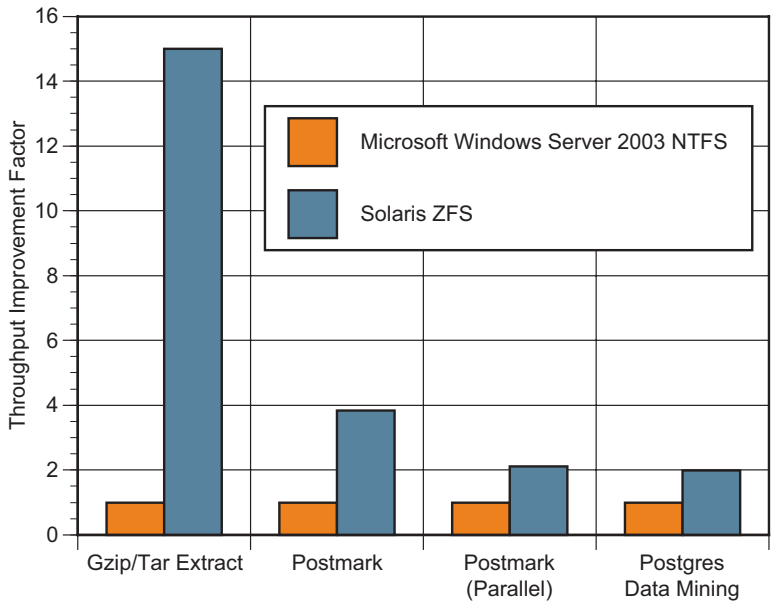

Figure 1-1. Testing summary for the Microsoft Windows Server 2003 NTFS file system and Solaris ZFS

Chapter 2

File System Overview

This chapter provides a brief technical introduction to Solaris ZFS and the Microsoft Windows Server 2003 NTFS file system, and highlights the features that can impact performance. More information can be found in the references listed at the end of this document. Detailed information on the design and operation of the Microsoft Windows Server 2003 environment and file system can be found in *Microsoft Windows Internals,* Fourth Edition, by Mark E. Russinovich and David A. Solomon. Further detail on the design and operation of Solaris ZFS can be found in the *Solaris ZFS Administration Guide* available at *docs.sun.com*, as well as the OpenSolaris Project Web site located at *opensolaris.org/os/community/zfs*.

Solaris™ ZFS

Solaris ZFS is designed to overcome the limitations of existing file systems and volume managers in UNIX® environments.

Simplified Storage Device and File System Administration

In many operating systems, disk partitioning, logical device creation, and new file system formatting tend to be detailed and slow operations. Because these relatively uncommon tasks are only performed by system administrators, there is little pressure to simplify and speed such administrative tasks. Mistakes are easy to make and can have disastrous consequences. As more users handle system administration tasks, it can no longer be assumed that users have undergone specialized training.

In contrast, Solaris ZFS storage administration is automated to a greater degree. Indeed, manual reconfiguration of disk space is virtually unnecessary, but is quick and intuitive when needed. Administrators can add storage space to, or remove it from, an existing file system without unmounting, locking, or interrupting file system service. Administrators simply state an intent, such as *make a new file system*, rather than perform the constituent steps.

Pooled Storage and Integrated Volume Management

The Microsoft Windows Server 2003 file system makes a one-to-one association between a file system and a particular storage device. Using the volume manager provided, the file system is assigned to a specific range of blocks on the logical storage device. Such a scheme is counterintuitive — file systems are intended to virtualize physical storage, and yet a fixed binding remains between the logical namespace and a logical or physical device.

Solaris ZFS decouples the namespace from physical storage in much the same way that virtual memory decouples address spaces from memory banks. Multiple file systems can share a pool of storage. Allocation is moved out of the file system into a storage space allocator that parcels out permanent storage space from a pool of storage devices as file systems make requests. In addition, the volume management and file system model used by the Microsoft Windows Server 2003 NTFS file system makes it difficult to expand and contract file systems, share space, or migrate live data. By folding the functionality and namespace of the volume manager into the file system level, operations can be undertaken in terms of files and file systems rather than devices and blocks.

Strong Data Integrity

File systems such as Microsoft Windows Server 2003 NTFS permit on-disk data to be inconsistent for varying amounts of time. If an unexpected system crash or reboot occurs while the on-disk state is inconsistent, the file system requires repair during the next boot cycle using a combination of utilities such as `CHKDSK` and metadata logging that requires a log replay. On the other hand, Solaris ZFS provides consistent on-disk data and error detection and correction, ensuring data consistency while maintaining high performance levels.

File system corruption can be caused by disk corruption, hardware or firmware failures, or software or administrative errors. Validation at the disk block interface level can only catch some causes of file system corruption. Traditionally, file systems trust the data read in from disk. However, if the file system does not validate read data, errors can result in corrupted data, system panics, or more. File systems should validate data read in from disk in order to detect downstream data corruption, and correct corruption automatically, if possible, by writing the correct block back to the disk. Such a validation strategy is a key design goal for Solaris ZFS.

Immense Capacity

Many of today's file systems utilize 32-bit block addresses and are often limited to a few terabytes in size. Using the largest cluster size available, the Microsoft Windows Server 2003 NTFS file system can support up to 256 TB. However, one petabyte datasets are plausible today, and storage capacity is currently doubling approximately every nine to 12 months. Assuming the rate of growth remains the same, 16 exabyte (EB) datasets may begin to emerge in only ten years. The lifetime of a typical file system implementation is measured in decades. Unlike many of today's file systems, Solaris ZFS uses 128-bit block addresses and incorporates scalable algorithms for directory lookup, metadata allocation, block allocation, I/O scheduling, and other routine operations, and does not depend on repair utilities such as `fsck` or `CHKDSK` to maintain on-disk consistency.

Microsoft Windows Server 2003 NTFS File System

More information on the Microsoft Windows Server 2003 NTFS file system can be found in *NTFS Technical Reference* located at http://technet2.microsoft.com/WindowsServer/en/Library/81cc8a8a-bd32-4786-a849-03245d68d8e41033.mspx

Designed to overcome the limitations of the previous generation FAT32 file system, the Microsoft Windows Server 2003 NTFS file system aims to:

- *Increase reliability.* The Microsoft Windows Server 2003 NTFS file system incorporates a common log file and checkpoint mechanisms to ensure consistency in the event of a failure. The Microsoft Windows Server 2003 NTFS file system recovers metadata by replaying log files during the boot or mount time consistency check performed by the CHKDSK utility. Bad sector errors are handled by dynamically moving the data in the damaged cluster to a new cluster, marking the failed cluster, and avoiding its use.
- *Increase security.*The Microsoft Windows Server 2003 NTFS file system sets permissions on files and directories and specifies the groups and users that can access those items, along with the type of access permitted any access restrictions.
- *Support large volumes.*The Microsoft Windows Server 2003 NTFS file system limits volumes to 16 TB using a 4 KB cluster size, or 256 TB using the maximum 64 KB cluster size. Larger files (up to 16 TB) and more files per volume are now supported, and disk space is managed efficiently by using smaller cluster sizes.

File System Creation

By integrating volume management and file system capabilities, Solaris ZFS eases administrative efforts and simplifies the steps needed to create a file system. Table 2-1 lists the activities required to create usable storage using Solaris ZFS and the Microsoft Windows Server 2003 NTFS file system, as well as the time observed for these tasks. Only 32 disks were configured for the file systems under test, as the Microsoft Windows volume manager supports a maximum of 32 disks per volume.

Table 2-1. The file system creation procedure for Solaris ZFS and the Microsoft Windows Server 2003 NTFS file system

Solaris ZFS	Microsoft Windows Server 2003 NTFS File System
# **zpool create -f tank** (32 disks) # **zfs create tank/fs**	diskpart> **create volume stripe disk=**(32 disks) DiskPart successfully created the volume. diskpart> **assign letter=s** DiskPart successfully assigned the driver letter or mount point. diskpart> **exit** c:/> **format s: /fs:ntfs** The type of the file system is RAW. The new file system is NTFS. WARNING, ALL DATA ON NON-REMOVABLE DISK DRIVES S: WILL BE LOST! Proceed with Format (Y/N)? Y Verifying 840012M Volume label (32 characters, ENTER for none)? **<rtn>** Creating file system structures. Format complete. 860172284 KB total disk space. 860080288 KB are available.
Time: 17.5 seconds	**Time:** 4.5 hours

Alternatively, a graphical volume manager can be used to create the volume and file system. Figure 2-1 shows one such tool that is bundled with the operating system. The volume manager can be executed by going to the `Control Panel->Administrative Tools->Compute Management->Disk Managment`. It is important to note that during the testing effort, the Indexing Service was turned off in the root file system and all subdirectories using the File Manager graphical user interface to ensure unneeded functionality did not hinder performance.

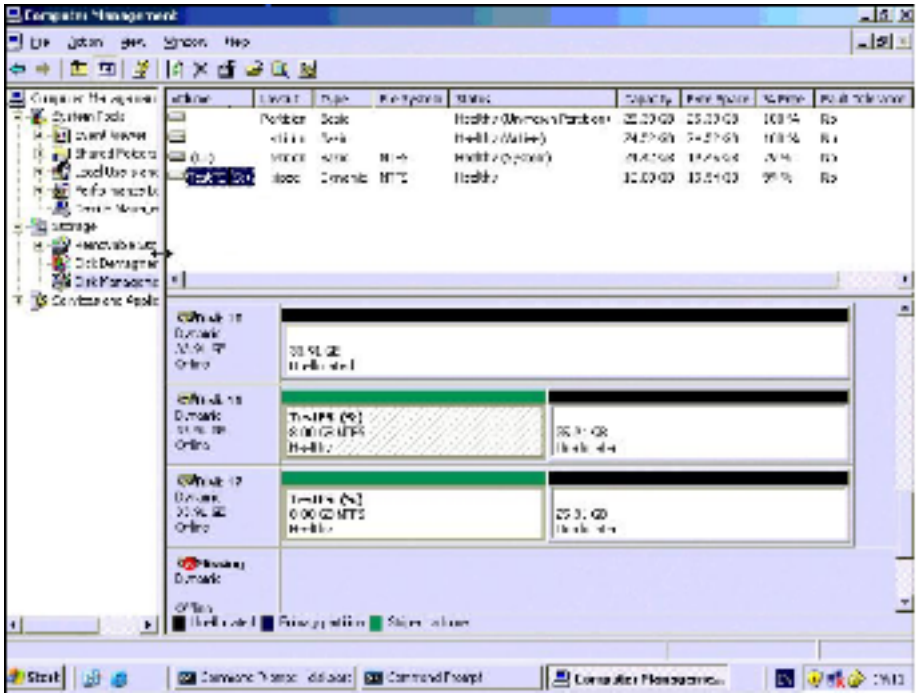

Figure 2-1. The Microsoft Windows Server 2003 Disk Management graphical user interface

Chapter 3
Benchmark Tests and Results

This chapter describes the hardware platforms, operating system and software versions, and benchmarks used for testing efforts, as well as the results observed.

Reproducing the Benchmark

Having default parameters that become outdated creates two problems. First, there is no such thing as a standard configuration. In addition, different workloads exercise the system differently and results across research papers are not comparable. Second, not all research papers precisely describe the parameters used, making it difficult to reproduce results[1].

1. *Benchmarking File System Benchmarks*, N. Joukov, A. Traeger, CP Wright, Zadok, ETechnical Report FSL-05-04b, CS Department, Stony Brook University, 2005. http://www.fsl.cs.sunysb.edu/docs/fsbench/fsbench.pdf

During competitive system software testing, It is important to ensure the underlying server and storage hardware — and the benchmarking code — is as close to identical as possible so that proper comparisons can be drawn. The testing effort described in this document aimed to ensure comparability through:

- Use of the same physical server and storage hardware for all tests
- Use of identical benchmark source code compiled on both platforms
- Use of operating systems that were installed and used out of the box, with no special tuning

Hardware and Operating System Configuration

More information on Microsoft Windows Services for UNIX can be found at http://www.microsoft.com/technet/interopmigration/unix/sfu/default.mspx

Table 3-1 describes the platforms on which the testing was conducted. All testing was performed on Sun Fire™ x64 servers incorporating AMD Opteron™ processors. As all testing was performed prior to the release of the Solaris 10 06/06 OS, a pre-release version of Solaris ZFS was used. Postmark version 1.5 and BenchW 1.1 were used on both platforms. PostgreSQL version 8.0.1 was used on the Solaris OS, while version 8.1.0-2 was used on the Microsoft Windows Server 2003 platform. Note that Postmark runs in a POSIX environment which is available for Microsoft Windows Server 2003 as Microsoft Windows Services for UNIX.

Table 3-1. Hardware and operating system configuration used for testing efforts

Platform	Operating System	Server	Storage
1	Microsoft Windows Server 2003 Enterprise Edition R2	• Sun Fire™ x4200 server • One 2.2 GHz AMD Opteron processor (two core) • 1 GB RAM	• Sun StorEdge™ 3500 array with 12 x 72 GB disks • Fiber channel interface 1 x 2 Gb PCI-X 133 MHz
2	Solaris 10 01/06 OS with Solaris ZFS packages	• Sun Fire™ x4200 server • One 2.2 GHz AMD Opteron processor (two core) • 1GB RAM	• Sun StorEdge™ 3500 array with 12 x 72 GB disks • Fiber channel interface 1 x 2 Gb PCI-X 133 MHz
3	Microsoft Windows Server 2003 Enterprise Edition R2	• Sun Fire™ x4200 server • Two 2.2 GHz AMD Opteron processors (four cores) • 8 GB RAM	• Sun StorEdge 3500 arrays (4) with 32 x 72 GB disks • Fiber channel interface 4 x 2 Gb PCI-X 133 MHz
4	Solaris 10 06/06 OS with Solaris ZFS packages	• Sun Fire™ x4200 server • Two 2.2 GHz AMD Opteron processors (four cores) • 8 GB RAM	• Sun StorEdge 3500 arrays (4) with 32 x 72 GB disks • Fiber channel interface 4 x 2 Gb PCI-X 133 MHz

Postmark — A Web and Mail Server Benchmark

Designed to emulate applications such as software development, email, newsgroup servers and Web applications, the Postmark utility from Network Appliance has emerged as an industry-standard benchmark for small file and metadata-intensive workloads. More information on the Postmark benchmark can be found in *Postmark: A New File System Benchmark*.

Postmark works by creating a pool of random text files that range in size between configurable high and low bounds. During the test, the files are modified continually. Building the file pool enables 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 helps overcome the effects of caching in various parts of the system.

Single Threaded Postmark Test

In the first set of tests, Postmark ran as a single threaded process that iterated over an increasing number of created files and transactions. Table 3-2 defines the terms small, medium, and large that are referenced in the graphs and discussions that follow.

Table 3-2. Single threaded Postmark variables

Test	Files	Transactions
Small	1,000	50,000
Medium	20,000	50,000
Large	20,000	100,000

Table 3-3 lists the parameters and settings used during testing efforts.

Table 3-3. Single threaded Postmark parameters

Parameter Syntax	Comment
`set number 1000`	Number of files (variable)
`set transactions 50000`	Number of transactions (variable)
`set subdirectories 1000`	Directories across which the files are scattered
`set size 10000 15000`	Range of file sizes (9.77 KB to 14.65 KB)
`set location /cygdrive/s`	Root directory of the test (target file system)
`set buffering false`	Do not use buffering in the C library
`set report verbose`	
`random`	The random number generator seed (default 42)

Figure 3-1 illustrates the results obtained on the single threaded Postmark test. Platforms 3 and 4, configured as described in Table 3-1, were used for testing efforts.

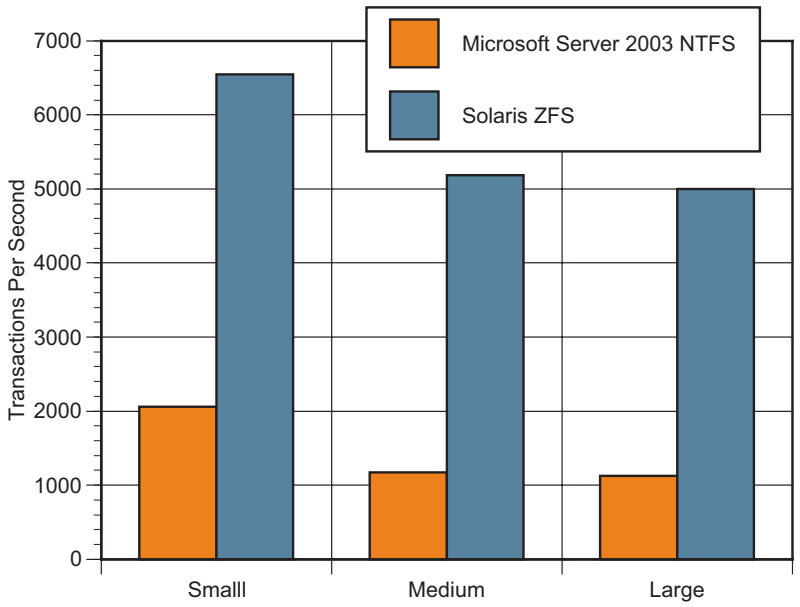

Figure 3-1. Single threaded Postmark test results

Multithreaded Postmark Test

The multithreaded Postmark test is designed to simulate a server hosting multiple applications. In this test, Postmark ran as multiple parallel processes in a scripted harness designed to stress the multithreaded capabilities of the file system and operating system. The test iterates over an increasing number of parallel Postmark instances and increasing transaction workloads. During this test, iterations of six to 16 parallel processes were carried out on workloads of 25,000 transactions, 100,000 transactions and 300,000 transactions.

Table 3-4 details the statistics obtained during multithreaded Postmark testing. Testing was performed on platforms 1 and 2, configured as described in Table 3-1.

Table 3-4. Threads per workload achieved during multithreaded Postmark testing

	Threads/Workload				
Number of Transactions	**6 Threads**	**8 Threads**	**12 Threads**	**14 Threads**	**16 Threads**
25,000	431/206	378/168	273/122	228/103	207/96
50,000	285/208	252/170	226/121	218/107	203/91
75,000	295/199	241/162	211/119	201/107	204/91
100,000	285/205	253/157	228/116	222/107	223/96
200,000	269/191	236/147	220/111	225/98	212/85
300,000	260/184	225/141	225/100	205/79	192/91

Figure 3-2 illustrates the results obtained.

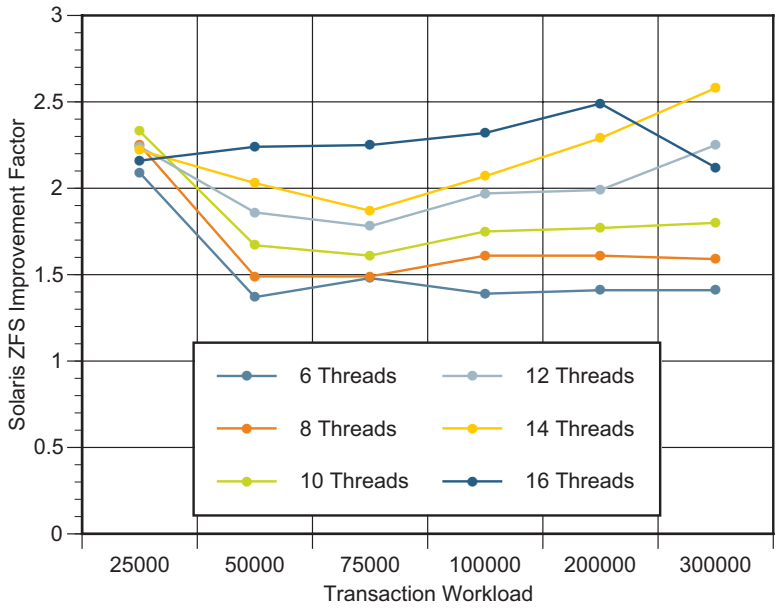

Figure 3-2. Multithreaded Postmark test results

PostgreSQL and BenchW — A Data Warehousing Benchmark

More information can be found at http://postgresql.org and http://benchw.sourceforge.net

PostgreSQL is an open source relational database manager released under a BSD license. Note that PostgreSQL is integrated into the Solaris 10 06/06 OS, and Sun provides support to organizations looking to develop and deploy open source database solutions and use PostgreSQL in enterprise environments. PostgreSQL was used for testing efforts as it is can be deployed in both Solaris OS and Microsoft Windows Server 2003 environments.

BenchW is a data warehouse benchmarking toolkit that aims to compare the capabilities of several different database managers for data warehouse activities, such as data loading, index creation, and query performance. The BenchW benchmark attempts to keep things simple and realistically model the environment in which many ad hoc query tools work. As a result, many of the elaborate tuning optimizations for data warehousing are not used. The comparison concentrates on bulk queries rather than testing multiple threads or concurrent updates. The testing effort documented here used the BenchW benchmark because of its wide availability and ability to work with PostgreSQL. In addition to PostgreSQL, BenchW can generate appropriate code and data for several commercial and community-based relational database managers.

BenchW Test

BenchW uses a data model in an idealized *star* schema with three default dimension tables: `dim0`, `dim1`, `dim2`. The first dimension is time, and includes a date column representing a time measure. The other two dimensions are generic. A fact table, `fact0`, represents generic observations of interest. The *star* relationship is expressed through the inclusion of a foreign key column in the fact table for each primary key of the dimension tables. These relationships are not formally represented as constraints, since they cannot be implemented by all database managers. During testing efforts, the code was modified to encapsulate the whole suite of tests in one shell script and capture timing information in SQL.

The default tuning settings described in the BenchW documentation were used across all operating systems for the testing effort. The settings are listed below. Only one variable, `checkpoint_segments`, used a non-default value. The value was raised for both the Solaris OS and Microsoft Windows Server 2003 environments to prevent excessive checkpointing and warning messages. While using default values constrains database managers, it provides a level basis across platforms.

```
shared_buffers = 10000
sort_mem = 20480
effective_cache_size = 48000
random_page_cost = 0.8
wal_buffers = 1024
checkpoint_segments = 32
```

The SQL script used during testing efforts is listed in Appendix A. The tests documented here used the `loadgen` utility to generate a 40 GB main table.

Figure 3-3 depicts the Solaris ZFS improvement factor over Microsoft Windows Server 2003 NTFS during BenchW testing on indexed and unindexed queries.

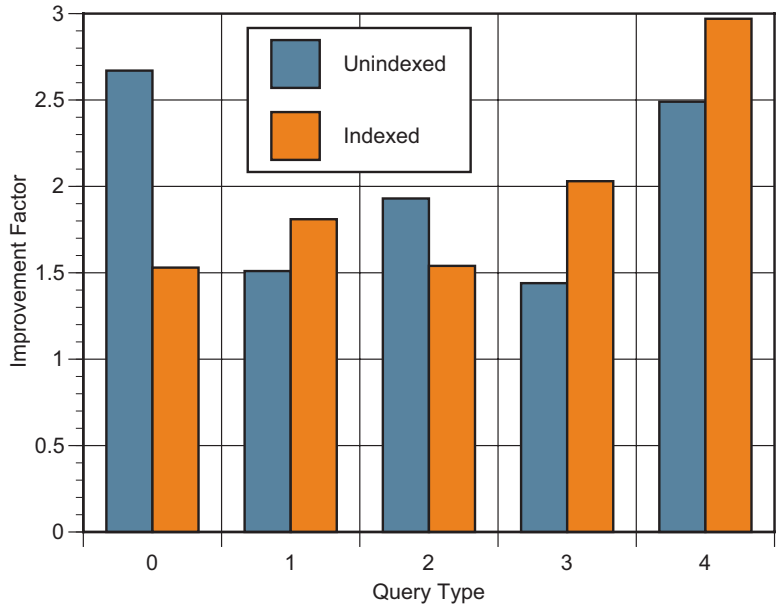

Figure 3-3. BenchW test results

Compressed Archives — A System Configuration Benchmark

Often discussed on the OpenSolaris forum related to Solaris ZFS, the compressed archives benchmark appears to be a real world indicator of file system performance. Indeed, unpacking an archive downloaded from the Internet or copied from removable media is a common task. Furthermore, making `tar(1)` run as fast as `dd(1)` is a good objective for file system implementations.

Easily reproduced across platforms and operating systems, the unarchive test uses the `freedb-complete-20060305.tar.gz` compressed archive consisting of 12 directories containing 1,972,765 files. For purposes of the test, the compressed archive resides in the same file system in which it is to be unarchived. In addition, the original `bzip2archive` was repackaged to use `gzip`, as `gzip` is available on Microsoft Windows Server 2003 environments.

Figure 3-4 shows the results obtained during testing efforts. Date was gathered using the `time(1)` command. Results reveal that Solaris ZFS required 3.22 minutes to complete (3.32 minutes with compression), while Microsoft Windows Server 2003 NTFS required 48.6 minutes (48.02 minutes with compression) to complete.

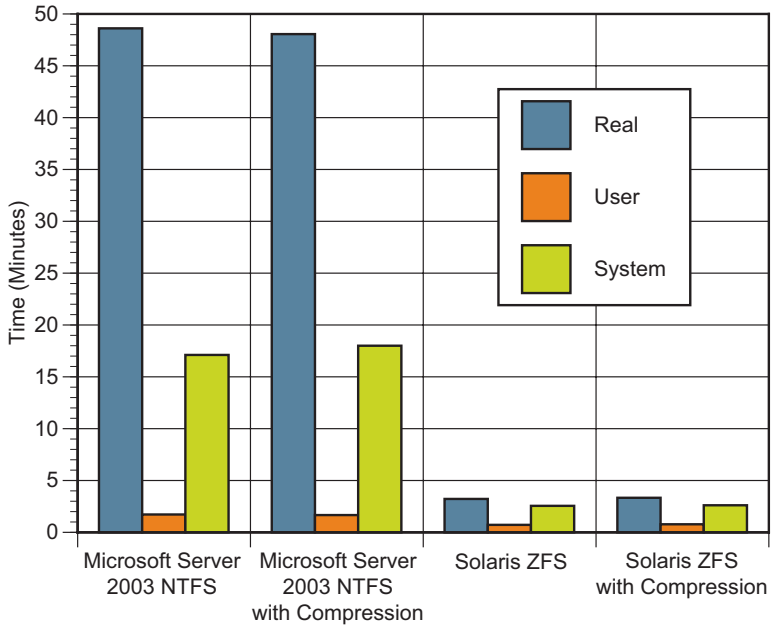

Figure 3-4. Unarchive performance test results

Chapter 4
Summary

This white paper details the performance characteristics of Solaris ZFS and the Microsoft Windows Server 2003 NTFS file system through Postmark, BenchW, and unarchive benchmark testing. For the conditions tested, Solaris ZFS can significantly outperform the Microsoft Windows Server 2003 NTFS file system for many workloads.

The following points should be taken into consideration:

- The tests were performed on Sun Fire systems incorporating powerful processors, a large memory configuration, and a very wide interface to an array of high-speed disks to ensure tat the fewest possible factors would inhibit file system performance. It is possible that the file system differences could be reduced on a less powerful system simply because all file systems could run into bottlenecks in moving data to the disks.
- 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 10 Operating System, combined with powerful Sun servers, can provide organizations with high application performance levels. Additionally, proof-of-concept implementations are invaluable in supporting purchasing decisions for specific configurations and applications.
- Benchmarks provide general guidance to performance. The test results presented in this document suggest that in application areas such as e-mail, Web server and software development, Solaris ZFS performs better in a side by side comparison with the Microsoft Windows Server 2003 NTFS file system. Proof-of-concepts and real world testing can help evaluate performance for specific applications and services.
- With low acquisition and ownership costs, integrated Sun and open source applications, and enterprise-class security, availability and scalability, the Solaris OS provides organizations with an attractive price/performance ratio over solutions from other vendors.

For More Information

More information on Solaris ZFS can be found in the references listed in Table 4-1 below.

Table 4-1. Related Web sites

Description or Title	URL
OpenSolaris Project	opensolaris.org
Solaris Operating System	sun.com/solaris
Solaris ZFS	sun.com/solaris
Solaris ZFS Administration Guide	docs.sun.com (Document Number: 817-2271-2)
Sun Announces Support for Postgres Database on Solaris 10	sun.com/smi/Press/sunflash/2005-11/sunflash.20051117.1.xml

More information on the ext3 file system and related topics can be found in the following:

Benchmarking File System Benchmarks, N. Joukov, A. Traeger, CP Wright, Zadok, ETechnical Report FSL-05-04b, CS Department, Stony Brook University, 2005. http://www.fsl.cs.sunysb.edu/docs/fsbench/fsbench.pdf

Postmark: A New File System Benchmark, Jeffrey Katcher, Netapps Tech Report 3022, 1997. http://www.netapp.com/tech_library/3022.html

Performance Benchmark Report: UNIX File System and VERITAS File System 3.5 on Solaris 9 Operating System 12/02 Release, Dominic Kay, 2003. http://sun.com/software/whitepapers/solaris9/filesystem_benchmarkd.pdf

File System Performance: The Solaris OS, UFS, Linux ext3, and ReiserFS, August 2004. http://sun.com/software/whitepapers/solaris10/fs_performance.pdf

NTFS Technical Reference. http://technet2.microsoft.com/WindowsServer/en/Library/81cc8a8a-bd32-4786-a849-03245d68d8e41033.mspx

Microsoft Windows Internals, Fourth Edition, Mark E. Russinovich, David A. Solomon, Microsoft Press, 2005.

Microsoft Windows Command Line, William R. Stanek, Microsoft Press, 2004.

Appendix A
BenchW SQL Used During Testing

```
Appendix - Copyright for BenchW

Copyright 2002 Mark Kirkwood. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification,are permitted provided that the following conditions are
met:

1. Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESS OR IMPLIED
WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN
NO EVENT SHALL THE MARK KIRKWOOD OR CONTRIBUTORS BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.

The views and conclusions contained in the software and documentation
are those of the author(s).
```

```
-- Rolled up benchw transaction scripts with timing functionality.
-- Timing info not part of original benchmark
-- Before this script run "psql -d template1 -c "CREATE DATABASE benchw"
-- Run this script as "psql -d benchw -f main.sql
-- After this script run "psql -c "DROP DATABASE benchw"

SET DATESTYLE TO 'ISO';

CREATE TABLE tasklog(taskname char(15), timebegun timestamp, timefinished timestamp);

--  benchw schema : generated by schemagen

CREATE TABLE dim0 (d0key INTEGER NOT NULL, ddate DATE NOT NULL,
dyr INTEGER NOT NULL, dmth INTEGER NOT NULL, dday INTEGER NOT NULL);

CREATE TABLE dim1 (d1key INTEGER NOT NULL, dat VARCHAR(100) NOT NULL,
dattyp VARCHAR(20) NOT NULL, dfill VARCHAR(100) NOT NULL);

CREATE TABLE dim2 (d2key INTEGER NOT NULL,dat VARCHAR(100) NOT NULL,
dattyp VARCHAR(20) NOT NULL, dfill VARCHAR(100) NOT NULL);

CREATE TABLE fact0 (d0key INTEGER NOT NULL, d1key INTEGER NOT NULL, d2key
INTEGER NOT NULL, fval INTEGER NOT NULL,ffill VARCHAR(100) NOT NULL);

--  benchw load ; paths Windows style, modify appropriately for other OSs

COPY dim0 FROM '\\dump/dim0.dat' USING DELIMITERS ',';
COPY dim1 FROM '\\dump/dim1.dat' USING DELIMITERS ',';
COPY dim2 FROM '\\dump/dim2.dat' USING DELIMITERS ',';

INSERT INTO tasklog (taskname, timebegun) VALUES ('MainTableLoad', 'now');

COPY fact0 FROM '\\dump/fact0.dat' USING DELIMITERS ',';

UPDATE tasklog SET timefinished = 'now' WHERE taskname = 'MainTableLoad' ;

--Start: Pre-index tests : query type 0

INSERT INTO tasklog (taskname, timebegun) VALUES ('qtype0noindex', 'now');
SELECT d0.dmth, count(f.fval ) FROM dim0 AS d0, fact0 AS f WHERE
d0.d0key = f.d0key AND d0.ddate BETWEEN '2010-01-01' AND '2010-12-28'
GROUP BY d0.dmth ;
UPDATE tasklog SET timefinished = 'now' WHERE taskname = 'qtype0noindex' ;

-- : query type 1

INSERT INTO tasklog (taskname, timebegun) VALUES ('qtype1noindex', 'now');
SELECT d0.dmth,count(f.fval )FROMdim0 AS d0,fact0 AS f WHERE
d0.d0key = f.d0key AND d0.dyr = 2010
GROUP BY d0.dmth ;
UPDATE tasklog SET timefinished = 'now' WHERE taskname = 'qtype1noindex' ;
```

```
-- : query type 2

INSERT INTO tasklog (taskname, timebegun) VALUES ('qtype2noindex', 'now');
SELECT d0.dmth,d1.dat,count(f.fval )FROM dim0 AS d0, dim1 AS d1,fact0 AS f
WHERE d0.d0key = f.d0key AND d1.d1key = f.d1key AND d0.dyr BETWEEN 2010 AND 2015
AND d1.dattyp BETWEEN '10th measure type' AND '14th measure type'
GROUP BY d0.dmth, d1.dat ;
UPDATE tasklog SET timefinished = 'now' WHERE taskname = 'qtype2noindex' ;

-- : query type 3

INSERT INTO tasklog (taskname, timebegun) VALUES ('qtype3noindex', 'now');
SELECT d0.dmth, d1.dat, d2.dat, count(f.fval )FROM dim0 AS d0, dim1 AS d1,
dim2 AS d2, fact0 AS f
WHERE d0.d0key = f.d0key AND d1.d1key = f.d1key AND d2.d2key = f.d2key
AND d0.dyr BETWEEN 2010 AND 2015 AND d1.dattyp BETWEEN '10th measure type' AND '14th
measure type' AND d2.dattyp BETWEEN '1th measure type' AND '4th measure type'
GROUP BY d0.dmth, d1.dat, d2.dat ;
UPDATE tasklog SET timefinished = 'now' WHERE taskname = 'qtype3noindex' ;

-- : query type 4

INSERT INTO tasklog (taskname, timebegun) VALUES ('qtype4noindex', 'now');
SELECT d0.dyr, count(f.fval )FROM dim0 AS d0, fact0 AS f WHERE d0.d0key = f.d0key
AND d0.dyr < 2010
GROUP BY d0.dyr ;
UPDATE tasklog SET timefinished = 'now' WHERE taskname = 'qtype4noindex';

--  benchw (suggested) indexes : generated by schemagen

INSERT INTO tasklog (taskname, timebegun) VALUES ('IndexCreate', 'now');

CREATE UNIQUE INDEX dim0_d0key ON dim0(d0key) ;
CREATE UNIQUE INDEX dim1_d1key ON dim1(d1key) ;
CREATE UNIQUE INDEX dim2_d2key ON dim2(d2key) ;
CREATE INDEX fact0_d0key ON fact0(d0key) ;
CREATE INDEX fact0_d1key ON fact0(d1key) ;
CREATE INDEX fact0_d2key ON fact0(d2key) ;

UPDATE tasklog SET timefinished = 'now' WHERE taskname = 'IndexCreate' ;

ANALYSE dim0; ANALYSE dim1; ANALYSE dim2; ANALYSE fact0;
-- The same queries are then run again with the benefit of indexes and analysis.

SELECT taskname, timefinished - timebegun AS timespent FROM tasklog;
```

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA **Phone** 1-650-960-1300 or 1-800-555-9SUN (9786) **Web** sun.com

SunWIN #505691 Lit #STWP12871-0 06/07