UNISYS
imagine it. done.

white paper

Linux File Systems

Comparative Performance

insight

This white paper explores comparative Linux file system performance for enterprise Linux deployments. Tailored for technical professionals, it provides information for understanding the most popular Linux file systems currently available.

Table of Contents

Section 1

Introduction

This white paper explores comparative Linux file system performance for enterprise Linux deployments. This is a broad topic and, consequently, it is next to impossible to exhaust even a significant percentage of the information on this subject in a single document. Instead, this document will serve as an additional source for individuals interested in the performance comparison of popular Linux file systems.

All performance testing was done with the open-source file system benchmark tool named iozone. As stated on the iozone site, iozone is useful for performing a broad file system analysis. More information on iozone is available at:

http://www.iozone.org

The file systems tested were ext2, ext3, JFS, XFS and reiser3. These file systems were tested using different I/O schedulers, external journals, file system creation options and file system mount options.

The testing was large in scope with an accumulated total of more than 1,000 hours of testing. The raw data was incorporated into a spreadsheet where simple, yet thorough data analysis was performed. The results spreadsheet is the backbone of the comparison testing. This white paper was created as a means of explaining the results and consequently serves as a summary of the data. The size and format of the spreadsheet make its inclusion as an appendix infeasible. The results are available to the public upon request. Please contact the author of this document if interested in obtaining an electronic copy, as further analysis is encouraged.

The testing, although large in scope, only begins to scratch the surface of the topic of file system performance. Therefore, even though conclusions are drawn in this document based on both the performance metrics and general research, no guarantees can be made and results will vary based on a number of factors unique to each specific environment.

Scope

This white paper is tailored for technical professionals and addresses comparative file system performance. It is intended to provide the information for understanding the most popular file system offerings available for enterprise Linux deployments. Given the focus on the enterprise, only Red Hat Enterprise Linux 4 and SUSE Linux Enterprise Server 9 distributions are covered. Only 32-bit (ia32) versions are addressed in order to further reign in the scope.

Network file systems, removable media file systems, and in general, non-journaling file systems (with the exception of ext2) will not be covered. The scope will instead focus on only the most popular journaling file systems for the 2.6 kernel. Because of the limitations of the 2.4 kernel and the usage of the 2.6 kernel in current Red Hat Enterprise Linux and SUSE Linux Enterprise Server releases, all file system specifications and limitations are based on the 2.6 kernel.

File system tuning is addressed and tested as much as possible, with some tuning omitted. For example, elevator tuning, now done through the */sys* file system, is not covered because the quantity of test iterations would have increased by a significant order of magnitude.

File system features such as Access Control Lists (ACLs), quotes, etc., are catalogued in the feature set description of the file systems. However, that is where the comparison ends. There is no testing, whether functional or performance related, done to compare these features between file systems. In order to focus on the performance aspect, the scope of this document omits usage of these features.

Audience

Administrators, technical professionals and those with an interest in Linux file systems will find the information contained within useful. File system developers may also find useful content, although the level of depth is not indicative of a development-focused document.

Section 2

Executive Summary

From a performance standpoint, ext2 is almost always the fastest file system with very few exceptions. Thanks to an efficient design and low overhead stemming from its journal-free design model, ext2 performance is excellent in all scenarios.

In enterprise-class deployments, file system integrity goes hand in hand with performance. Therefore, it is in the best interest of most administrators to choose a journaling file system. Ext3 offers journaling on top of the well-established ext2 file system, leverages the extensive history of ext2 bug-fixes, and includes a versatile collection of user space tools. Additionally, it employs vendor support by both Red Hat and SUSE for their enterprise Linux distributions. However, since ext3's journaling is an extension to the original ext2 file system, there is merit in looking at file systems which were designed from the ground-up to include journaling in addition to ext3.

JFS, XFS and the Reiser family of file systems were designed from their inception to be high-performing, journaling file systems and each has its own unique advantages and caveats which are detailed throughput this white paper. When compared against each other, JFS and XFS consistently scored impressive throughput numbers rivaling their journaling competitors. Refer to the body of this white paper for a variety of throughput metrics and further information on the individual file systems and how they stack up against each other using the open-source tool iozone as the benchmark.

Section 3

File Systems Overview

As a precursor to the comparison data and for the sake of completeness, some key file system concepts are included in this section.

File systems are software components that support the I/O infrastructure of operating systems. A file system serves as the mechanism that allows multiple files and directories of data to be stored on a single partition of fixed or removable media. The file system provides the facilities for accessing, organizing and further managing the stored data.

Data can be written directly to a disk partition in raw format, circumventing both the file system and the buffer cache. This method can be very efficient because the overhead of an additional software layer is removed. However, without a file system to generate and manage metadata, a disk partition is limited to a single file. Raw partitions cannot be the targets of utilities such as *tar, cpio, dump,* etc. – all of which work with the file system. Traditionally, raw partitions are used when performance is the highest priority or when the application itself will provide file system-like functionality and/or manage its own cache.

For Linux file systems, files within a file system are represented by *inodes*. Inodes are file system objects that contain metadata, as well as the list of data blocks occupied by the file. The inode is the first point of access when the file system attempts to access a file.

Journaling

A file system journal is a log of transactions executed against the file system. Usually, the recorded transactions are limited strictly to modifications of the inodes, or the metadata. The journal is updated prior to the file system operation and then updated again to reflect the successful completion of the file operation. Using this method, a file system can replay its journal in the event of a system failure to systematically determine which operations completed successfully and which did not. This will ensure that the file system structure remains intact in the event of a system failure. However, this does not guarantee that the user data itself will reflect the most recent changes.

The journal does impose additional overhead to the file system. But because the journal is written serially and only records metadata changes, the journal is less resource intensive than the actual file operations, which are a collection of random and serial reads, writes and transfers. It is the extra overhead, and consequently the reduced performance, that have led some administrators to suffer reduced performance after migrating their ext2-based file systems to journal-based file systems.

File System Blocks

File systems typically rely on blocks as the smallest unit of storage. The file system block size is a multiple of the media sector size—512 bytes on standard hard drives. Linux does not support a block size greater than page size, which is 4kb on Intel-based, 32-bit systems. Therefore, even though most modern file systems support greater block sizes, they have a practical limitation of 4kb.

File systems are built with their own internal limitations, but in the end are inherently limited by the constraints of the Linux kernel. The following table shows the kernel limitations for individual files and file systems:

	2.4.x	2.6.x
IA32 Max File Size	2TB	2TB
IA32 Max File System Size	2TB	16TB

Section 4

Ext2

In January 1993, the alpha version of the Second Extended File System (ext2) was released. Ext2 is based on the first Extended File System (ext), but contains many improvements and has the ability to accommodate future modifications more easily than its predecessor. It was developed to fix problems and limitations of ext, as well as provide increased performance. A comparison between the original ext file system and the ext2 file system specifications are shown in the chart below:

	Ext	Ext2
Max File Size	2GB	2TB
Max File System Size	2GB	16TB
Max File Name	255 characters	255 characters
Extensible	No	Yes
Three Time Stamps	No	Yes
Dynamic Inode Allocation	No	No
Removable Media Support	Yes	Yes

Ext2 is arguably the most widely used file system in the Linux community. Its strong Unix roots, long history of stability and excellent performance by even today's standards are contributing factors to its popularity. In general a file system's reliability, in reference to the reliability of the code itself, is directly related to its active lifespan. After so many years of regular usage by an established base, the code has seen enough real-world testing and bug fixes to establish a reliable code base.

File system state is maintained and designated as either clean when it is not mounted, not clean when mounted in read/write mode, or erroneous when an inconsistency is detected. The file system state is critical because there is a journal to guarantee the consistency of the file system. Consequently, an unclean or erroneous file system must undergo a complete check by the file system utility *fsck*. Unplanned system shutdowns or system failures will often result in unclean and possibly erroneous ext2 file systems. Despite many optimizations to decrease fsck times, they can be lengthy on large, densely populated disks.

Features

- **Unique File Attributes (Linux specific)**

Immutable – Read-only, cannot be written to, deleted, or renamed.

Append only – All writes are appended to the end of file. Cannot be deleted.

Undeletable – Cannot be deleted with this flag set.

- **Secure File Deletion**

Users can request secure file deletion. Ext2 will write random data to the data blocks previously occupied by the file in question. Therefore, obtaining raw access to the disk will not yield direct access to the previously deleted file. This does not guarantee that data recovery tools will not be able to obtain the original data, but it provides an additional level of security over a standard delete, which does not guarantee the original data blocks will be immediately overwritten.

- **Excellent Community Support**

There are numerous user-space tools available for administration of ext2 file systems, which are readily available and even bundled with popular distributions. Common tools include *fsck, debugfs, dump* and *restore* that provide maintenance and administrative functionality to ext2 file systems. Advanced tools such as *ext2resize* allow growing and shrinking of the file system; *ext2online* (http://ext2resize.sourceforge.net/online.html) allows growing of a mounted ext2 file system and *ore2salvage* enables (http://e2salvage.sourceforge.net/) recovery of damaged data. There are even Windows drivers (http://sourceforge.net/projects/ext2fsd) that allow read/write functionality of an ext2 file system.

Limitations

- No journaling.
- Possibility of premature inode depletion.

Tuning

Ext2 is addressed in this document only as a high-water mark for the journaling file systems. Therefore, tuning parameters were not tested aside from running with the different I/O schedulers.

Section 5

Ext3

Ext3 is, in effect, an enhancement of the ext2 code base with the addition of journaling functionality. The two file systems are so similar that an ext2 file system is both forward and backward compatible with ext3 and all the inherent limitations of ext2 apply to ext3. An ext3 file system can be mounted as an ext2 file system and an ext2 file system can be converted to an ext3 file system without data loss. Ext2-based file system utilities, such as dump and restore, are compatible with ext3-based file systems. Because of its roots in ext2, ext3 has been widely accepted as a stable journaling file system and has been included in the vanilla Linux kernel since 2.4.16.

Ext3 file system specifications are listed in the chart below:

Max File Size	2TB
Max File System Size	8TB
Max File Name	255 characters
Extensible	Yes
Three Time Stamps	Yes
Dynamic Inode Allocation	No
Removable Media Support	Without journal (ext2)

As a result of the similar roots, ext2 file systems can be converted to ext3. *E2fsprogs*, a collection of utilities for ext2 and ext3 file systems, includes the *tune2fs* utility. Tune2fs adjusts tunable file system parameters on ext2 file systems and can also add a journal to ext2-based file systems, in turn converting them to ext3. Refer to the following link for the exact procedure:

http://www.redhat.com/docs/manuals/linux/RHL-7.3-Manual/ref-guide/s1-filesystem-ext3-convert.html

Converting root file systems from ext2 to ext3 requires booting off of an initial ram disk (initrd) image prior to running tune2fs. For more information on this procedure, please refer to the following link:

http://www.redhat.com/docs/manuals/linux/RHL-7.3-Manual/ref-guide/s1-filesystem-ext3-convert.html

Features

- **Optimized Head Movement**

Although usually slower than ext2, ext3 in some instances will outperform ext2 because the journal is designed to optimize hard disk head movement.

- **Journal Versatility**

The ext3 journaling capabilities are versatile. For starters, an ext3 file system can be mounted as ext2, foregoing the journaling when performance is of the utmost importance. With journaling enabled, there are 3 options that allow an administrator to tweak the journal based on the application and the level of performance desired. The ext3 journal settings are covered in the ext3 tuning section below.

- **Flexible Migration**

Ext2 administrators can move to ext3 by adding a journal to the file system without reformatting using *tune2fs*. In the reverse case, the ext3 journal can be removed with *tune2fs*, or the ext3 file system can simply be mounted as an ext2 file system.

Limitations

- Possibility of premature inode depletion.
- Held to the same restrictions as the aging ext2 file system.

Tuning

Journal Configuration

The ext3 journal is usually the source of performance disadvantages observed when comparing the often faster ext2 to ext3. The journal mode is configurable, and is set at mount time under the options in */etc/fstab*. The three modes are as follows:

- **data=ordered**

This is the default ext3 journal setting and guarantees data consistency. Orders write by updating data blocks then updating the metadata. This way the metadata is always in sync with the user data.

- **data=writeback**

This setting provides the best performance for ext3, but at a reduced data integrity level. Commits file system changes in FIFO order and data blocks can be updated after the respective metadata update—a more aggressive write strategy than the data=ordered journal option. File system integrity is still guaranteed, but user data may not reflect the most recent information in the metadata after a system crash.

- **data=journal**

This setting uses a larger journal that records changes to the file system metadata and the user data, resulting in a longer journal replay after a system crash. In some cases this setting can provide increased performance for database applications, but generally results in reduced performance from the additional overhead of a more comprehensive journal.

External Journal

By default, the ext3 journal resides on the same block device as the rest of the file system. An external journal, as the name implies, resides on a block device external to the rest of the file system. By allocating a dedicated spindle to the task of journaling, head movement on the data drive is optimized. If the drive is heavily utilized, increased throughput should result.

Caveats

• An external journal must be created with the same block size as the target file system.

• Referencing the *mke2fs* man page, the file system journal must be a minimum of 1024 file system blocks and a maximum of 102,400 file system blocks. For a 100GB file system based on the maximum allowable block size of 4k, the journal size must be at least 4MB(1024 * 4k) and no greater than 400 MB (102,400 * 4k).

Creating an External EXT3 Journal

1. Create the journal on a dedicated partition:
   mke2fs -O journal_dev /dev/<external_journal_partition>

2. Create an ext3 file system on /dev/sdXY that uses the external journal:
   mke2fs -J device=/dev/<journal_partition> /dev/sdXY

3. Mount the partition (must specify file system type):
   mount /dev/sdXY /mnt/sdXY -t ext3

Section 6

Reiser3

Reiser3 (aka reiserfs) development was originally led by Hans Reiser of Namesys (http://www.namesys.com). It was designed from the ground up as a space efficient, multi-purpose, journaling file system using balanced trees (B-trees). B-trees are designed for slower storage mediums (hard disks for example) because they are capable of referencing large amounts of data while minimizing I/O overhead. B-tree usage is common in modern file systems. More can be read on the functionality of B-trees at: http://www.bluerwhite.org/btree.

Reiser3 was designed to handle large directories better than other file systems while exceeding the performance of ext2 and offering guaranteed file system integrity via a journal. Subsequently, reiser3 was the first journaling file system to be included in the vanilla Linux kernel—fully integrated as of 2.4.1 and is the most widely used journaling file system for Linux to date. Reiser3 is the preferred and default file system for SUSE Linux Enterprise Server 9.

The 3.6 release of reiser3 offers the following specifications:

Max File Size	8TB
Max File System Size	16TB
Max Files	$2^{32} - 3$ (4,294,976,293)
Max File Name	255 characters
Extensible	Yes
Dynamic Inode Allocation	Yes
Removable Media Support	No

Features

- **Efficiency**

Reiser3 does not use traditional block space allocation where files are stored in as many fixed-size blocks as necessary to contain them. Typically, a file system of 4k blocks would require 2 blocks to store a file of size 4097 (4kb + 1 byte). Instead, reiser3 relies on B-trees instead of inodes and stores all file system structures with the B-trees. This technique allows reiser3 to avoid the efficiency pitfalls of block-based allocation by storing multiple files within a single block. On systems with many small files, such as a typical user workstation, reiser3 can yield significant space benefits. This efficiency is compounded by metadata being stored next to the file on the hard disk platter in order to reduce head movement when accessing small files.

- Journal Versatility

Like ext3, the reiser3 journaling capabilities are versatile. Three options allow an administrator to tweak the journal based on the application and the level of performance desired. The reiser3 journal settings are covered in the resier3 tuning section below.

- Efficiency

Namesys states that reiser3 is a general purpose file system and is extremely efficient in a variety of situations, including numerous directories containing numerous small files. The versatility of reiser3 makes it a good choice for a general purpose workstation.

Limitations

- No removable media support.
- No defragmentation utility, file system must be dumped and restored to defragment.

Tuning

Journal Configuration

The reiser3 file system, similar to ext3, allows multiple journaling modes set on file system mount. In fact, the journal settings are identical between the two file systems. An explanation of the journal settings is included in the man page for the mount command and is also detailed as follows:

- **data=ordered**

This is the default reiser3 journal setting and guarantees data consistency. Orders writes by updating data blocks then updating the metadata. This way the metadata is always in sync with the user data.

- **data=writeback**

This setting provides the best performance for reiser3, but at a reduced data integrity level. Commits file system changes in FIFO order and data blocks can be updated after the respective metadata update—a more aggressive write strategy than the data=ordered journal option. File system integrity is still guaranteed, but user data may not reflect the most recent information in the metadata after a system crash.

- **data=journal**

This setting uses a larger journal that records changes to the file system metadata and the user data, resulting in a longer journal replay after a system crash. In some cases this setting can provide increased performance for database applications, but generally results in reduced performance from the additional overhead of a more comprehensive journal.

External Journal

Reiser3 does not offer external journaling capabilities.

Section 7

Reiser4

Reiser4, the next generation of the reiser line of file systems, was released in late 2004. Because of the young age of reiser4, Namesys states that, “It will also be some substantial time before v4 is as stable as v3.” Instead of enhancing the existing reiser3 code base, reiser4 is a complete redesign of the file system. As a testament to its stability, Namesys touts experienced development and testing values that have attributed to a high degree of stability in reiser4’s current form.

Reiser4 support is available as an option in the akpm tree of the Linux kernel starting with version 2.6.8.1-mm2.

Therefore, usage of reiser4 requires usage of an unsupported kernel, regardless of vendor, rendering it infeasible in its current state for enterprise deployment.

For instructions on preparing a system for reiser4, including patching the kernel, compiling the reiser4 tools, and creating the file system, refer to the Namesys reiser4 installation guide:

http://www.namesys.com/install_v4.html

Reiser4 was not subjected to iozone testing. This document is focused on the enterprise and usage of an unsupported kernel, which is antithetical to maintaining a reliable datacenter. A subsequent version of this paper will include ext2, ext3, JFS, reiser3, reiser4 and XFS on a newer Linux kernel when reiser4 is picked up by either Red Hat Enterprise Linux and/or SUSE Linux Enterprise Server.

The specifications of reiser4 are listed below:

Max File Size	*unknown*
Max File System Size	*unknown*
Max File Name	> 1024 characters
Removable Media Support	Yes
Dynamic Inode Allocation	Yes

Features

- **Dancing Trees**

An improvement over the balancing trees in reiser3, dancing trees are touted by Namesys as both faster and more efficient. They work by merging insufficiently full nodes when memory shortage triggers a flush to disk and when a transaction closure triggers a flush to disk. Because of the dancing tree model, reiser4 is even more space efficient than reiser3.

- **Atomicity**

Reiser4 is an atomic file system; disk operations happen completely or do not happen at all. This can prevent consistency issues that sometimes even journaling cannot prevent in the case of in-progress operations.

- **Plug-in Infrastructure**

Reiser4 not only supports plug-ins, but the architecture itself is plug-in based to allow a very modular approach to modifying the file system. This allows custom security modules and changes in functionality to be added or changed, giving the file system the potential for a long life span.

Limitations

- Limited out of the box journal configuration.
- No journal flexibility, no external journal feature.
- Not currently supported by either Red Hat Enterprise Linux 4 or SUSE Linux Enterprise Server 9.

Tuning

Journal Configuration

Reiser4 does not offer configurable journal settings at mount like reiser3 or ext3.

External Journal

Reiser4 does not offer external journaling capabilities out of the box.

Section 8

JFS

In February 2000, IBM open sourced their enterprise server file system known as Journaling File System (JFS) to the Linux community to further the penetration of Linux into the enterprise datacenter. Currently, four IBM developers are in charge of maintaining the code and integrating patches, as well as other changes from the open source community. The developer contact information is listed here:

Barry Arndt	barndt@us.ibm.com
Dave Blaschke	blaschke@us.ibm.com
Dave Kleikamp	shaggy@austin.ibm.com
Steve Best	sbest@us.ibm.com

JFS is available as part of the vanilla kernel in the 2.5.6 release and up, and is identified as JFS file system support (CONFIG_JFS_FS) during the kernel build process. JFS can be built into the kernel, or as a module.

JFS was designed by IBM for high-throughput, heavy-load servers that require a robust file system and the availability of full journaling. It was originally open sourced because there were no journal-based file systems for the Linux operating system at the time.

JFS and Enhanced JFS differ in that JFS is optimized for 32-bit kernels, where as Enhanced JFS is optimized for 64-bit kernels. The standard JFS will run on both 32-bit and 64-bit hardware, but the Enhanced JFS takes advantage of the increased address space to offer degrees of magnitude increase in the standard file system specifications. Aside from the increased scaling, both versions of JFS share the same libraries, commands, utilities, and header files.

JFS is a journaling file system, and as with most journaling file systems, it only logs changes to metadata. Therefore, JFS will not recover file data to a consistent state after a system failure and data loss may occur in some situations. Refer to the JFS Log whitepaper referenced in this document for more information. Unlike ext3 or the reiser line of file systems, JFS cannot be configured to log user data.

The specifications of JFS are listed below:

Max File Size	8EB
Max File System Size	4PB
Max File Name	255 characters
Removable Media Support	Must be >= 4096 blocks (16MB)
Dynamic Inode Allocation	Yes

Features

- **Mainframe Roots**

Ported from the IBM AIX JFS implementation, JFS has a history of stability in high-end IBM mainframes.

- **Advanced Directory Organization**

JFS has efficient algorithms for directory organization. Small directories of 8 or less entries are stored within the directory inode. Larger directories are sorted and stored in B+ trees, allowing faster lookup than traditional unsorted directories.

- **Maintenance Friendly**

JFS has several tools available to ease maintenance and provide as much functionality as possible without taking the file system offline. An online defragmentation utility is available, as well as online snapshotting of the file system and an online file system growth utility.

Limitations

- No quota support.
- Limited removable media support.
- Cannot journal user data.

Tuning

External Journaling

To isolate the file system journal on a separate, dedicated disk in order to optimize head movement and increase throughput on the data disk, external journaling can be used. According to the man pages for *jfs_utils*, there are several syntax formats for creating and attaching an external journal. Interestingly enough, one syntax works, while the other commands all producing the error:

> The following disk did not finish formatting.

To create an external journal on */dev/<external_dev>*, create a JFS file system on */dev/<jfs_partition>* and attach the journal to the file system with the following command:

> mkfs.jfs –j /dev/<external_dev> /dev/<JFS_partition>

The system will warn that all data on both */dev/<external_dev>* and */dev/<JFS_partition>* will be lost. After user confirmation, mkfs will then automatically create a working journal 0.4 times the partition size, rounding up to the nearest megabyte. The maximum journal size is capped at 128MB regardless of partition size.

No additional mount parameters need to be specified to mount a JFS file system with an external journal.

Section 9

XFS

On March 30, 2000, SGI, not to be outdone by IBM, released the 64-bit XFS file system code to the open source community in alpha stage. At the time, the beta release of XFS was still 6 months away. XFS was destined to replace SGI's aging EFS file system, which was reaching its limits on large files and large file systems. It was not until May 1, 2001–13 months later–that the 1.0 version of XFS was released. On September 16, 2002, XFS was merged into Linus's 2.5 kernel tree, and was available in the vanilla Linux kernel since the 2.6.0 release. It is set with the XFS file system support kernel option (CONFIG_XFS_FS).

XFS is designed to be an efficient and high-performing file system developed to manage extremely large file systems and stream bandwidth-intensive content in real time. XFS is SGI's standard file system, shipping on everything from uniprocessor workstations to their SMP server line. Despite lack of widespread XFS adoption, it is in fact the oldest journaling file system for Linux.

The specifications of XFS are listed below:

Max File Size	9EB (SUSE: 8EB)
Max File System Size	18EB (SUSE: 8EB)
Max File Name	255 characters
Removable Media Support	Must be >= 4096 blocks (16MB)
Dynamic Inode Allocation	Yes

Features

Direct I/O

XFS has the ability to write directory to user space I/O, completely bypassing the file system buffer cache. This is beneficial to certain database applications that utilize their own cache manager. As an example, Oracle typically recommends raw disk partitions not just for lack of file system overhead, but also to avoid the file system buffer cache.

Delayed Allocation

XFS is the only file system mentioned in this document that uses delayed allocation. When a file write is performed, total free disk space is decremented, but the actual write does not occur; the data is only written out to disk when required. By doing this, XFS can combine writes to increase sequential accesses, reduce total writes to disk by making changes in main memory when possible, and eliminate writes that become obsoleted before they are finally committed to disk. This technique has the added benefit of reduced disk fragmentation.

Scalability

XFS uses allocation groups which act as independent file systems within an XFS file system. Multiple allocation groups can be accessed in parallel by the kernel to provide efficient concurrent access on SMP systems. The allocation group count is specified on file system creation.[1]

Guaranteed Rate I/O (GRIO)

Guaranteed Rate I/O (GRIO) is a unique XFS feature that allows applications to reserve bandwidth from the file system. A process requests to receive data from an XFS file system at a specific rate for a specific length of time. As long as GRIO request is within the capabilities of the system, the request will be met. This can be useful when streaming real-time audio or video, or when interacting with an information source that is only available for brief periods of time.

Two XFS utilities allow GRIO functionality:

cfg - Scans the hardware available on the system and creates a file, */etc/grio_config*, that describes the rates that can be guaranteed on each I/O device

ggd - Manages the I/O-rate guarantees that have been granted to processes on the system

[1] XFS maximum block size is 64k and it will allow you to create a file system with this block size. However, because the Linux page size of 4kb dictates the maximum block size, the 64kb device is un-mountable. The error received is:

linux kernel: XFS: Attempted to mount file system with blocksize 65536 bytes
linux kernel: XFS: Only page-sized (4096) or less blocksizes currently work

Limitations

- No undelete functionality.
- No disk quota support.
- Cannot journal user data.
- Limited removable media support.

Tuning

External Journal

The file system journal can be externalized to optimize head movement and increase throughput on the data drive. To create a file system on the first partition on the first SCSI disk with an external journal located on the first partition on the second SCSI disk, use:

mkfs.xfs -l logdev=/dev/<external_dev> /dev/<xfs_partition>

ex. – mkfs.xfs –l logdev=/dev/sdc1 /dev/sdb1

If size is not specified, mkfs.xfs will determine a log size based on the file system. However, if the file system is too large, the mkfs command will not execute, returning with error message "log size xxxxxx blocks too large, maximum size is 65536 blocks." Note that even though the error message states that a log size up to 64k blocks is acceptable, a subsequent mkfs command specifying a 64k block log returns this:

log size 268435456 bytes [64k blocks] too large, maximum size is 134217728 bytes [32k blocks].

Further supporting the latter error message is the fact that the attempted creation of a 63k block journal (one less than 64k), as well as 33k block journal (one more than 32k) yield the same error. Therefore, an external journal of a maximum of 32k blocks [size=32768b] must be specified.

Note: An externally journaled XFS file system will not mount unless the mount command references the external log. This differs from most file systems, where external journal location can be determined by the data disk. Usage of the following syntax when mounting the file system is required:

mount –t XFS –o logdev=/dev/<ext_journal> /dev/<xfs_partition> /<mount_point>

While cumbersome, mandatory specification of an external XFS journal can be beneficial from an administrative standpoint because listing the mounted file systems via the mount command immediately informs the user of the external journal configuration. An example of how the external journal mount option is displayed when issuing the mount command is displayed here:

type xfs (rw,logdev=/dev/<ext_journal>)

Data Section

The data section of an XFS file system can be tuned at file system creation. Notable performance chances include altering the size and/or number of allocation groups. These two values are directly related, and only one needs to be modified. Increasing the number of allocation groups allows increased parallelism, additional resource consumption, and implied throughput increases. The default number of allocation groups is 8, unless the file system is greater than 8GB. After which the number of allocation groups periodically doubles – to 16, 32, and so on. The syntax for creating an XFS file system with a specific allocation group count:

mkfs.xfs –d agcount=<agcount>

The sector size of an XFS file system can be specified, ranging from 512 bytes to 32k, but less than or equal to the file system block size. Since the file system block size is limited to 4k, the sector size defaulted to the maximum value of 4k. Smaller values were not tested.

Section 10

File System Support

Red Hat

Red Hat Enterprise Linux 4 only allows usage of the ext3 file system during the installation process. Red Hat states,"In our judgment, ext3 currently has the best fit for our customers' requirements. We will continue to evaluate other file systems for inclusion in future versions of Red Hat Linux."

Despite Red Hat's limited file system selection, they have chosen a journaling file system with the flexible journaling modes and external journal support. Ext3 is built on the time-tested ext2 code base and has also undergone several Red Hat-specific improvements to further increase its reliability and performance.

The Red Hat-specific patches applied to the vanilla ext3 code base are listed here:

linux-2.6.5-ext3-online-resize.patch

linux-2.6.5-ext3-reservations.patch

linux-2.6.8-ext3-reservations-update.patch

linux-2.6.9-ext3-cleanup-abort.patch

linux-2.6.9-ext3-file-limit.patch

linux-2.6.9-ext3-handle-bitmapdel.patch

linux-2.6.9-ext3-handle-double-revoke.patch

linux-2.6.9-ext3-mbcache.patch

linux-2.6.9-ext3-release-race.patch

linux-2.6.9-ext3-umount-leak.patch

SUSE

SUSE Linux Enterprise Server 9 supports ext2, ext3, reiser3, XFS and JFS. These file systems are all available as options during the disk partitioning portion of the installation wizard. SUSE provides excellent file system flexibility and fully supports usage of all file systems mentioned in this document, with the exclusion of reiser4 at the current time.

Section 11

File System Tuning

For this document, all file system tuning will be limited to parameters used during file system creation, I/O scheduler selection and mount options. Options that disable journaling on journal-based file systems such as the JFS nointegrity mount option or the reiser3 nolog mount option were not tested. The focus of this document is on the enterprise. Because file system integrity and fast file system recovery are critical considerations in enterprise deployments, there is little interest in comparing file systems with journaling disabled.

However, some ext2 performance testing was done; the results were used as a reference point for comparing the performance differences between a stable, well-performing, non-journaling file system and the current generation of high-performance, journaling file systems.

Also note that not all performance tuning possibilities were exhausted. Tuning parameters on the cutting block were those deemed counterintuitive for performance gains, inapplicable for enterprise environments or features that would not be utilized by iozone. Directory search hashes, security features, and advanced application features were among the tuning parameters omitted from testing.

On 32-bit hardware, linux file systems have a maximum effective block size of 4kb. Most allow usage of smaller block sizes with the minimum size being equal to the hard disk sector size of 512 bytes. Block sizes smaller than 4k were not tested, as large block sizes usually provide better performance at the expense of space efficiency when storing small files.

It is important to understand that the 4k block size limitation is not because of the file system itself; in fact many file systems support larger block sizes such as XFS's 64kb maximum. Instead, this is an ia32 Linux operating system limitation.

Noatime

Most modern file systems store and maintain three separate time stamps per file. These are created, last modified and last accessed. The last accessed time stamp, by virtue of its function, may have to be updated quite frequently. For example, listing the contents of a directory will cause all the containing files to receive new last accessed time stamps, amassing additional writes and adding to the file system overhead.

The file system mount option noatime can be used to disable updating the last accessed time stamp resulting in reduced file system overhead. Usage of the noatime flag is tested. Systems with fewer large files, such as database servers, will most likely not experience noticeable improvement with the usage of the noatime mount option.

IO Schedulers

The vanilla Linux kernel bundles four different I/O schedulers as part of the package. These schedulers are as follows:

anticipatory (as)

completely fair queuing (cfq)

deadline (deadline)

noop (noop)

The vanilla 2.6.x kernel sets the default scheduler for all block devices as *anticipatory*. Both SUSE Linux Enterprise Server 9 and Red Hat Enterprise Linux 4 distributions use the completely fair *queuing* scheduler as the default. The I/O scheduler is changed for all block devices by modifying the elevator kernel boot parameter, as per the following syntax used in */boot/grub/menu.lst:*

elevator=as | cfq | deadline | noop

The I/O scheduler can also be set on a per block device basis through echoing the scheduler designation in */sys/block/<sdx>/queue/scheduler.*

Free Space

Many file systems run at peak efficiency when the file system has ample free space. In general, a file system should have no less than 15 – 20% free space in order to run at its maximum performance level. This is a caveat of the file system itself and the recommended amount of free space varies between file systems. Unfortunately, there are few specifics available on this metric in relation to the individual file systems, so a general conservation principle should be applied where no more than 80 – 85% of a disk capacity is consumed.

Even greater performance can be achieved by implementing an aggressive policy restricting the space utilization of hard disks to the lowest percentages possible. Hard disks store data on the outer edges of hard disk platters first. Since hard disk platters spin at a constant angular velocity, the linear velocity at the outer edge of the platter is significantly faster than at the inner edges of the platter, resulting in faster seek times and faster transfer rates. While not necessarily a cost-effective performance solution, this should be taken into consideration for high-performance deployments.

Section 12

Testing

At the core of the test environment resides a Unisys ES7000/540 server. The ES7000/540 server is a 32-processor, Intel-based, 32-bit server built on mainframe architecture. The server was partitioned into an 8-processor system for the purpose of testing. More information about the ES7000/540 server can be found at:

http://www.unisys.com/products/es7000__linux/es7000__32__bit__servers.htm

All testing was done with SUSE Linux Enterprise Server 9 RTM. Duplicating the full gamut of testing on an identical Red Hat Enterprise Linux 4 environment would have been ideal. Unfortunately, in order to keep the testing scope to manageable proportions, all testing was done on SUSE Linux Enterprise Server 9 RTM.

The high-level specifics of the server configuration:

1x Unisys ES7000/540 G3 V1.1, Single Cell[2]

8x Intel Xeon MP 3.0GHz 4.0MB L3 cache, HT disabled

1x QLogic 2310F FC-HBA

64.0MB L4 cache[3]

8.0GB Main Memory[4]

The SAN configuration:

1x EMC CX600, dual SP equipped, single SP used (SPA)

4022MB SPA Cache

(1) 16-disk RAID-0 LUN (I/O testing target)

Seagate ST336753 36GB 15K rpm Fibre-Channel disks

(1) 2-disk RAID-1 LUN (external journal target, where applicable)

Seagate ST336753 36GB 15K rpm Fibre-Channel disks

1x Brocade Silkworm 3800 Fibre Channel Switch

2 The Unisys ES7000/540 server is an Intel-based SMP server configurable with up to 32 Intel Xeon MP processors. The ES7000 Family of Servers is a high-end server line designed with performance and redundancy in mind. The ES7000/540 is comprised of 4 "cells," which are physical and logical hardware units containing processors and memory that can be used to build a single 32-processor partition, 4 separate 8-processor partitions, or various cell combinations in-between.

3 The Unisys ES7000/540 includes 32MB of shared, Level-4 cache for each 4 processor group. Each cache units supports multiple banks and independent read/write ports, tracks up to 64 processor requests, and features cache lines specifically tailored for Intel processors.

4 Each ES7000 cell contains a minimum of 4.0GB and a maximum of 16.0GB of main memory. 8.0GB was chosen for testing as it is the most commonly ordered customer configuration. The storage technology used is SDRAM DIMMs.

The operating system:

SUSE Linux Enterprise Server 9 RTM (SLES 9)

Kernel Version: 2.6.5-7.97-bigsmp

file systems[5] :

ext2	1.34
ext3	1.34
JFS	1.1.7
reiserFS v3 (reiser3)	3.6.13
XFS	2.6.25

and benchmark tool:

Iozone revision 3.226 (compiled for ia32 Linux)

Methodology

In order to achieve high levels of test integrity and repeatability, certain testing methodologies were used. These methodologies are listed below:

- All tests were run twice.
- Each metric was averaged with the corresponding metric for the subsequent test.
- System was rebooted between each test.
- Software environment remained unchanged during testing.
- The two EMC CX-Series LUNs were on the same SP, isolated from the other LUNs on the storage system.

Section 13

Comparative Analysis

This section details the comparative read, write, re-write, re-read, random write, and random read performance between the tested file systems.

5 File System versions were obtained from the output of a “mkfs.<fs_name> -V” command.

Write Performance

Write performance was analyzed on several scales. For each test performed, numbers were calculated for maximum[6], minimum[7], total average[8], target average[9], ora-ta[10] and specfp-ta[11] write throughput. Also, an overall write score was artificially created by taking the summation of the five above mentioned categories. The overall write score serves as a general indicator of how the file system performs in comparison to the other file systems.

For each file system, the best of each category was compared against the best of each category for the competing file systems. For example, the best ext3 test in regards to total average write performance was compared against the best JFS test in regards to total average write performance. Using this technique, we can see how the file systems compare with each other in their respective best-case scenarios.

For best maximum write throughput, XFS came out on top, with a value of 692,055.00KB/s. It beat the second place file system, JFS, by a margin of 11.27%.

For best minimum write throughput, XFS came out on top, with a value of 74,001.00KB/s. It beat the second place file system, JFS, by a margin of 19.91%.

For best total average write throughput, JFS came out on top, with a value of 109,455.16KB/s. It beat the second place file system, XFS, by a margin of 5.14%.

For best target average write throughput, JFS came out on top, with a value of 142,904.89KB/s. It beat the second place file system, XFS, by a margin of 20.10%.

For best ora-ta write throughput, JFS came out on top, with a value of 124,183.84KB/s. It beat the second place file system, XFS, by a margin of 13.42% increase.

For best specfp-ta write throughput, JFS came out on top, with a value of 108,244.74KB/s. It beat the second place file system, ext2, by a margin of 7.85%.

JFS exhibited the best overall write performance, but XFS produced almost identical results. The best overall write score for JFS was obtained in the *jfs_noatime* run, where a score of 1,147,056.63 was obtained. XFS was very comparable, with a score of 1,145,665.21–only 0.12% worse. After JFS and XFS, the remaining scores drop off dramatically. The last-place file system in this category, ext3, produced a best overall write score of 662,505.87, a considerable 42.24% less than the JFS score.

6 The maximum metric is obtained by extracting the maximum value among the collective data sets of both runs for any given test.
7 The minimum metric is obtained by extracting the minimum value among the collective data sets of both runs for any given test.
8 The total average metric is obtained by averaging all values collected between the both runs for any given test.
9 Total target average metric is obtained by first averaging each corresponding metric for both runs for any given test. Then, the subset encompassing record sizes 1kb to 32kb and file sizes 4kb to 2MB is averaged to produce the target average metric. This metric is designed to provide a more applicable measurement of file system performance than total average when applied to real-world situations.
10 ora-ta, or Oracle Target Average, is obtained by first averaging each corresponding metric for both runs for any given test. Then, the subset encompassing record sizes 4kb to 16kb and file sizes 4kb to 4GB is averaged to produce the ora-ta metric. This metric is designed to provide a more applicable measurement of file system performance than total average when applied to Oracle deployments. The record size and file size subset was chosen by analyzing strace logs from an Oracle server. Of course, the subset chosen will not necessarily represent any specific real-world Oracle deployment, but it can provide a good indicator of potential Oracle performance for the given file system.
11 specfp-ta, or SpecFP Target Average, is obtained by first averaging each corresponding metric for both runs for any given test. Then, the subset encompassing record sizes 1 kb to 4 kb and file sizes 4kb to 4GB is averaged to produce the specfp-ta metric. This metric is designed to provide a more applicable measurement of file system performance than total average when applied to benchmarking with the SpecFP benchmark tool. The record size and file size subset was chosen by analyzing strace logs from an in-house SpecFP benchmark run on an ES7000 server.

Read Performance

Read performance was analyzed on several scales. For each test performed, numbers were calculated for maximum, minimum, total average, target average, ora-ta and specfp-ta read throughput. Also, an overall read score was artificially created by taking the summation of the five above categories. The overall read score serves as a general indicator of how the file system performs in comparison to the other file systems.

For each file system, the best of each category was compared against the best of each category for the competing file systems. For example, the best ext3 test in regards to total average read performance was compared against the best JFS test in regards to total average read performance. Using this technique, we can see how the file systems compare with each other in their respective best-case scenarios.

For best maximum read throughput, the file systems compared almost exactly. Ext2, ext3, JFS and XFS all recorded a value of 4,000,000.00KB/s. Reiser3, the only file system not to record this number recorded a maximum score of 3,924,555.00. Since the reiser3 score is a mere 1.89% less than the other scores, it is assumed this deviation is a result of random testing variation.

For best minimum read throughput, ext3 came out on top, with a value of 201,009.00KB/s. It beat the second place file system, XFS, by 1.05%.

For best total average read throughput, ext3 came out on top, with a value of481,221.25KB/s. It beat the second place file system, ext2, by a margin of 1.32%.

For best target average read throughput, ext3 came out on top, with a value of 525,976.81KB/s. It beat the second place file system, ext2, by a margin of 2.53%.

For best ora-ta read throughput, ext3 came out on top, with a value of 532,294.45KB/s. It beat the second place file system, ext2, by a margin of 0.92%.

For best specfp-ta read throughput, ext2 came out on top, with a value of 488,116.46KB/s. It beat the second place file system, ext2, by a margin of 1.78%.

Ext2 scored the best overall read score – 6,177,114.22 – compared to ext3's second place score of 6,152,515.24. Considering the benefit of ext3 journaling, ext2's 0.40% performance benefit is negligible. It should be noted that in the read tests, all the file systems exhibited very similar performance numbers; in fact, the difference between the best and the worst overall read score was a scant 2.55%.

The best overall ext3 score was obtained in the *ext3_wb_noatime_dl* run, where ext3 was configured with the *data=writeback* and *noatime* mount options, and run with the deadline I/O scheduler.

Re-Write Performance

Re-write performance was analyzed on several scales. For each test performed, numbers were calculated for maximum, minimum, total average, target average, ora-ta and specfp-ta re-write throughput. Also, an overall re-write score was artificially created by taking the summation of the five above mentioned categories. The overall re-write score serves as a general indicator of how the file system performs in comparison to the other file systems.

For each file system, the best of each category was compared against the best of each category for the competing file systems. For example, the best ext3 test in regards to total average re-write performance was compared against the best JFS test in regards to total average re-write performance. Using this technique, we can see how the file systems compare with each other in their respective best-case scenarios.

For best maximum re-write throughput, reiser3 came out on top, with a value of 120,896.00KB/s. It beat the second place file system, JFS, by a margin of 0.41%.

For best minimum re-write throughput, JFS came out on top, with a value of 86,441.00KB/s. It beat the second place file system, XFS, by a margin of 3.96%.

For best total average re-write throughput, JFS came out on top, with a value of 110,386.72KB/s. It beat the second place file system, ext2, by a margin of 0.91%.

For best target average re-write throughput, JFS came out on top, with a value of 112,430.98KB/s. It beat the second place file system, ext2, by a margin of 1.44%.

For best ora-ta re-write throughput, JFS came out on top, with a value of 113,429.85KB/s. It beat the second place file system, ext2, by a margin of 1.53%.

For best specfp-ta re-write throughput, JFS came out on top, with a value of 105,005.71KB/s. It beat the second place file system, ext2, by a margin of 0.88%.

JFS exhibited the best overall re-write performance, with XFS very close — within 1.98%. The remaining file systems were also very comparable, with only a 5.65% difference between the best and worst overall scores.

The best overall re-write score for JFS was the *jfs_xj_noop* run, where JFS recorded a score of 646,027.27. The run consisted of specifying an external journal and running with the noop scheduler. Despite the excellent score with the noop scheduler, the *jfs_xj_dl* run recorded a 639,206.53 score using the deadline scheduler–only a 1.06% drop in performance. The versatility of the deadline scheduler is worth the almost inconsequential performance difference.

Re-Read Performance

Re-read performance was analyzed on several scales. For each test performed, numbers were calculated for maximum, minimum, total average, target average, ora-ta and specfp-ta re-read throughput. Also, an overall re-read score was artificially created by taking the summation of the five above mentioned categories. The overall re-read score serves as a general indicator of how the file system performs in comparison to the other file systems.

For each file system, the best of each category was compared against the best of each category for the competing file systems. For example, the best ext3 test in regards to total average re-read performance was compared against the best JFS test in regards to total average re-read performance. Using this technique, we can see how the file systems compare with each other in their respective best-case scenarios.

For best maximum re-read throughput, ext2 came out on top, with a value of 8,000,000.00KB/s. It beat the second place file system, ext3, by a margin of 2.44%.

For best minimum re-read throughput, reiser3 came out on top, with a value of 285,281.00KB/s. It beat the second place file system, JFS, by a margin of 0.11%.

For best total average re-read throughput, ext3 came out on top, with a value of 1,514,149.15KB/s. It beat the second place file system, JFS, by a margin of 2.70%.

For best target average re-read throughput, ext3 came out on top, with a value of 2,731,142.96KB/s. It beat the second place file system, JFS, by a margin of 1.42%.

For best ora-ta re-read throughput, ext3 came out on top, with a value of 2,001,637.73KB/s. It beat the second place file system, JFS, by a margin of 5.19%.

For best specfp-ta re-read throughput, ext3 came out on top, with a value of 1,255,214.02KB/s. It beat the second place file system, JFS, by a margin of 2.90%.

Ext2 exhibited the best overall re-read performance with a score of 15,466,564.01, with its journaling counterpart—ext3—within 0.17% at 15,440,345.68. No other file system was able to break the 15,000,000 mark, with the next best file system—reiser3—coming in at 12,742,470.92. The difference between the best and the worst overall re-read performance was 24.10%.

Considering the almost identical performance of ext3 and the added journal integrity, ext3 is the clear winner. The best overall re-read run for ext3 was obtained in the *ext3_wb_noatime_dl_xj* run, where ext3 was configured with an external journal, run with the *data=writeback* journal option, the noatime mount option and the deadline scheduler was used.

Oddly enough, neither XFS nor JFS obtained a single first place score. In fact, XFS scored last place in every category. In the best maximum re-read category, ext2 scored 72.14% better than XFS and 53.18% better than JFS!

Random Write Performance

Random write performance was analyzed on several scales. For each test performed, numbers were calculated for maximum, minimum, total average, target average, ora-ta and specfp-ta random write throughput. Also, an overall random write score was artificially created by taking the summation of the five above categories. The overall random write score serves as a general indicator of how the file system performs in comparison to the other file systems.

For each file system, the best of each category was compared against the best of each category for the competing file systems. For example, the best ext3 test in regards to total average random write performance was compared against the best JFS test in regards to total average random write performance. Using this technique, we can see how the file systems compare with each other in their respective best-case scenarios.

For best maximum random write throughput, JFS came out on top, with a value of 227,559.00KB/s. It beat the second place file system, XFS, by a margin of 2.05%.

For best minimum random write throughput, JFS came out on top, with a value of 33,290.00KB/s. It beat the second place file system, reiser3, by a margin of 0.87%.

For best total average random write throughput, JFS came out on top, with a value of 136,866.80KB/s. It beat the second place file system, ext2, by a margin of 1.48%.

For best target average random write throughput, JFS came out on top, with a value of 153,029.19KB/s. It beat the second place file system, ext2, by a margin of 1.43%.

For best ora-ta random write throughput, JFS came out on top, with a value of 141,167.47KB/s. It beat the second place file system, ext2, by a margin of 1.22%.

For best specfp-ta random write throughput, JFS came out on top, with a value of 126,405.71KB/s. It beat the second place file system, ext2, by a margin of 0.60%.

JFS exhibited the best overall random write performance, logging a score of 810,465.53. This was obtained in the *jfs_xj* run, where JFS was configured with an external journal. There was little variation between file systems in this category with the last place file system, ext3, scoring 757,645.99, a 6.52% decrease from the JFS score.

Random Read Performance

Random read performance was analyzed on several scales. For each test performed, numbers were calculated for maximum, minimum, total average, target average, ora-ta and specfp-ta random read throughput. Also, an overall random read score was artificially created by taking the summation of the five above mentioned categories. The overall random read score serves as a general indicator of how the file system performs in comparison to the other file systems.

For each file system, the best of each category was compared against the best of each category for the competing file systems. For example, the best ext3 test in regards to total average random read performance was compared against the best JFS test in regards to total average random read performance. Using this technique, we can see how the file systems compare with each other in their respective best-case scenarios.

For best maximum random read throughput, ext3 came out on top, with a value of 4,056,575.00KB/s. It beat the second place file system, reiser3, by a margin of 0.34%.

For best minimum random read throughput, ext3 came out on top, with a value of 171,064.00KB/s. It beat the second place file system, XFS, by a margin of 0.77%.

For best total average random read throughput, ext3 came out on top, with a value of 1,369,537.82KB/s. It beat the second place file system, ext2, by a margin of 2.32%.

For best target average random read throughput, ext3 came out on top, with a value of 2,063,384.40KB/s. It beat the second place file system, ext2, by a margin of 2.34%.

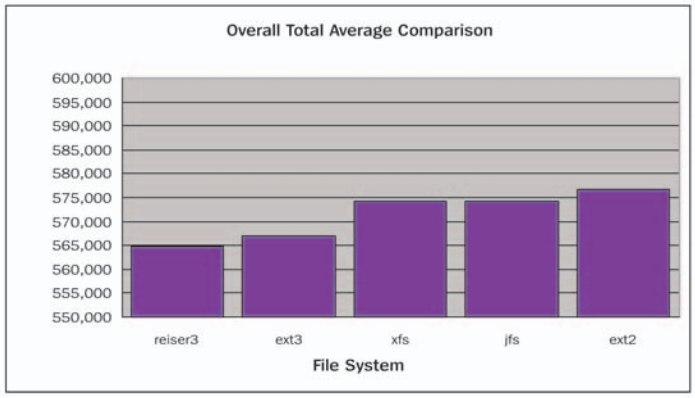
For best ora-ta random read throughput, ext3 came out on top, with a value of 1,583,687.65KB/s. It beat the second place file system, ext2, by a margin of 1.75%.

For best specfp-ta random read throughput, ext2 came out on top, with a value of 894,261.81KB/s. It beat the second place file system, reiser3, by a margin of 3.87%.

Similar to the read performance, all the file systems exhibit very comparable random read performance when properly tuned. Overall, ext3 scored the best overall random read score of 10,009,709.67. The second place file system, ext2, scored 1.93% worse, and the difference between the best and worst performers in this category was only 5.59%. However, ext3 was the only file system to break a score of 10,000,000 in the overall random read score category. This score was obtained on the *ext3_wb_noatime_dl* run, where the ext3 file system was mounted with both the *data=writeback* and *noatime* options and the deadline I/O scheduler was used.

Total Average Read/Write

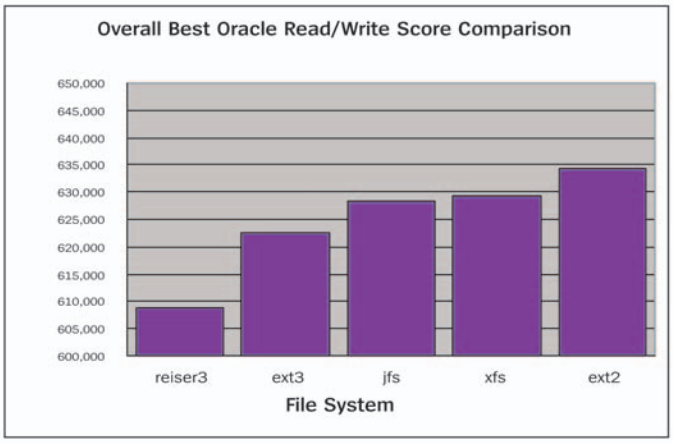
The total average is the average of all the data points for a test run, comprising a wide variety of record sizes and file sizes. Specifically, record sizes from 1k to 1MB were tested and file sizes from 4kb to 4GB were tested. While somewhat unrealistic when applied directly to real-world workloads, this measurement provides a good indicator for the flexibility of a file system. This metric shows which file systems produced the greatest aggregate numbers in the read tests and the write tests, suggesting the raw power of the file system.



As illustrated in the chart above, ext2 generated the best overall numbers. Focusing on journaling file systems, JFS and XFS achieved scores of 576,812.61 and 574,300.44 respectively and offer performance within 1% of ext2.

Ora-ta Performance

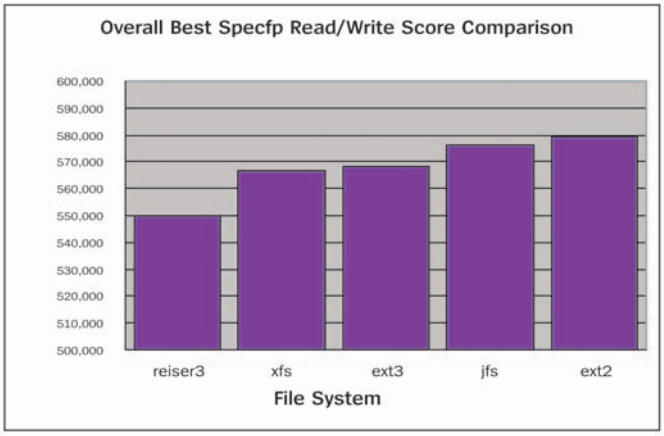
Ora-ta performance is an artificial score created by summing the best ora-ta write score and the best ora-ta read score for a file system. The ora-ta scores are generated by taking a subset of the data points for a test. The subset consists of all data points in the area occupied by record sizes ranging from 4kb to 16kb and file sizes ranging from 4kb to 4GB. This subset was chosen based on strace records of an Oracle server under load. This score is not necessarily indicative of all Oracle deployments and actual results will vary based on the environment and workload experienced. This score is intended to serve as a general guideline for Oracle performance based on file system.



Ext2 scored the best ora-ta read/write score, but JFS and XFS were once again within 1% of ext2's high water mark and offer guaranteed file system integrity.

Specfp-ta Read/Write

Specfp-ta performance is an artificial score created by summing the best specfp-ta write score and the best specfp-ta read score for a file system. The specfp-ta scores are generated by taking a subset of the data points for a test. The subset consists of all data points in the area occupied by record sizes ranging from 1kb to 4kb and file sizes ranging from 4kb to 4GB. This subset was chosen based on strace records of a server under load while running the specfp benchmark. This score is not indicative of all specfp test beds or test scenarios and actual results will vary based on a number of factors. Instead, this score is intended to serve as a general guideline for specfp performance based on file system.

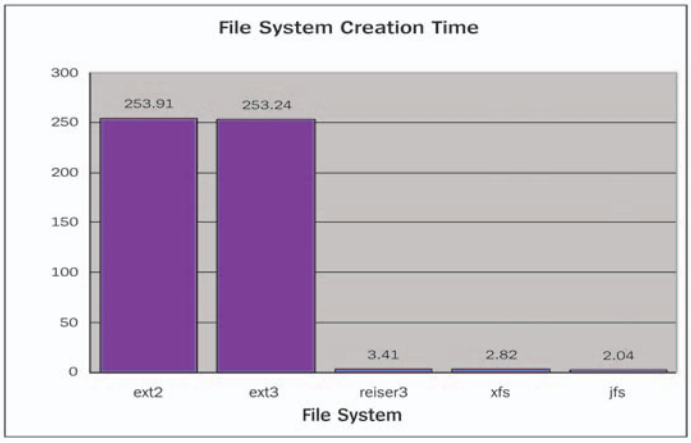


JFS scored the best specfp-ta read/write score by a journaled file system. Once again, we see reiser3 at the bottom end of the performance comparison.

File System Creation

It is rare that an administrator or user is in a position where the time to create a new file system is a deciding factor when selecting a file system. Nevertheless, in disaster recovery planning for mission critical systems, every aspect of system recovery must be addressed and time-efficient procedures are the backbone of a solid contingency plan. For whatever value the reader may find, included is a graph showing the amount of time to create a new file system.

The target of the file system is a single, primary partition occupying the total space on a 16-disk RAID 0 LUN with a raw capacity of 566,772,105,216 bytes (527.848GB).



Even more interesting than JFS's leading file system creation time of 2.04 seconds is that ext2 and ext3 are significantly slower than their competitors—each taking in excess of 4 minutes to create a file system.

Section 14

Summary

Overall, JFS and XFS offer pack-leading performance in a wide variety of scenarios and can stand head to head against ext2 from a performance standpoint. JFS and XFS offer journaling to guarantee file system integrity, a robust collection of user-space tools, fast file system creation, and in the case of XFS, unique enterprise features such as Guaranteed Rate I/O (GRIO). For an enterprise deployment, JFS and XFS are both excellent choices.

Between JFS and XFS, JFS offers slightly better overall performance and a smaller deviation of throughput metrics than XFS, where as XFS offers slightly better specfp-ta performance. As for limitations, neither XFS nor JFS offer journaling of user data.

If reliability is a concern that complicates a decision to move off of ext2/ext3, remember that XFS is the oldest journaling file system for Linux (just not the first in the mainstream kernel) and ported from the file system shipped with the entire line of SGI hardware. IBM has a similar philosophy and bundles JFS in their enterprise server line. Both are listed as “Development Status: 5 - Production/Stable” under their respective SourceForge sites.

The combination of excellent performance, guaranteed file system integrity, and strong roots as mainframe-class file systems lends to the enterprise credibility of both JFS and XFS.

Looking forward, reiser4 is exceptionally interesting on paper and is worth keeping an eye on. As a next-generation file system, the feature list is well encompassing—supporting removable media, long file names in excess of 1024 characters, atomic operations, and a plug-in interface to provide a means of customization and long lifespan for the file system. The downsides of reiser4 include non-existent enterprise distribution support, unproven reliability, and questionable performance given the relatively lackluster iozone metrics generated by its predecessor reiser3. In regards to reiser4 performance, Namesys states that reiser4 is the fastest file system, and has benchmarks to prove it at: http://www.namesys.com/benchmarks.html.

Before deciding on the appropriate file system, be sure to verify that the operating system vendor will support the decision. Refer to the File System Support section for this information.

References

An In-Depth Look at ReiserFS
http://www.linuxplanet.com/linuxplanet/tutorials/2926/4/

Appendix A. File Systems in Linux / A.4. Large File Support in Linux
http://www.novell.com/documentation/suse91/suselinux-adminguide/html/apas04.html

Design and Implementation of the Second Extended File System
http://e2fsprogs.sourceforge.net/ext2intro.html

Ext2 Introduction
http://web.mit.edu/tytso/www/linux/ext2intro.html

Ext3 External Journal How to
https://listman.redhat.com/archives/ext3-users/2003-September/msg00003.html

Ext3 FAQ
http://batleth.sapienti-sat.org/projects/FAQs/ext3-faq.html

File System Comparisons
http://en.wikipedia.org/wiki/Comparison_of_file_systems#fn_4

JFS
http://jfs.sourceforge.net

JFS Log White Paper
http://jfs.sourceforge.net/project/pub/jfslog/jfslog.pdf

JFS Tuning
http://www.hp.com/products1/unix/operating/infolibrary/whitepapers/JFS_Tuning_1.pdf

Journaled File Systems
http://www.webopedia.com/TERM/J/journaled_file_system.html

Large File Support in Linux
http://www.suse.de/~aj/linux_lfs.html

Linux ext3 FAQ
http://batleth.sapienti-sat.org/projects/FAQs/ext3-faq.html

Open source: JFS project Web Site
http://jfs.sourceforge.net

Open Source XFS for Linux Datasheet
http://www.sgi.com/pdfs/2508.pdf

Performance Management Guide – Monitoring and Tuning File Systems
http://publibn.boulder.ibm.com/doc_link/en_US/a_doc_lib/aixbman/prftungd/prftungd57.htm#HDRJ12A04N2001DIFF

Scalability and Performance in Modern Filesystems
http://linux-xfs.sgi.com/projects/xfs/papers/xfs_white/xfs_white_paper.html

Tuning SUSE Linux Enterprise Server on IBM e-server xSeries Servers
http://www.redbooks.ibm.com/redpapers/pdfs/redp3862.pdf

Whitepaper: Red Hat's New Journaling File System: ext3
http://www.redhat.com/support/wpapers/redhat/ext3/

XFS
http://oss.sgi.com/projects/xfs

XFS mkfs.xfs
http://www.die.net/doc/linux/man/man8/mkfs.xfs.8.html

Appendix A

Summary of Comparative Write Performance

Best Maximum Write Performance

fs	throughput/score	best	delta1	delta2
ext3	333,045.00		51.88%	-107.80%
reiser3	336,937.00		51.31%	-105.40%
ext2	516,312.00		25.39%	-34.04%
JFS	614,084.00		11.27%	-12.70%
XFS	692,055.00	best	0.00%	0.00%

Best Minimum Write Performance

fs	throughput/score	best	delta1	delta2
reiser3	48,723.00		34.16%	-51.88%
ext2	51,920.00		29.84%	-42.53%
ext3	54,325.00		26.59%	-36.22%
JFS	59,266.50		19.91%	-24.86%
XFS	74,001.00	best	0.00%	0.00%

Best Total Average Write Performance

fs	throughput/score	best	delta1	delta2
ext3	85,830.84		21.58%	-27.52%
reiser3	90,177.15		17.61%	-21.38%
ext2	101,966.32		6.84%	-7.34%
XFS	103,830.92		5.14%	-5.42%
JFS	109,455.16	best	0.00%	0.00%

Best Target Average Write Performance

fs	throughput/score	best	delta1	delta2
ext3	86,060.02		39.78%	-66.05%
reiser3	87,836.81		38.53%	-62.69%
ext2	106,645.74		25.37%	-34.00%
XFS	114,186.42		20.10%	-25.15%
JFS	142,904.89	best	0.00%	0.00%

Best Ora-ta Write Performance

fs	throughput/score	best	delta1	delta2
reiser3	89,474.50		27.95%	-38.79%
ext3	90,088.63		27.46%	-37.85%
ext2	106,885.41		13.93%	-16.18%
XFS	107,514.24		13.42%	-15.50%
JFS	124,183.84	best	0.00%	0.00%

Best Specfp-ta Write Performance

fs	throughput/score	best	delta1	delta2
reiser3	71,133.30		34.28%	-52.17%
ext3	80,076.29		26.02%	-35.18%
XFS	99,675.54		7.92%	-8.60%
ext2	99,746.26		7.85%	-8.52%
JFS	108,244.74	best	0.00%	0.00%

Best Overall Write Performance

fs	throughput/score	best	delta1	delta2
ext3	662,505.87		42.24%	-73.14%
reiser3	705,070.90		38.53%	-62.69%
ext2	954,424.73		16.79%	-20.18%
XFS	1,145,665.21		0.12%	-0.12%
JFS	1,147,056.63	best	0.00%	0.00%

Summary of Comparative Re-write Performance

Best Maximum Re-write Performance

fs	throughput/score	best	delta1	delta2
ext3	117,750.00		2.60%	-2.67%
XFS	119,493.00		1.16%	-1.17%
ext2	119,867.00		0.85%	-0.86%
JFS	120,402.00		0.41%	-0.41%
reiser3	120,896.00	best	0.00%	0.00%

Best Minimum Re-write Performance

fs	throughput/score	best	delta1	delta2
reiser3	73,501.00		14.97%	-17.61%
ext3	76,842.00		11.10%	-12.49%
ext2	76,851.00		11.09%	-12.48%
XFS	83,016.00		3.96%	-4.13%
JFS	86,441.00	best	0.00%	0.00%

Best Total Average Re-write Performance

fs	throughput/score	best	delta1	delta2
reiser3	106,199.33		3.79%	-3.94%
ext3	107,072.91		3.00%	-3.09%
XFS	108,625.58		1.60%	-1.62%
ext2	109,381.12		0.91%	-0.92%
JFS	110,386.72	best	0.00%	0.00%

Best Target Average Re-write Performance

fs	throughput/score	best	delta1	delta2
reiser3	105,970.33		5.75%	-6.10%
ext3	109,259.73		2.82%	-2.90%
XFS	109,997.27		2.16%	-2.21%
ext2	110,809.53		1.44%	-1.46%
JFS	112,430.98	best	0.00%	0.00%

Best Ora-ta Re-write Performance

fs	throughput/score	best	delta1	delta2
reiser3	108,568.48		4.29%	-4.48%
ext3	110,329.70		2.73%	-2.81%
XFS	111,522.75		1.68%	-1.71%
ext2	111,698.90		1.53%	-1.55%
JFS	113,429.85	best	0.00%	0.00%

Best Specfp-ta Re-write Performance

fs	throughput/score	best	delta1	delta2
reiser3	96,436.78		8.16%	-8.89%
ext3	101,413.49		3.42%	-3.54%
XFS	101,985.47		2.88%	-2.96%
ext2	104,084.05		0.88%	-0.89%
JFS	105,005.71	best	0.00%	0.00%

Best Overall Re-write Performance

fs	throughput/score	best	delta1	delta2
reiser3	609,504.78		5.65%	-5.99%
ext3	621,144.68		3.85%	-4.01%
ext2	632,127.20		2.15%	-2.20%
XFS	633,246.90		1.98%	-2.02%
JFS	646,027.27	best	0.00%	0.00%

Summary of Comparative Read Performance

Best Maximum Read Performance

fs	throughput/score	best	delta1	delta2
reiser3	3,924,555.00		1.89%	-1.92%
ext3	4,000,000.00	best	0.00%	0.00%
ext2	4,000,000.00	best	0.00%	0.00%
JFS	4,000,000.00	best	0.00%	0.00%
XFS	4,000,000.00	best	0.00%	0.00%

Best Minimum Read Performance

fs	throughput/score	best	delta1	delta2
ext2	182,727.00		9.10%	-10.01%
reiser3	190,761.00		5.10%	-5.37%
JFS	197,505.00		1.74%	-1.77%
XFS	198,902.00		1.05%	-1.06%
ext3	201,009.00	best	0.00%	0.00%

Best Total Average Read Performance

fs	throughput/score	best	delta1	delta2
JFS	464,845.28		3.40%	-3.52%
XFS	470,414.42		2.25%	-2.30%
reiser3	474,546.94		1.39%	-1.41%
ext2	474,846.29		1.32%	-1.34%
ext3	481,221.25	best	0.00%	0.00%

Best Target Average Read Performance

fs	throughput/score	best	delta1	delta2
JFS	487,649.83		7.29%	-7.86%
XFS	493,754.77		6.13%	-6.53%
reiser3	509,074.02		3.21%	-3.32%
ext2	512,690.81		2.53%	-2.59%
ext3	525,976.81	best	0.00%	0.00%

Best Ora-ta Read Performance

fs	throughput/score	best	delta1	delta2
JFS	504,085.67		5.30%	-5.60%
reiser3	519,277.93		2.45%	-2.51%
XFS	521,711.15		1.99%	-2.03%
ext2	527,409.93		0.92%	-0.93%
ext3	532,294.45	best	0.00%	0.00%

Best Specfp Read Performance

fs	throughput/score	best	delta1	delta2
XFS	467,124.42		4.30%	-0.53%
JFS	468,162.75		4.09%	-0.50%
reiser3	479,018.26		1.86%	-0.23%
ext2	479,440.20		1.78%	-0.22%
ext3	488,116.46	best	0.00%	0.00%

Best Overall Read Performance

fs	throughput/score	best	delta1	delta2
reiser3	6,019,770.38		2.55%	-2.61%
JFS	6,091,441.53		1.39%	-1.41%
XFS	6,144,560.75		0.53%	-0.53%
ext3	6,152,515.24		0.40%	-0.40%
ext2	6,177,114.22	best	0.00%	0.00%

Summary of Comparative Re-Read Performance

Best Maximum Re-Read Performance

fs	throughput/score	best	delta1	delta2
XFS	4,647,284.00		41.91%	-72.14%
JFS	5,222,625.00		34.72%	-53.18%
reiser3	5,488,104.00		31.40%	-45.77%
ext3	7,804,539.00		2.44%	-2.50%
ext2	8,000,000.00	best	0.00%	0.00%

Best Minimum Re-Read Performance

fs	throughput/score	best	delta1	delta2
XFS	273,969.00		3.97%	-4.13%
ext2	275,404.00		3.46%	-3.59%
ext3	284,232.00		0.37%	-0.37%
JFS	284,957.00		0.11%	-0.11%
reiser3	285,281.00	best	0.00%	0.00%

Best Total Average Re-Read Performance

fs	throughput/score	best	delta1	delta2
XFS	1,425,724.29		5.84%	-6.20%
ext2	1,448,794.73		4.32%	-4.51%
reiser3	1,460,633.26		3.53%	-3.66%
JFS	1,473,209.88		2.70%	-2.78%
ext3	1,514,149.15	best	0.00%	0.00%

Best Target Average Re-Read Performance

fs	throughput/score	best	delta1	delta2
XFS	2,502,830.52		8.36%	-9.12%
ext2	2,666,977.03		2.35%	-2.41%
reiser3	2,673,586.88		2.11%	-2.15%
JFS	2,692,294.09		1.42%	-1.44%
ext3	2,731,142.96	best	0.00%	0.00%

Best Oracle Re-Read Performance

fs	throughput/score	best	delta1	delta2
XFS	1,786,969.66		10.72%	-12.01%
ext2	1,884,594.69		5.85%	-6.21%
reiser3	1,887,196.74		5.72%	-6.06%
JFS	1,897,789.67		5.19%	-5.47%
ext3	2,001,637.73	best	0.00%	0.00%

Best Specfp Re-Read Performance

fs	throughput/score	best	delta1	delta2
XFS	1,109,845.98		11.58%	-3.13%
ext2	1,197,430.63		4.60%	-1.11%
reiser3	1,205,615.83		3.95%	-0.90%
JFS	1,218,819.33		2.90%	-0.47%
ext3	1,255,214.02	best	0.00%	0.00%

Best Overall Re-Read Performance

fs	throughput/score	best	delta1	delta2
XFS	11,738,466.45		24.10%	-31.76%
JFS	12,671,238.64		18.07%	-22.06%
reiser3	12,742,470.92		17.61%	-21.38%
ext3	15,440,345.68		0.17%	-0.17%
ext2	15,466,564.01	best	0.00%	0.00%

Summary of Comparative Random Write Performance

Best Maximum RandWrite Performance

fs	*throughput/score*	*best*	*delta1*	*delta2*
ext3	212,380.00		6.67%	-7.15%
ext2	218,521.00		3.97%	-4.14%
reiser3	218,615.00		3.93%	-4.09%
XFS	222,891.00		2.05%	-2.09%
JFS	227,559.00	best	0.00%	0.00%

Best Minimum RandWrite Performance

fs	*throughput/score*	*best*	*delta1*	*delta2*
XFS	15,616.00		53.09%	-113.18%
ext3	28,667.00		13.89%	-16.13%
ext2	32,958.00		1.00%	-1.01%
reiser3	33,000.00		0.87%	-0.88%
JFS	33,290.00	best	0.00%	0.00%

Best Total Average RandWrite Performance

fs	*throughput/score*	*best*	*delta1*	*delta2*
reiser3	130,490.11		4.66%	-4.89%
ext3	130,512.48		4.64%	-4.87%
XFS	131,938.40		3.60%	-3.74%
ext2	134,834.66		1.48%	-1.51%
JFS	136,866.80	best	0.00%	0.00%

Best Target Average RandWrite Performance

fs	*throughput/score*	*best*	*delta1*	*delta2*
reiser3	143,777.26		6.05%	-6.43%
ext3	145,068.02		5.20%	-5.49%
XFS	148,413.43		3.02%	-3.11%
ext2	150,838.48		1.43%	-1.45%
JFS	153,029.19	best	0.00%	0.00%

Best Ora-ta RandWrite Performance

fs	*throughput/score*	*best*	*delta1*	*delta2*
reiser3	135,432.16		4.06%	-4.23%
XFS	136,179.76		3.53%	-3.66%
ext3	136,468.70		3.33%	-3.44%
ext2	139,451.48		1.22%	-1.23%
JFS	141,167.47	best	0.00%	0.00%

Best Specfp RandWrite Performance

fs	*throughput/score*	*best*	*delta1*	*delta2*
reiser3	116,930.63		7.50%	-8.10%
XFS	118,069.76		6.59%	-7.06%
ext3	120,093.63		4.99%	-5.26%
ext2	125,642.19		0.60%	-0.61%
JFS	126,405.71	best	0.00%	0.00%

Best Overall RandWrite Performance

fs	*throughput/score*	*best*	*delta1*	*delta2*
ext3	757,645.99		6.52%	-6.97%
XFS	767,737.30		5.27%	-5.57%
reiser3	776,330.15		4.21%	-4.40%
ext2	801,030.81		1.16%	-1.18%
JFS	810,465.53	best	0.00%	0.00%

Summary of Comparative Random Read Performance

Best Maximum RandRead Performance

fs	*throughput/score*	*best*	*delta1*	*delta2*
XFS	3,891,483.00		4.07%	-4.24%
JFS	3,892,778.00		4.04%	-4.21%
ext2	4,010,355.00		1.14%	-1.15%
reiser3	4,042,615.00		0.34%	-0.35%
ext3	4,056,575.00	best	0.00%	0.00%

Best Minimum RandRead Performance

fs	*throughput/score*	*best*	*delta1*	*delta2*
ext2	160,610.00		6.11%	-6.51%
reiser3	164,152.00		4.04%	-4.21%
XFS	165,782.00		3.09%	-3.19%
JFS	169,754.00		0.77%	-0.77%
ext3	171,064.00	best	0.00%	0.00%

Best Total Average RandRead Performance

fs	*throughput/score*	*best*	*delta1*	*delta2*
XFS	1,298,154.95		5.21%	-5.50%
reiser3	1,313,787.95		4.07%	-4.24%
JFS	1,321,598.37		3.50%	-3.63%
ext2	1,337,810.10		2.32%	-2.37%
ext3	1,369,537.82	best	0.00%	0.00%

Best Target Average RandRead Performance

fs	*throughput/score*	*best*	*delta1*	*delta2*
XFS	1,877,748.28		9.00%	-9.89%
reiser3	1,948,144.98		5.58%	-5.92%
JFS	1,974,142.53		4.33%	-4.52%
ext2	2,015,002.45		2.34%	-2.40%
ext3	2,063,384.40	best	0.00%	0.00%

Best Oracle RandRead Performance

fs	*throughput/score*	*best*	*delta1*	*delta2*
XFS	1,458,374.78		7.91%	-8.59%
reiser3	1,504,505.13		5.00%	-5.26%
JFS	1,527,955.30		3.52%	-3.65%
ext2	1,555,951.31		1.75%	-1.78%
ext3	1,583,687.65	best	0.00%	0.00%

Best Specfp RandRead Performance

fs	*throughput/score*	*best*	*delta1*	*delta2*
XFS	782,230.01		12.53%	-2.88%
JFS	838,225.25		6.27%	-1.44%
ext2	856,870.60		4.18%	-0.93%
reiser3	859,669.01		3.87%	-0.86%
ext3	894,261.81	best	0.00%	0.00%

Best Overall RandRead Performance

fs	*throughput/score*	*best*	*delta1*	*delta2*
XFS	9,450,224.02		5.59%	-5.92%
JFS	9,614,021.05		3.95%	-4.12%
reiser3	9,756,897.23		2.53%	-2.59%
ext2	9,816,513.47		1.93%	-1.97%
ext3	10,009,709.67	best	0.00%	0.00%

Appendix B

The author of this document encourages readers to send feedback on this document. Also, anyone wishing to receive an electronic copy of the results data for scrutiny and/or further analysis should contact the author at: troy.stepan@unisys.com

About the Author

John Troy Stepan is an engineer in the Linux Systems Group of Unisys Corporation and has worked on performance testing and benchmarking on the Unisys ES7000 server platform for both Windows and Linux operating systems. He is a graduate of the Pennsylvania State University and holds a Bachelor's degree in MS&IS.

For more information, contact your Unisys representative.

Or call:

1-800-874-8647, ext. 111 (U.S. and Canada)

00-1-585-742-6780, ext. 111 (Other countries)

In a hurry to learn more? Visit:

http://www.unisys.com/es7/linux

For even more detail, visit:

http://www.unisys.com/es7/linux/ecommunity

12/05

4126 7048-000