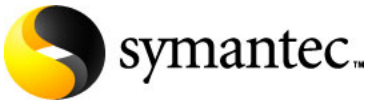

Veritas Storage Foundation™ and Sun Solaris™ ZFS

A performance study based on commercial workloads

August 02, 2007

Introduction **3**

Executive Summary **4**

About Veritas Storage Foundation **5**

Veritas Storage Foundation 5
Veritas Storage Foundation for Databases 5

About Sun Solaris™ ZFS **6**

Solaris ZFS 6

Hardware and Software specifications **7**

Hardware 7
Software 7

File serving benchmark **8**

Benchmark software 8
Configuration 8
ZFS configuration 8
Veritas Storage Foundation configuration 8
Benchmark results 9
Sustained throughput 9
System Resources: CPU usage 10
System Resources: I/O bandwidth 11
System Resources: Memory utilization 12

Database benchmark **13**

Benchmark software 13
Configuration 14
TPC-C benchmark settings 14
Oracle settings 14
ZFS configuration 14
Veritas Storage Foundation for Oracle configuration 15
Benchmark results 16
Transaction throughput 16
System and Storage resource utilization 17

Conclusion **18**

Introduction

This whitepaper compares how Veritas Storage Foundation and Solaris ZFS perform for commercial workloads. The paper describes not only the raw performance numbers, but also the system and storage resources required to achieve those numbers.

Pure performance numbers are interesting, but what is often ignored is how much CPU time, memory or I/O to disk was required to reach that number. In a performance benchmark situation, the resources used are less relevant, but for sizing server and storage infrastructure in the data center it is vital information. To see the bigger picture both performance numbers and resource utilization need to be considered.

This white paper covers two of the most common deployment scenarios for Veritas Storage Foundation, as a file server and as a database server.

Executive Summary

Solaris ZFS is a new file system with an integrated volume manager that was released with Update 2 of Solaris 10 in mid-2006. It is a strong replacement for Sun's previous products Sun Volume Manager and UFS, but its design and immaturity still make it a less than ideal fit for enterprise deployment. Storage Foundation is based on the Veritas Volume Manager and Veritas File System and is a proven leader in storage management software and mission-critical enterprise class deployments.

As shown in this document, Veritas Storage Foundation consistently performs about 2.7 times more operations per second than ZFS in Solaris 10 Update 3 for NFS file serving and as much as 2.3-5 times more transactions per second for OLTP databases. ZFS also required more system resources from the server and storage systems.

The non-overwriting design of the file system together with aggressive read-ahead algorithms and sub-optimal handling of synchronous writes has ZFS reading up to 48 times more data than Veritas Storage Foundation for the same file serving workload. For synchronous writes the current implementation of ZFS also requires two separate writes, one to the file system log and one to the file.

In single host environments and direct attached storage this behavior is only limiting a single host, but in a data center and in a SAN (Storage Area Network) environment, the increased bandwidth requirement will have a significant impact on the size and performance of the storage infrastructure.

Veritas Storage Foundation

Veritas Storage Foundation provides easy-to-use online storage management, enables high availability of data, optimized I/O performance, and allows freedom of choice in storage hardware investments. Veritas Storage Foundation is the base storage management offering from Symantec. It includes Veritas™ File System and Veritas™ Volume Manager. Veritas Storage Foundation includes advanced features such as a journaling file system, storage checkpoints, dynamic multi-pathing, off-host processing, volume snapshots and dynamic tiered storage. Storage Foundation comes in three flavors, Basic, Standard and Enterprise, each targeted for different environments:

Storage Foundation works across all major UNIX, Linux and Windows operating systems and has broad storage array support. All instances can be centrally monitored/managed across thousands of hosts.

http://www.symantec.com/enterprise/products/overview.jsp?pcid=1020&pvid=203_1

Veritas Storage Foundation for Databases

Storage Foundation for Databases represents integrated suites of industry-leading Veritas technologies that deliver easier manageability, superior performance and continuous access to Oracle, DB2 and Sybase. This suite is built on Veritas Storage Foundation, a storage infrastructure layer that enables database storage optimization with online storage virtualization and RAID. Storage Foundation for Databases also delivers the manageability of file systems with the performance of raw devices through accelerator technologies such as ODM (Oracle), CIO (DB2) and Quick I/O (Sybase).

http://www.symantec.com/enterprise/products/overview.jsp?pcid=2245&pvid=208_1

Solaris ZFS

In June 2006, Sun introduced the Solaris ZFS (Zettabyte File System) 128-bit file system as part of the Solaris 10 operating system. Rather than having separate file systems and volume manager, Sun designed ZFS as a new file system intended to replace their older generation Solaris Volume Manager (SVM) and file system (UFS). Sun describes ZFS as a file system that replaces the need for a separate volume manager, through an architecture that shares free space in storage pools. ZFS incorporates advanced features including snapshots, clones, storage pool management, and checksums. These address many of the shortcomings in SVM and UFS, but with the focus on the needs of direct attach customers rather than the enterprise-class SAN customers.

Hardware

The following hardware configuration was used for all tests.

Server

Sun Fire E6800 with 12 750 MHz UltraSparc-III processors and 12 GB RAM

Clients

16 SUN Netra T1 with 1 UltraSparc-III 400 MHz

Storage

4 StorEdge 3510 arrays each with 2 controllers, 2 expansion units and 1GB of cache

Each array contained 36 15K RPM drives

Each array was configured for 6 LUNs, each using 6 disks in a RAID 1+0 configuration

Default stripe size of 32k

Network

Gigabit backbone based on 2 Cisco Catalyst 3500XL switches

Software

Software configuration details are called out specifically in relation to each test. Below is a list of the major versions used for these tests:

Operating System: Solaris™ 10 Update 3 11/06

ZFS: Included in Solaris™ 10 Update 3 11/06

Veritas Storage Foundation: 5.0 GA

Database: Oracle 10gR2 (10.2.0.1)

Benchmark software

One of the more common deployment scenarios for both Veritas Storage Foundation and Solaris ZFS is as a file server. Sharing data between clients and servers on UNIX and Linux is usually done through the Network File System (NFS) protocol. The NFS protocol is the most popular protocol on the market to share data between UNIX and Linux servers; it was first introduced in 1984 by Sun Microsystems and later was made an official standard for transferring files between computers over the network.

The benchmark setup utilized 16 NFS clients to generate requests for the NFS server. The generated load was slowly increased until the response time from the server increased to above 15 milliseconds. The load generator was configured to perform normal file operations (create/delete/read/write/modify/stat etc) on a pre-created dataset. The size of the pre-created dataset grew linearly with the load to simulate larger and larger environments. At predetermined load points, measurements were taken of how many system resources were consumed and how many operations per second the 16 clients were able to sustain continuously.

Configuration

This benchmark was conducted in two different configurations, one using Veritas Storage Foundation for disk access and file system and one using ZFS. No other changes were done between the tests.

Solaris™ 10 was tuned for better NFS performance by increasing the number of NFS daemons to 128. NFS version 3 over UDP was used by both the server and clients.

ZFS configuration

ZFS was configured with a single pool using all 24 LUNs in a RAID-0 configuration. Six file systems were created and shared over NFS to the clients. The default ZFS properties were used with the exception of turning off checksums. Checksums were turned off to minimize any potential impact on CPU usage.

Veritas Storage Foundation configuration

Veritas Storage Foundation was configured with a single disk group, 6 RAID-0 volumes each consisting of 4 LUNs that hosted the 6 file systems. The file systems were shared over NFS to the clients. All of the file systems and volume manager objects were created with default parameters.

Benchmark results

Even though the number of NFS operations per second is the ultimate reference point, it is important that it not be the only consideration when you compare two different products like this. It is also important to consider the amount of system and storage resources that were consumed to reach that number. System resources like consumed CPU cycles and memory has a direct effect on the capital cost for a system. The amount of I/O operations to the storage sub-system has an effect on the capital cost as more I/O's will result in the need for more/faster storage quicker. In a SAN environment, this can have a dramatic effect as the effect increases with each additional host.

Sustained throughput

The figure below shows the number of NFS operations per second (read/write/create/deletes etc) the clients were able to sustain as a function of the response time from the server. The response time increases as the load on the server increases.

Note that the graph for ZFS does not contain as many load points as the one for Veritas Storage Foundation. The reason for this is that ZFS did not successfully scale beyond ~9000 operations/s with this configuration. Beyond the 9000 operations/second load point ZFS fully saturated the CPUs and was unable to scale further. Veritas Storage Foundation required much less CPU and I/O bandwidth to achieve the same number of operations and was able to sustain almost 3 times as many NFS operations per second on the same hardware

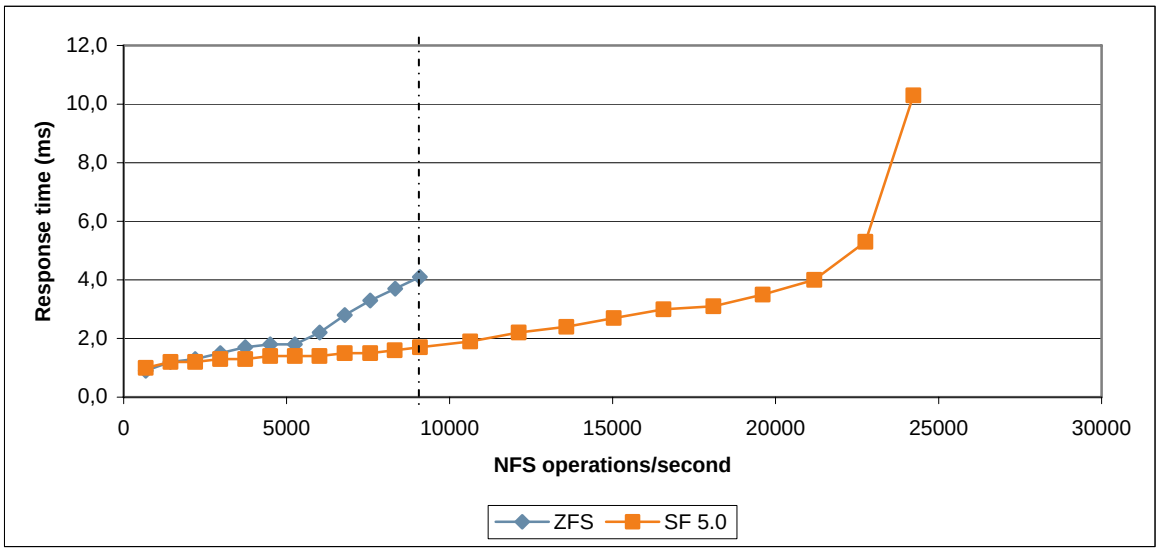

Figure 1 Sustained NFS operations per second

System Resources: CPU usage

The figure below shows the percent of used CPU time reported during the benchmark. As before, the ZFS graph ends prematurely as ZFS was unable to achieve the same result as Veritas Storage Foundation.

Comparing the two graphs at the common load point of ~9000 operations/s clearly illustrates that Veritas Storage Foundation performed the same task using significantly less CPU power.

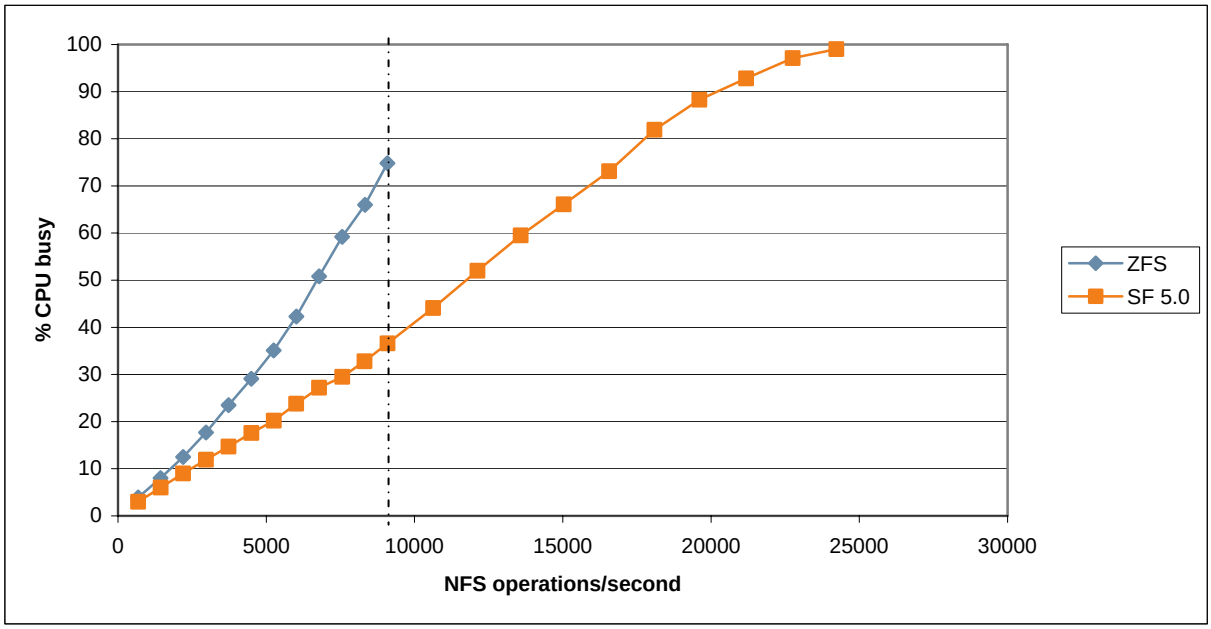

Figure 2 System resources: CPU usage

System Resources: I/O bandwidth

The figure below shows the amount of transferred data per second reported during the benchmark. As before the ZFS graph ends prematurely as ZFS was unable to achieve the same result as Veritas Storage Foundation.

Compare the lines at the common load point of ~9000 operations/s to clearly illustrate that Veritas Storage Foundation performed the same task using significantly less resources. In fact this figure shows that ZFS requires significantly more I/O bandwidth per NFS operation than Veritas Storage Foundation. The I/O statistics were gathered using *iostat* to ensure accurate measurements.

The spike in reads from ZFS at about 5000 operations/s is believed to be the result of ZFS coming under memory pressure together with the aggressive read-ahead caching a lot of unnecessary data.

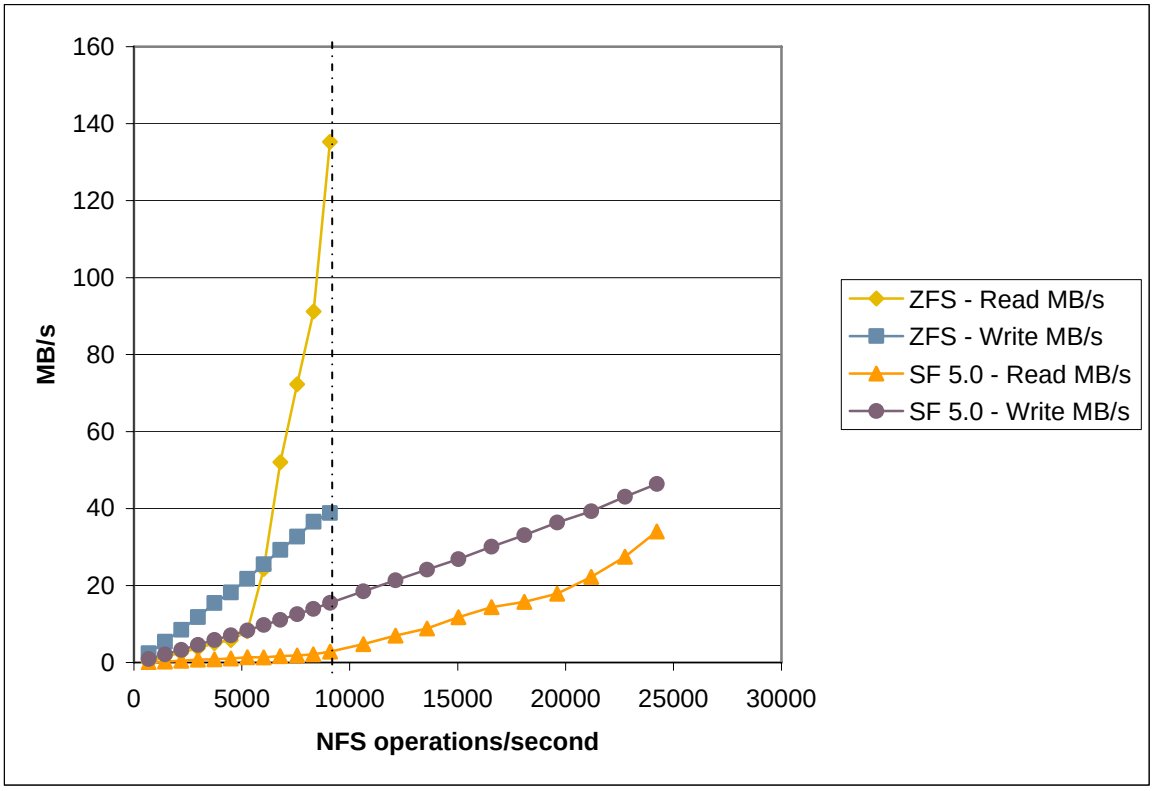

Figure 3 System Resources: I/O bandwidth

System Resources: Memory utilization

The figure below illustrates the memory utilization together with the size of the dataset. The left Y axis and the yellow line represent the dataset size as it grows from ~7GB to 240GB. The right Y-axis and lines represents how much of the total system memory was utilized during these tests.

During each run about 10% of the total dataset is accessed so at the common data point 9000 operations/s approximately 9GB of data was accessed.

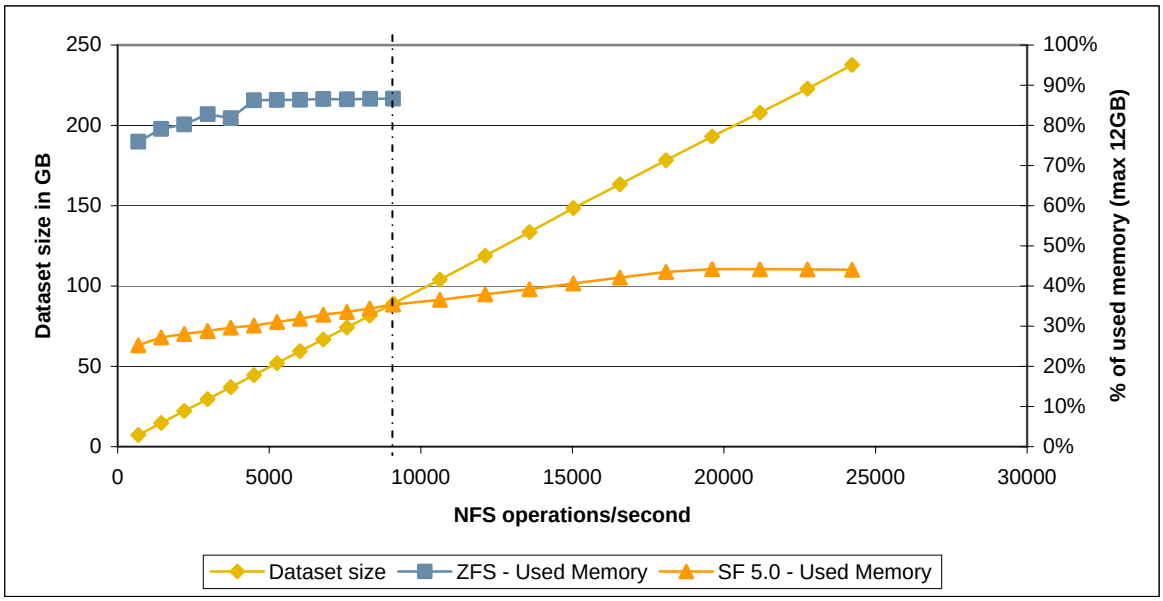

Figure 4 System Resources: Memory utilization

Benchmark software

TPC-C benchmark from the Transaction Processing Performance Council (TPC) is the most commonly used on-line transaction processing (OLTP) benchmark for databases. The benchmark is specifically designed to simulate a complex computing environment where a group of users executes transactions against a database. The benchmark is focuses on five transaction types and the principal activities of an order-entry environment. The simulation includes 1) order entry, 2) delivery, 3) recording payments, 4) checking order status and 5) stock inventory monitoring. While the benchmark is modeled around a wholesale supplier it is not limited to any particular business segment and should be considered a good representation for any industry that sells products or services.

For more detailed information about the database schema and the benchmark suite, please see the TPC website: http://tpc.org

Configuration

Several different software configurations were used to demonstrate the difference between the products in tuned and default configuration. Oracle and operating system parameters were kept identical across these tests.

TPC-C benchmark settings

- 2000 warehouses TPC-C kit, running in batch mode.
- 100 users
- Database size 223GB
- 168 Oracle datafiles, 4 Oracle redo logs

Oracle settings

- Database block size: 8KB
- Shared Global Area (SGA): 6GB (8GB with ODM)

ZFS configuration

To ensure that the best possible results were achieved, ZFS was set up according to the guidelines[1] published by Sun in guides and blogs.

Two configurations were used; ZFS base and ZFS tuned.

In the "ZFS base" case, two separate pools (instead of simply one pool) were created for the database data and the database redo log. As described in the guidelines from Sun this has proven to lower the latency of database log writes and significantly increase the overall performance of ZFS in database environments. The ZFS recordsize for the database pool was set to equal the Oracle database block size (8K), and checksums were turned off.

The "ZFS tuned" configuration was the same as "ZFS base" configuration with additional parameter changes based on the Sun guidelines that we found improved overall throughput.

The storage pool for the data files consisted of 23 LUNs in a RAID-0 and the log pool was configured on a single LUN.

[1] See "Databases and ZFS" in Neelakanth Nadgir's BLOG at http://blogs.sun.com/realneel/entry/zfs_and_databases, "ZFS and OLTP" in Roch Bourbonnais's BLOG at http://blogs.sun.com/roch/entry/zfs_and_oltp and the "ZFS Best Practices Guide" at http://www.solarisinternals.com/wiki/index.php/ZFS_Best_Practices_Guide

Veritas Storage Foundation for Oracle configuration

A typical Storage Foundation deployment scenario was mimicked by creating a single disk group consisting of 24 LUNs. File systems were created on 2 volumes, one 23 LUN RAID-0 stripe for data and a single LUN for the log volume. Both file systems were created with an 8KB block size and an intent log size of 1MB.

Two separate TPC-C runs were made for Storage Foundation. One called “SF 5.0 BIO”, with the file systems configured for normal buffered I/O (Not ideal for database workloads) and one labeled as “SF 5.0 ODM” where the Oracle Disk Manager (ODM) library from Veritas Storage Foundation for Databases was enabled. When the ODM library was used less memory was used by the file system on the server and the SGA size could be increased to 8GB.

Benchmark results

The following sections cover the test results of the TPC-C benchmark for a 2000 warehouse run simulating 100 users.

Transaction throughput

The figure below shows the performance of each configuration relative to the ZFS base configuration. Veritas Storage Foundation has a clear advantage over both of the ZFS configurations. Veritas Storage Foundation with ODM is able to perform more than 5 times the number of transactions that the ZFS base configuration is.

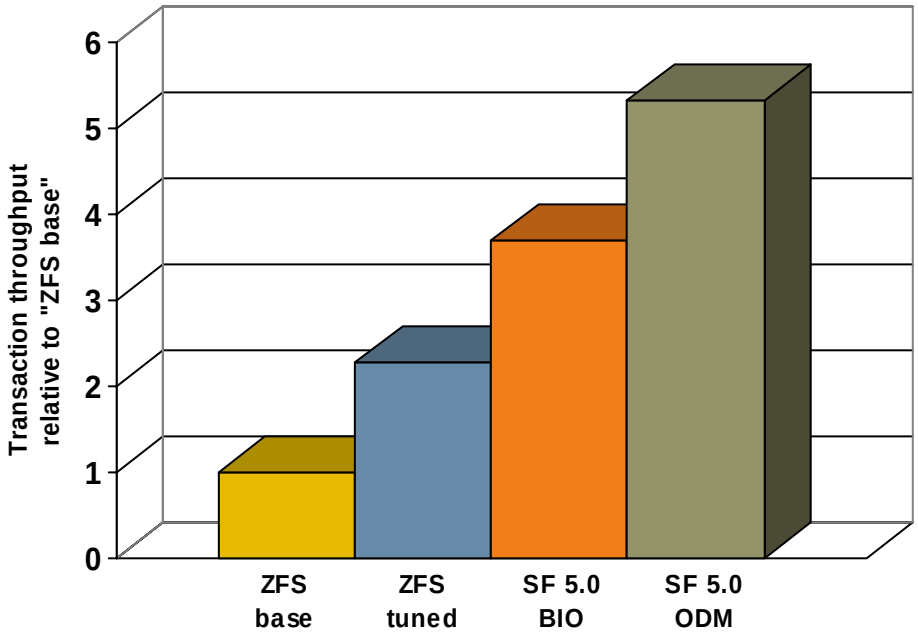

Figure 5 TPC-C throughput at peak load relative to the baseline "ZFS base"

System and Storage resource utilization

Typically, the number of transactions per minute are reported for OLTP benchmarks, but to determine how well the system performed it is important to look at both the I/O load and the CPU utilization. During this benchmark the CPU was fully saturated and the I/O statistics provide valuable information on how the system performed.

The figure below shows the disk read and write transfer rates, as reported by *iostat*, corresponding to the peak throughput reported in Figure 5 for each configuration. Looking at this graph, it is clear that ZFS transfers more bytes than Veritas Storage Foundation. As much as 6 times more I/O bandwidth is required and that is one of the key reasons ZFS cannot deliver the same performance as Veritas Storage Foundation.

The figure below show the transfer rates at the peak load for each configuration, but it does not factor in that the transaction throughput for the Veritas Storage Foundation configurations were significantly higher than for the ZFS configurations. Factoring in the difference in performance, ZFS tuned could be said to require close to 14 times the I/O bandwidth of Veritas Storage Foundation for Databases.

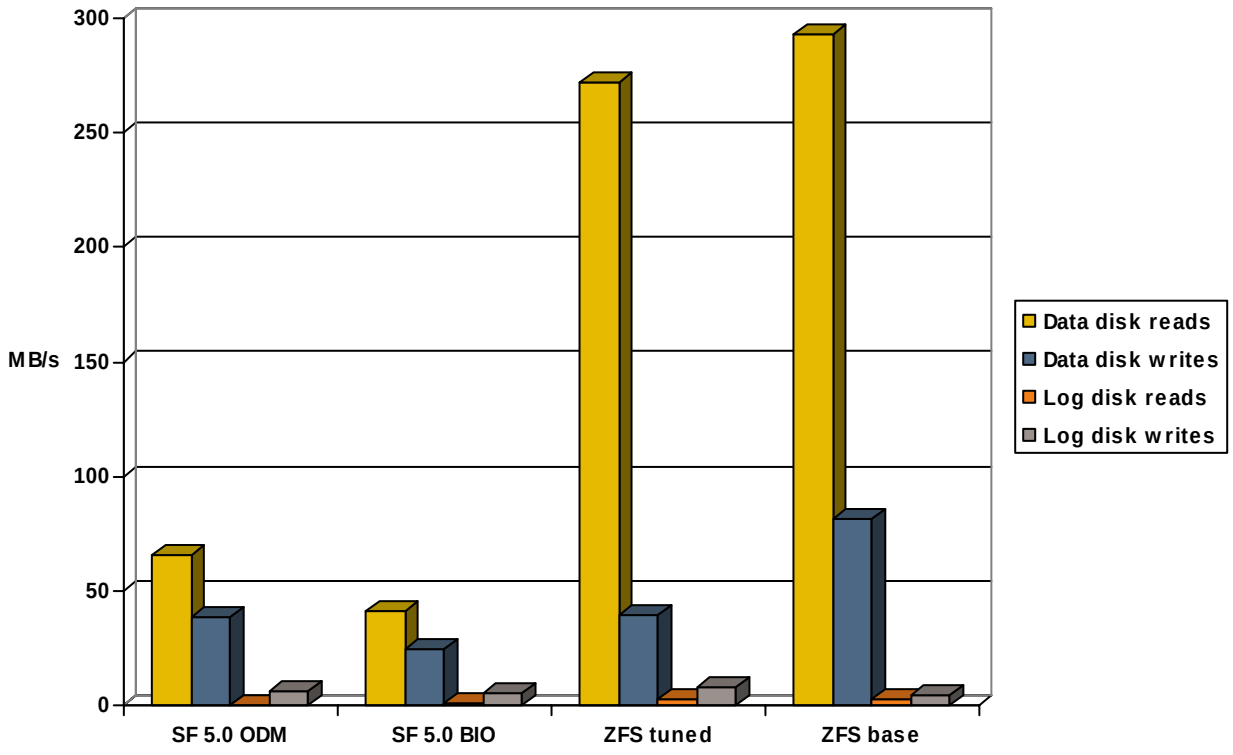

Figure 6 Sustained I/O load at peak load

Conclusion

This paper describes the performance characteristics of both Veritas Storage Foundation and ZFS in a storage area network environment when disk arrays with non-volatile caches are used. In these environments Veritas Storage Foundation clearly outperformed ZFS by a wide margin, both for database workloads and for file serving. ZFS performed well when the load was low and the system had plenty of spare resources. As the load increased, ZFS was not able to scale as well as Veritas Storage Foundation.

In the file serving benchmark, Veritas Storage Foundation was able to sustain 2.7 times as many operations per second. It's important to note that Veritas Storage Foundation also used significantly less system resources to reach that number. At a the same number of operations per second Veritas Storage Foundation used less than an eight of the disk bandwidth, and significantly less CPU system time, than ZFS. The large physical read I/O size and more aggressive read-ahead of ZFS worked well for lower load points, but as the load increased so did the memory pressure. When ZFS was not able to cache the data anymore, the large physical read size together with the read-ahead, hindered rather than helped.

The TPC-C benchmark showed that ZFS requires significant tuning to get better database performance. The tuned ZFS configuration doubled the performance when compared to the ZFS base run. However Veritas Storage Foundation for Databases with ODM outperformed the tuned ZFS configuration by as much as 2.3 times while using less CPU system time and disk bandwidth. The tuned ZFS configuration used more than 5 times as much CPU system time per OLTP transaction as Veritas Storage Foundation with ODM. The large physical read size of ZFS and partial re-writes attribute to a very high read I/O load. Even with the ZFS *recordsize* set to the database block size the average read size was 64kb. This is a result of the *vdev_cache* (softtrack buffer) pre-fetching data. This has been documented by Sun and at least three enhancement requests have been opened according to the entry about “Databases and ZFS” in Neelakanth Nadgir’s blog at http://blogs.sun.com/realneel/entry/zfs_and_databases. Contrary to the results in the blog, the performance of ZFS dropped when the prefetch size was lowered to *recordsize*. The numbers used in this paper, for the tuned ZFS configuration, are from the fastest tested configuration.

In both benchmarks, ZFS was hindered by how synchronous writes are handled. After each synchronous write ZFS initiates an explicit cache flush command to the disk to ensure that the data is physically written to disk. This is required in JBOD environments where the individual disk rarely maintains its cache in case of a power outage, but is not ideal in SAN environments where intelligent arrays with battery backed cache qualifies the cache as stable storage. ZFS also requires two copies of the data to be written to disk, one to the intent log (ZIL) to guarantee that the write made it to disk before returning the write() call, and one to update the actual data on disk when the transaction is flushed to disk.