

# Scaling Source Control for the Next Generation of Game Development

by Mike Sundy & Tobias Roberts

Perforce User's Conference

May 2007, Las Vegas

## Introduction:

This document intends to show performance differences between various Windows and Linux Perforce server configurations.

## About the Authors

Tobias Roberts is the Studio IT Architect for the Electronic Arts Redwood Shores (EARS) and Sims Studios. He has been focused on server performance and studio architecture for the past nine years. E-mail: [troberts@ea.com](mailto:troberts@ea.com).

Mike Sundy is the Senior Perforce Administrator for the EARS/Sims Studios. He has over eleven years of version control administration experience and has specialized in Perforce for the past five years. E-mail: [msundy@ea.com](mailto:msundy@ea.com) or [phastman@hotmail.com](mailto:phastman@hotmail.com)

## Installation Metrics

At Electronic Arts, we check in art binary data in addition to code. There are typically a much greater number of data assets compared to code. 90% of files we store in Perforce are binary, the other 10% are code. Some of our servers have upwards of 1200 users, 6.3 million files, 600,000 changelists, and a 80 GB db.have. Oldest server has 7 years of P4 history. EA is primarily a Windows-based development shop, with upwards of 90% of client desktops running Windows. EA has over 4,000 Perforce users and 90+ Perforce servers scattered across dozens of worldwide studios. Our largest server has 1.5 TB of data. We use 5 TB of Perforce RCS storage total at the EARS/Sims studios. The EARS/Sims studios have 10 Perforce servers and approximately 1,000 users. This document and testing was primarily done around November of 2005.

## CurrentGen vs. NextGen

The Current Generation of game development used 9 GB capacity DVD's. (Xbox, PS2). Next Generation game media capacity is 30-50 GB (Blu-Ray, HD-DVD). This is a 3-5x increase in storage capacity, with a commensurate increase in the amount of data generated by game teams. Consumers expect increasingly realistic graphics and audio, which means larger development teams and a lot more data being shuttled around.

## Win32 Scalability and Performance Issues

One of our first NextGen development teams hit some serious scalability limitations with Win32. The three critical Win32 scalability issues they hit were: POSIX file descriptors, 2 GB per process memory limitation, and performance degradation during multiple concurrent forced syncs.

- **A POSIX Windows subsystem file descriptor limitation** which caused the Perforce service to crash and reset up to six times per day. Here is the description from Brett Taylor from Perforce in July 2005:

"We have an explanation for the file handle problem. Turns out the posix Windows subsystem which we use for file access in our server, has a file descriptor table hard coded to a limit of 2048 entries. As these entries are used, event handles are allocated to service critical sections and spin counts. This is why we see the Event handle count rise. The Event handle count is not the actual problem. It only indicates that the posix file descriptor table is being used. The actual problem comes from the limited size of the posix table, 2048. If you have a fast enough machine with enough horse power, which you have. Each client connection uses around 15 posix descriptors, multiply that by around 130 busy engineers and presto, you run out of file descriptors."

Update: This issue has apparently been fixed as of server release 2006.2.98791:  
#98791 \*\*

The Perforce Server on Windows now uses native I/O routines for database access rather than using the POSIX layer. This fixes the problem of hitting the maximum number of open POSIX file handles. (Bug #19375).

- **2 GB per process memory limit.** Windows 32-bit can only support up to 2 GB of memory per process. We spun up three instances of 'p4s.exe' on the server and hit memory issues and occasionally soaked up a full 8 GB of RAM on this server. If we wanted to stay on Win32, we would have to upgrade to Windows DataCenter server to get past the 8 GB total memory limit on Advanced Server. We have to spin up these separate 'p4s' processes to work around this memory limitation. On Unix each thread spawns with its own memory space so you are practically limited only by the physical RAM in the server.
- **Performance during multiple concurrent syncs of large client workspaces.** We saw performance degrade significantly with syncs of 10 users syncing 250k file client workspaces simultaneously. If I ran a force sync in the middle of the night with no one around, I could sync 40 GB in 40 minutes. If I ran a sync in the middle of the day during typical load, that same forced sync could take 3+ hours. The bandwidth was only at about 20% percent usage so we were not bandwidth bound. A lot of this degradation is due to disk I/O contention for the versioned files from forced syncs. Forced syncs are typically generated by build farms and users who are trying to reconcile their workspaces for tools like RenderWare Studio.

These issues all present some significant scalability and performance limitations of Win32 for the EA development environment.

### Hypothesis

We ran multiple tests of Perforce under load and tried several different configurations. If you run a side by side test of Linux vs. Windows on a single operation, they tend to run about the same speed for many operations (see Roger March and Shiv Sikand's whitepaper from the 2003 Perforce User's conference:

<http://www.perforce.com/perforce/conferences/us/2003/marchsikand/marchsikand.pdf>).

Our theory was that Linux performs better under load, when the limits of the hardware are approached and the OS and filesystem performance becomes more critical. We had heard anecdotally that Linux performs better than Windows for a Perforce server platform (both from several attendees at the 2005 Perforce User's Conference and from Kenny Wu at EA Canada) but we needed to test this theory to make sure it would perform with our large Next Generation datasets. We worked extensively with Michael Shields and Brett Taylor from Perforce to get guidance on the best way to configure a Perforce server for performance and what hardware setup they recommended for our goals. They were excellent and greatly helped speed up our testing.

## Test Configurations

### General Info

We were somewhat limited on our hardware choices since there was a lot of pressure to complete this testing and get a new system into production. Dell is our current hardware supplier. Most of our testing was focused on OS, filesystem, and disk configuration.

### Client Machine Configuration

Client configuration for single-user operation client-side tests:

CPU: Dual Pentium 4 3.6 GHz

RAM: 2 GB

Network: 1 GB Ethernet card

OS: Microsoft Windows XP Professional Version 2002 Service Pack 2

Perforce Client Version: 2004.2.76641

### Perforce Client Test

We cloned the Godfather Perforce data as of 4/2/05. Size of db.have for test instance: 19 GB

'p4 sync -f //godfather/dev/...' of godfather\_user\_template (minus '//project') to changelist 108868 – 39 GB of data, p4 files shows 179,2983 files (no deletes). Local directory is empty.

## Server Configurations

Config	Server	OS	Bit	RAM	CPU	Internal disk speed	SAN disk speed	p4 db	p4 journal	OS config	RCS config
1	Dell 6650	Win2K Adv. Server	32	8 GB	Quad Xeon 2.0	10k RPM	10k RPM	shared RAID 5, SAN, 64k stripe	shared RAID 5, SAN	4 disk RAID 5	shared RAID 5
2	Dell 6650	Win2K Adv. Server	32	8 GB	Quad Xeon 2.0	10k RPM	10k RPM	2 disk RAID 1, 64k stripe	1 disk JBOD	2 disk RAID 1	RAID 5, SAN,
3	Dell 2850	Linux 2.4 kernel, RedHat 3.0 AS	32	4 GB	Dual Xeon 3.2	10k RPM	10k RPM	2 disk RAID 1, 8kb stripe; ext3 writeback	shared 2 disk RAID 1, 4kb stripe; ext3 writeback	shared 2 disk RAID 1, ext3	RAID 5, SAN, ReiserFS, 64k stripe
4	Dell 2850	Linux 2.6 kernel, SUSE 2.6.5-7.97-smp	64	4 GB	Dual Xeon 3.2	10k RPM	10k RPM	2 disk RAID 1, 8kb stripe; ext3 writeback	shared 2 disk RAID 1, 4kb stripe; ext3 writeback	shared 2 disk RAID 1; ReiserFS	N/A
5	Dell 2850	Linux 2.6 kernel, SUSE 2.6.5-7.155.29-smp	64	4 GB	Dual Xeon 3.2	10k RPM	10k RPM	2 disk RAID 1, 8kb stripe; ReiserFS	shared 2 disk RAID 1, 4kb stripe; ReiserFS	shared 2 disk RAID 1; ReiserFS	RAID 5, SAN, ReiserFS, 64k stripe

6	Dell 2850	Linux 2.6 kernel, SUSE 2.6.5-7.155.29-smp	64	4 GB	Dual Xeon 3.2	10k RPM	10k RPM	2 disk RAID 1, 8kb stripe; XFS	shared 2 disk RAID 1, 4kb stripe; XFS	shared 2 disk RAID 1;ReiserFS	RAID 5, SAN, XFS, 64k stripe
7	Dell 2850	Linux 2.6 kernel, SUSE 2.6.5-7.155.29-smp	64	4 GB	Dual Xeon 3.2	10k RPM	10k RPM	4 disk RAID 10, 8kb stripe; XFS	shared 2 disk RAID 1, 4kb stripe; XFS	shared 2 disk RAID 1;ReiserFS	RAID 5, SAN, XFS, 64k stripe
8	Dell 2850	Linux 2.6 kernel, SUSE 2.6.5-7.155.29-smp	64	4 GB	Dual Xeon 3.2	10k RPM	10k RPM	4 disk RAID 10, 8kb stripe; XFS	shared 2 disk RAID 1, 4kb stripe; XFS	shared 2 disk RAID 1;ReiserFS	RAID 5, SAN, ReiserFS, 64k stripe
9	Dell 2850	Linux 2.6 kernel, SUSE 2.6.5-7.155.29-smp	64	4 GB	Dual Xeon 3.2	10k RPM	10k RPM	4 disk RAID 10, 8kb stripe; XFS	shared 2 disk RAID 1, 4kb stripe; XFS	shared 2 disk RAID 1;ReiserFS	RAID 5, SAN, ReiserFS, 64k stripe, SAN readahead cache 3
10	Dell 2850	Windows 2003 Advanced Server EM64T	64	4 GB	Dual Xeon 3.2	10k RPM	10k RPM	4 disk RAID 10, 8kb stripe; XFS	shared 2 disk RAID 1, 4kb stripe; XFS	shared 2 disk RAID 1;ReiserFS	RAID 5, SAN, ReiserFS, 64kb stripe, readahead cache 3
11	Dell 2850	Linux 2.6 kernel, SUSE 2.6.5-7.155.29-smp	64	4 GB	Dual Xeon 3.2	10k RPM	10k RPM	2 disk RAID 1, 8kb stripe; XFS	shared 2 disk RAID 1, 4kb stripe; XFS	shared 2 disk RAID 1;ReiserFS	RAID 5, SAN, ReiserFS, 64k stripe, SAN readahead cache 3
12	Dell 6850	Linux 2.6 kernel, SUSE 2.6.5-7.201-smp	64	16 GB	Quad Xeon 3.66	15k RPM	15k RPM	2 disk RAID 1, 8kb stripe; XFS	shared 2 disk RAID 1, 4kb stripe; XFS	shared 2 disk RAID 1;ReiserFS	RAID 5, SAN, ReiserFS, 64k stripe, SAN readahead cache 3
13	Dell 6850	Linux 2.6 kernel, SUSE 2.6.5-7.201-smp	64	16 GB	Quad Xeon 3.66	15k RPM	15k RPM	shared 4 disk RAID 10, 8kb stripe; XFS	shared 4 disk RAID 10, 4kb stripe; XFS	shared 4 disk RAID 10;ReiserFS	RAID 5, SAN, ReiserFS, 64k stripe, readahead cache set to 3 on SAN
14	Dell 6850	Linux 2.6 kernel, SUSE 2.6.5-7.201-smp	64	16 GB	Quad Xeon 3.66	15k RPM	15k RPM	3 disk RAID 5, 8kb stripe; XFS	shared 2 disk RAID 1, 4kb stripe; XFS	shared 2 disk RAID 1;ReiserFS	RAID 5, SAN, ReiserFS, 64k stripe, SAN readahead cache 3
15	Dell 2850	RedHat 4.0 AS 2.6.9-11 (custom)	64	8 GB	Dual Xeon 3.2	15k RPM	15k RPM	4 disk RAID 10, 8kb stripe; XFS	shared 2 disk RAID 1, 4kb stripe; XFS	shared 2 disk RAID 1;ext3	RAID 5, SAN, ReiserFS, 64k stripe, SAN readahead cache 3

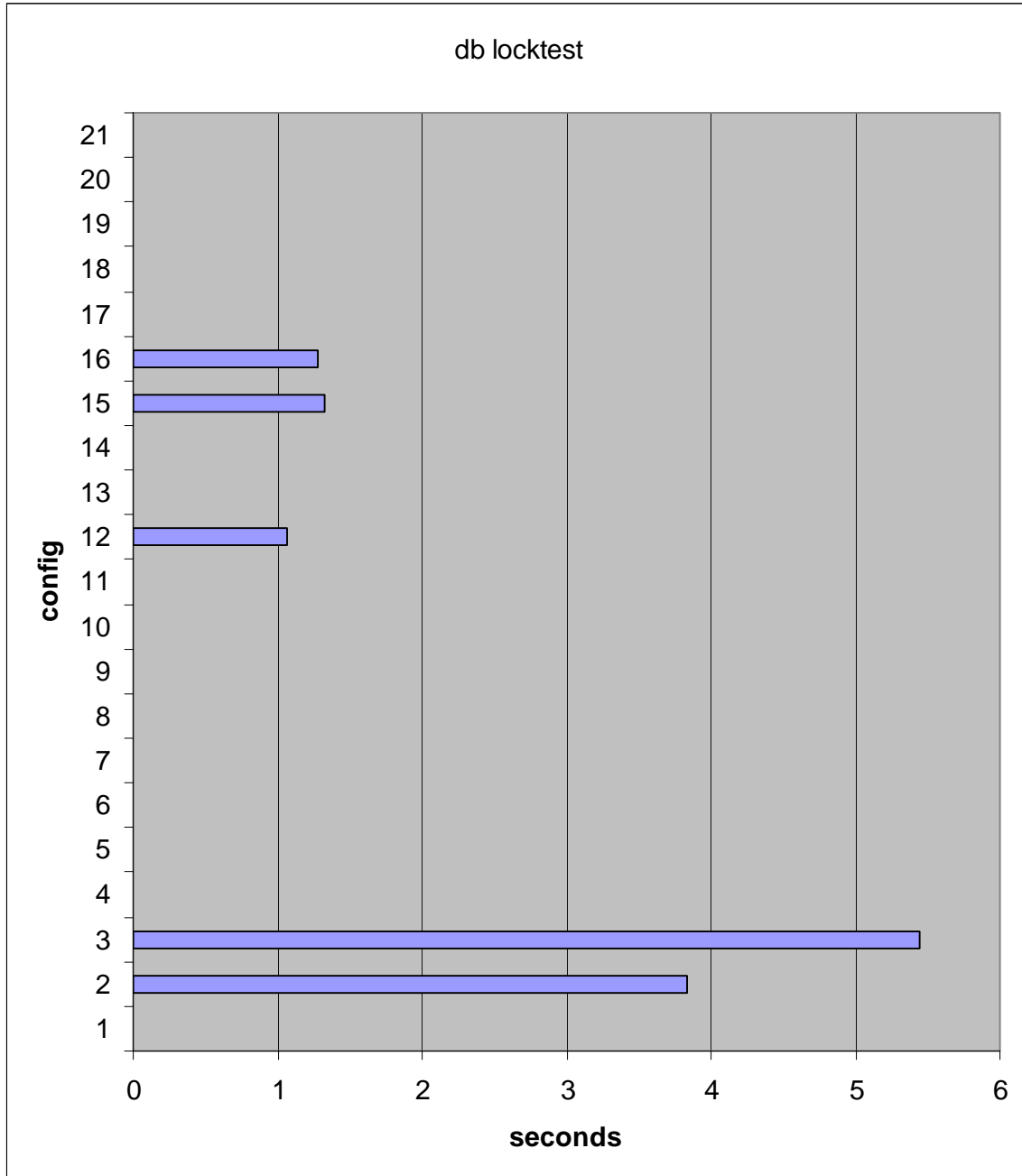
16	Dell 2850	Linux 2.6 kernel, SUSE 2.6.5-7.155.29-smp	64	8 GB	Dual Xeon 3.2	15k RPM	10k RPM	4 disk RAID 10, 8kb stripe; XFS	shared 2 disk RAID 1, 4kb stripe; XFS	shared 2 disk RAID 1;ReiserFS	RAID 5, SAN, ReiserFS, 64k stripe, SAN readahead cache 3
17	Dell 2850	Linux 2.6 kernel, SUSE 2.6.5-7.155.29-smp	64	8 GB	Dual Xeon 3.2	15k RPM	15k RPM	4 disk RAID 10, 8kb stripe; XFS	shared 2 disk RAID 1, 4kb stripe; XFS	shared 2 disk RAID 1;ReiserFS	RAID 5, SAN, ReiserFS, 64k stripe, readahead cache set to 3 on SAN
18	Dell 2850	Linux 2.6 kernel, SUSE 2.6.5-7.155.29-smp	64	8 GB	Dual Xeon 3.2	15k RPM	10k RPM	4 disk RAID 10, 8kb stripe; XFS	shared 2 disk RAID 1, 4kb stripe; XFS	shared 2 disk RAID 1;ReiserFS	RAID 5, SAN, XFS, 64k stripe, SAN readahead cache 3
19	Dell 2850	RedHat 4.0 AS 2.6.9-11 (custom)	64	8 GB	Dual Xeon 3.2	15k RPM	10k RPM	4 disk RAID 10, 8kb stripe; XFS	shared 2 disk RAID 1, 4kb stripe; XFS	shared 2 disk RAID 1;ext3	RAID 5, SAN, XFS, 64k stripe, SAN readahead cache 3
20	Dell 2850	RedHat 4.0 AS 2.6.9-11 (custom)	64	8 GB	Dual Xeon 3.2	15k RPM	10k RPM	4 disk RAID 10, 8kb stripe; XFS	shared 2 disk RAID 1, 4kb stripe; XFS	shared 2 disk RAID 1;ext3	RAID 5, SAN, ReiserFS, 64k stripe, SAN readahead cache 3
21	Dell 2850	RedHat 4.0 AS 2.6.9-11 (custom)	64	8 GB	Dual Xeon 3.2	15k RPM	15k RPM	4 disk RAID 10, 8kb stripe; XFS	shared 2 disk RAID 1, 4kb stripe; XFS	shared 2 disk RAID 1;ext3	RAID 5, SAN, XFS, 64k stripe, SAN readahead cache 3

## Test Results

### Test #1: Filesystem Lock Test

This test showed how quickly the configuration could grab locks on 1 million files on the filesystem. This test was run using the "locktest" benchmarking tool written by Brett Taylor of Perforce Software.

Lower is better

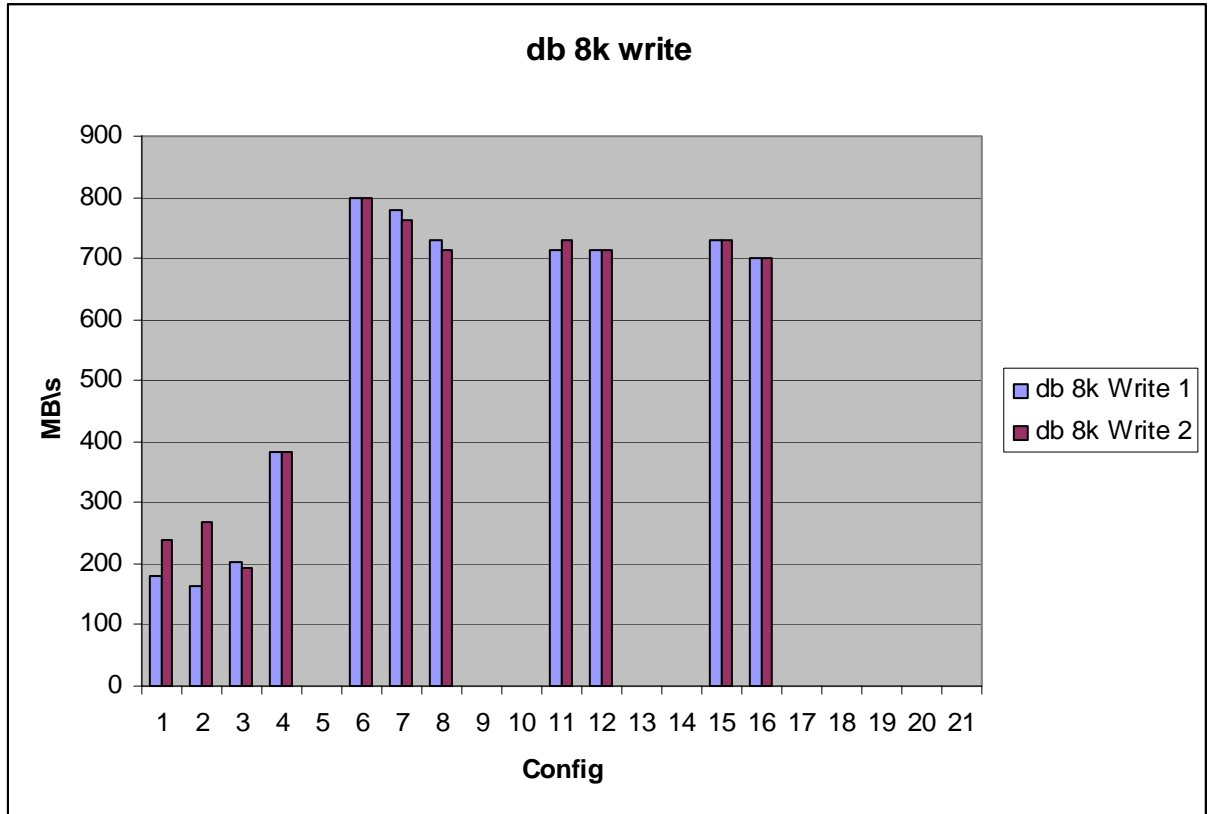


Analysis: The Linux 64-bit configs were the clear winner here, able to grab 1 million locks more than 3x faster than the closest 32-bit competitor. It trounced both Windows and Linux 32-bit.

### Test #2: DB 8k write

This test shows write throughput on the db volume with 8k block size. This test and tests 2-11 were run using the "fstst" benchmarking tool written by Brett Taylor of Perforce Software. This metric indicates how much data per second could theoretically be written to the Perforce databases.

Higher is better.

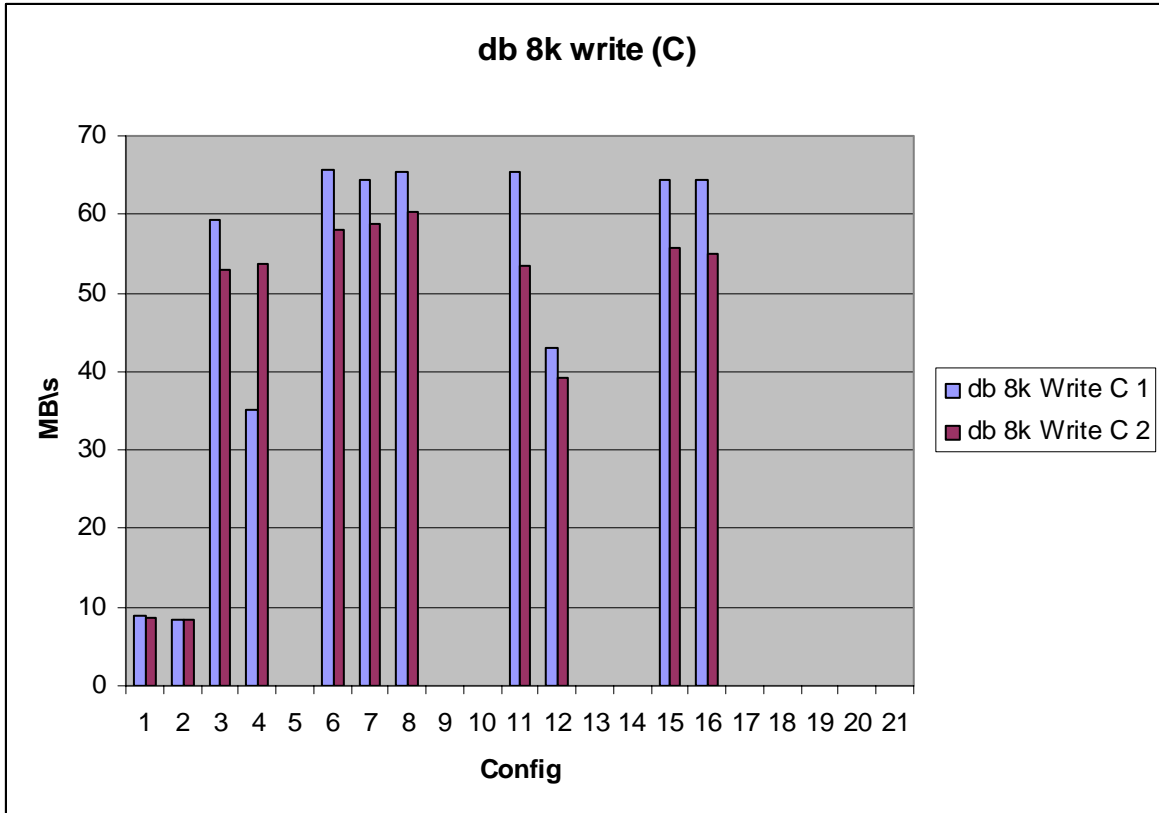


Analysis: The Win32 configs (1&2) were the weakest performers here, as was Linux 64-bit with ext3 filesystem (4). The XFS filesystem had 2x faster write throughput than ext3 (even with writeback enabled) on Linux 64-bit.

### Test #3: DB 8k write (Commit)

This test shows write with commit throughput on the db volume with 8k block size. The "Commit" means that the tool attempts to force a write commit to disk instead of just writing to cache.

Higher is better

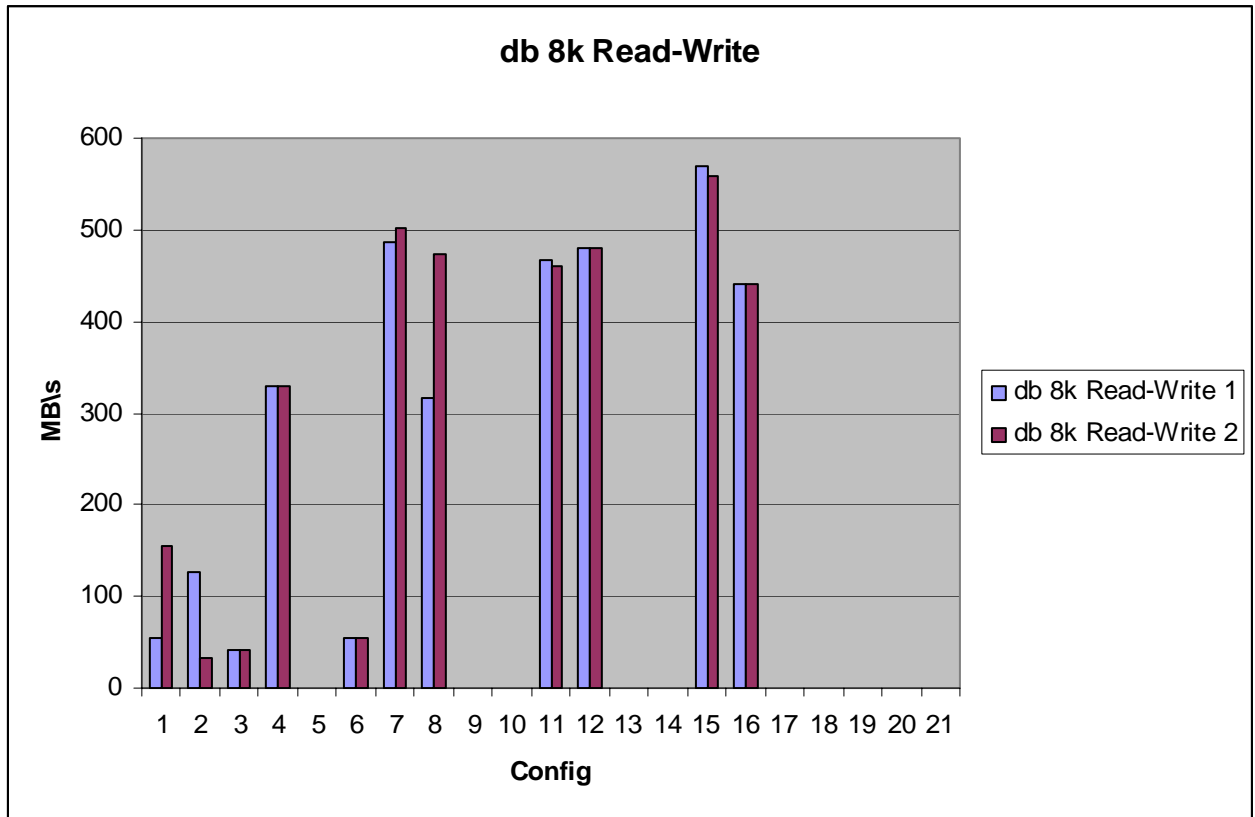


Analysis: The Win32 configs (1&2) were the weakest performers again. However, ext3 on both 32-bit (3) and 64-bit (4) was close to XFS on 64-bit (6-8,11-12, 15-16) on this particular metric.

#### Test #4: DB 8k Read-Write

This test shows read-write throughput on the db volume with 8k block size.

Higher is better

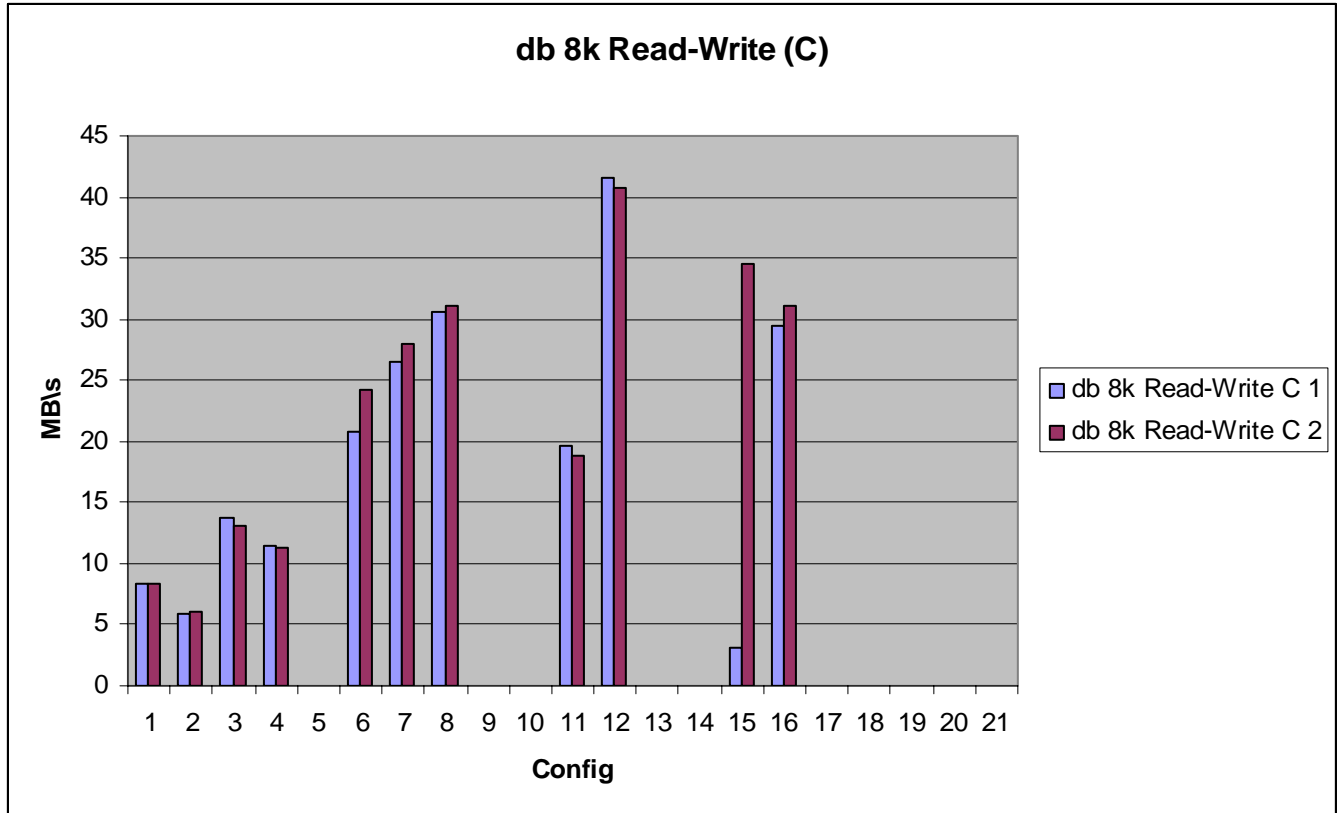


Analysis: The Win32 configs (1-2) were the weakest performers again. There is a large jump in performance from 32-bit Linux with 2.4 kernel (3) to 64-bit Linux with 2.6 kernel (4), even with ext3. However, 64-bit Linux on XFS trumps all other contenders (7-8,11-12, 15-16).

**Test #5: DB 8k Read-Write (Commit)**

This test shows read-write throughput with commit on the db volume with 8k block size. This is probably the most important metric to show Perforce database volume performance.

Higher is better

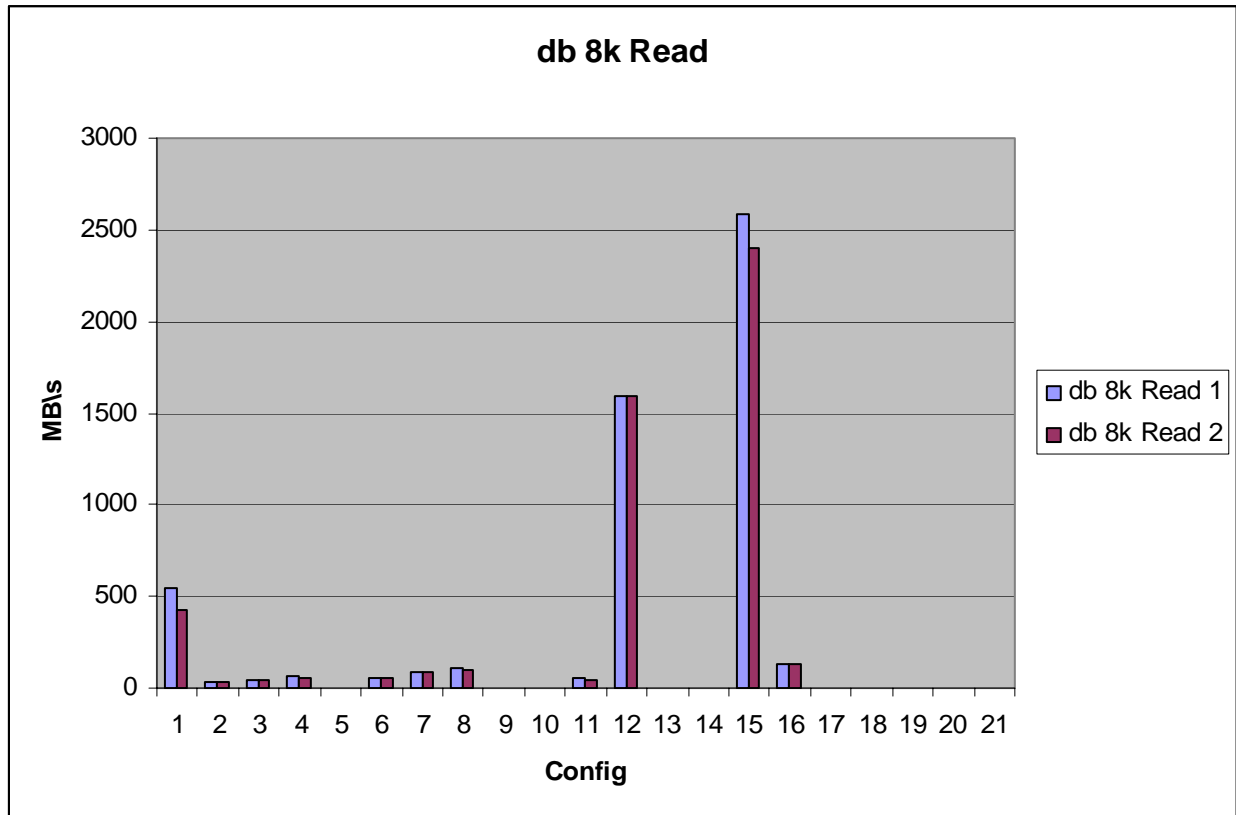


Analysis: Config 12 wins outright and all of the Linux 64-bit configs demonstrate the best throughput.

### Test #6: DB 8k Read

This test shows read throughput on the db volume with 8k block size.

Higher is better

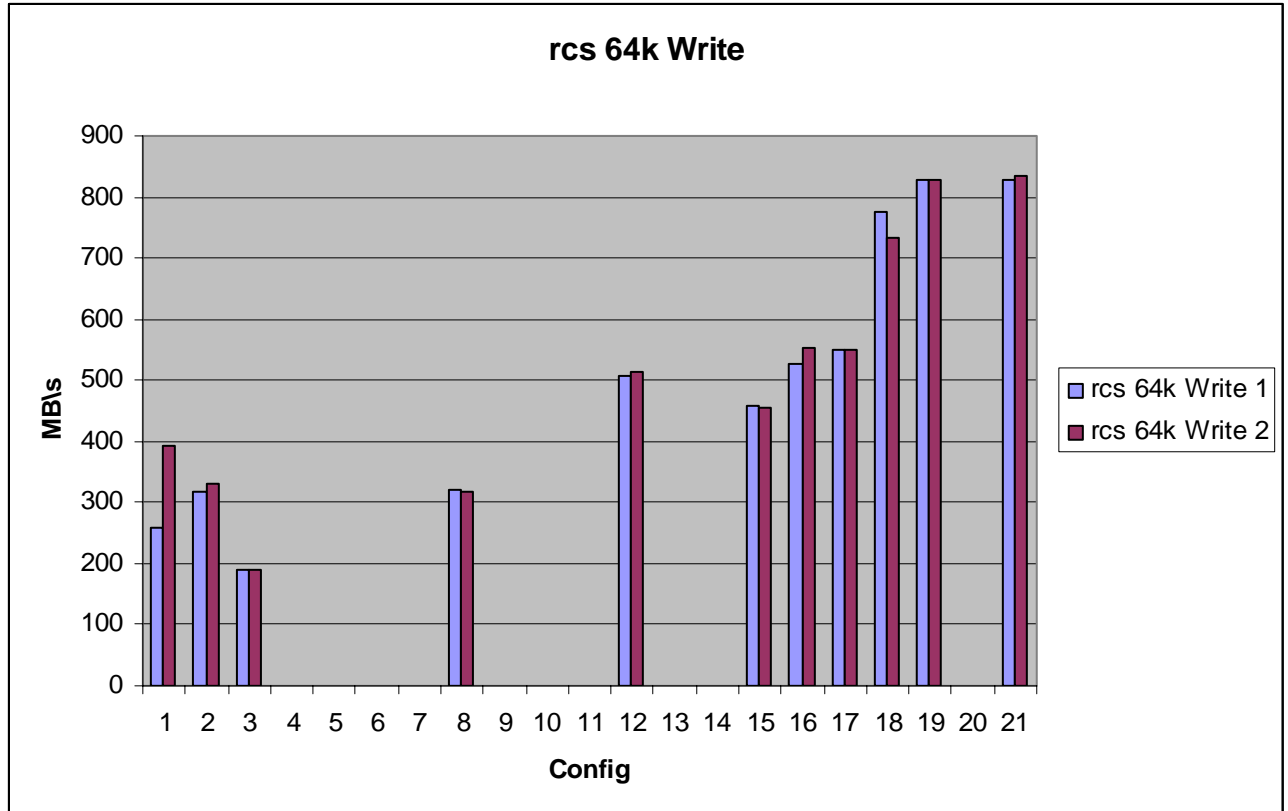


Analysis: Linux 64-bit configs 12 and 15 are the top performers here.

### Test #7: RCS 64k Write

This test shows write throughput on the SAN versioned files volume with 64k block size. This indicates the throughput one might expect on a submit upload.

Higher is better

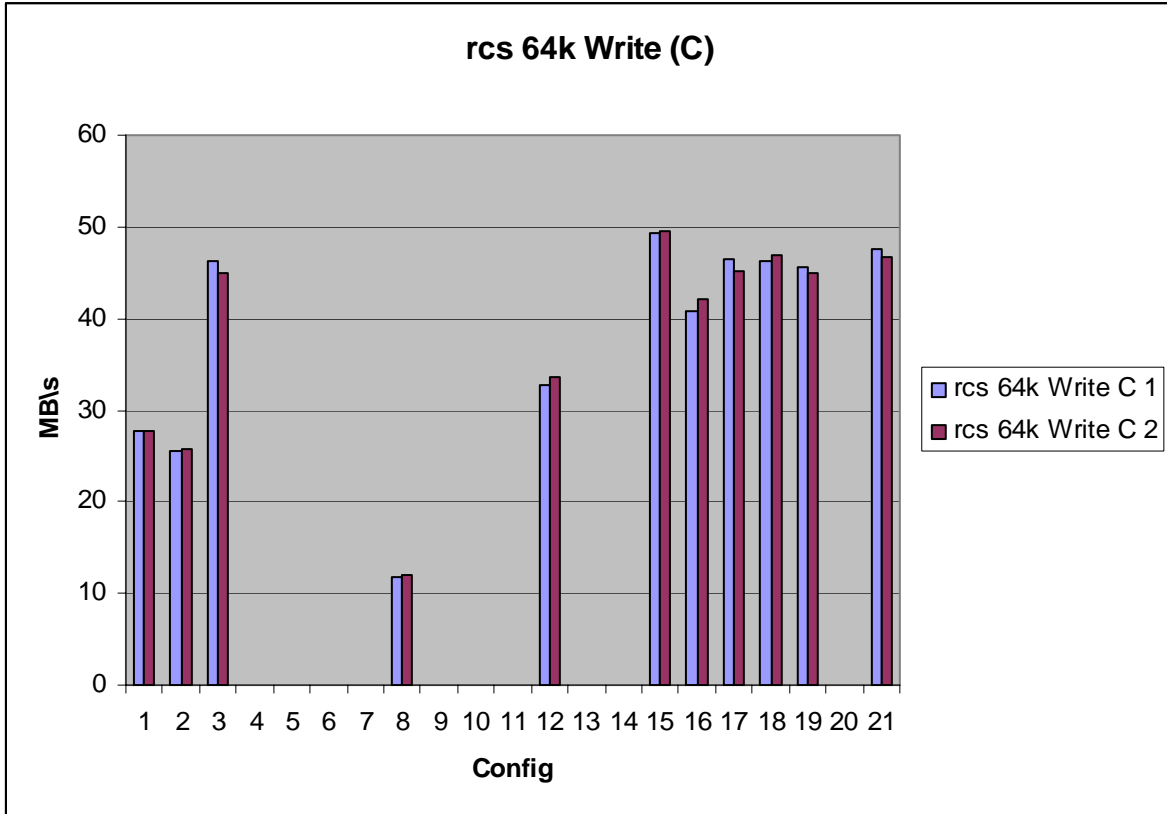


Analysis: The 64-bit Linux configs win again. And XFS (18-19, 21) demonstrates much better write throughput than ReiserFS (12, 15-17).

**Test #8: RCS 64k Write (Commit)**

This test shows write with commit throughput on the versioned files volume with 64k block size.

Higher is better

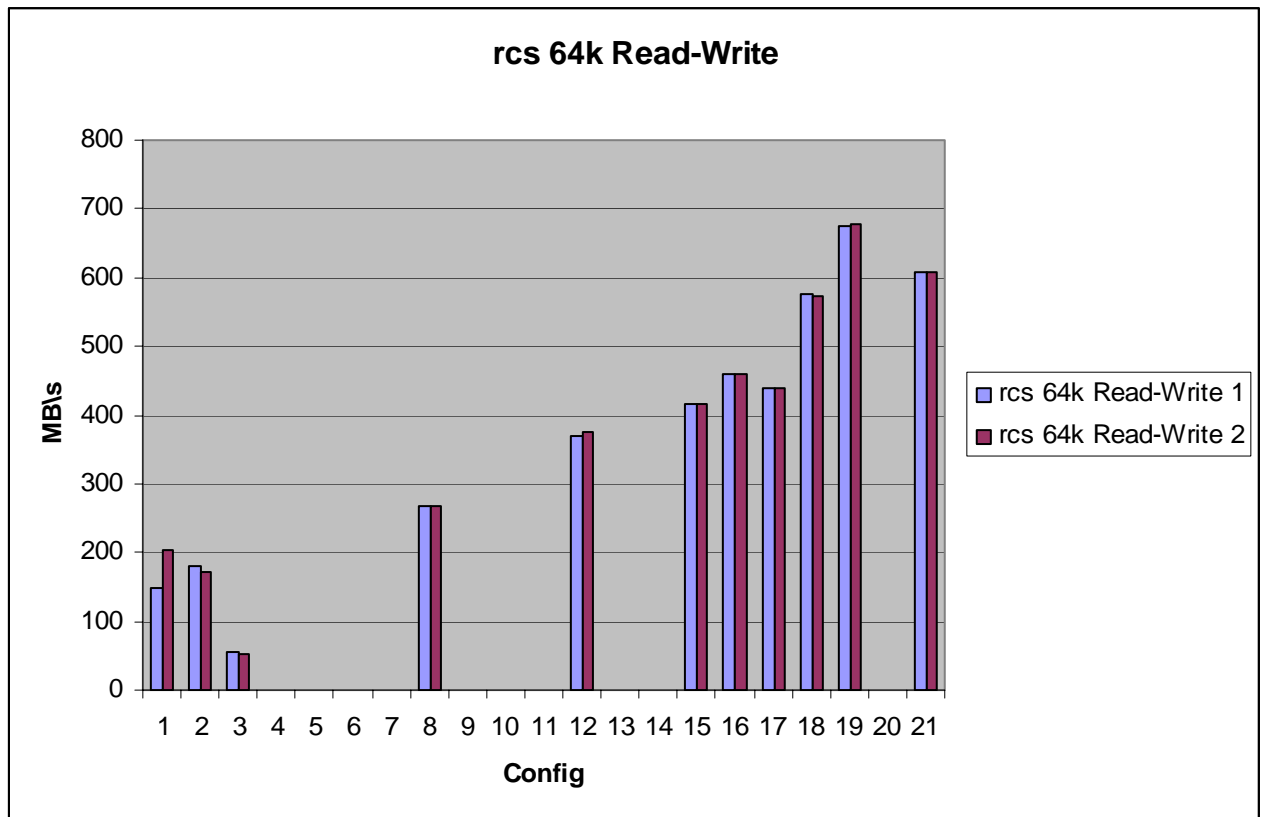


Analysis: Surprisingly, config 3 is a high performer here (Linux 2.4 kernel, 32-bit, no readahead cache on SAN). The Linux 64-bit configs still hold the top spots.

### Test #9: RCS 64k Read-Write

This test shows read-write throughput on the versioned files volume with 64k block size.

Higher is better

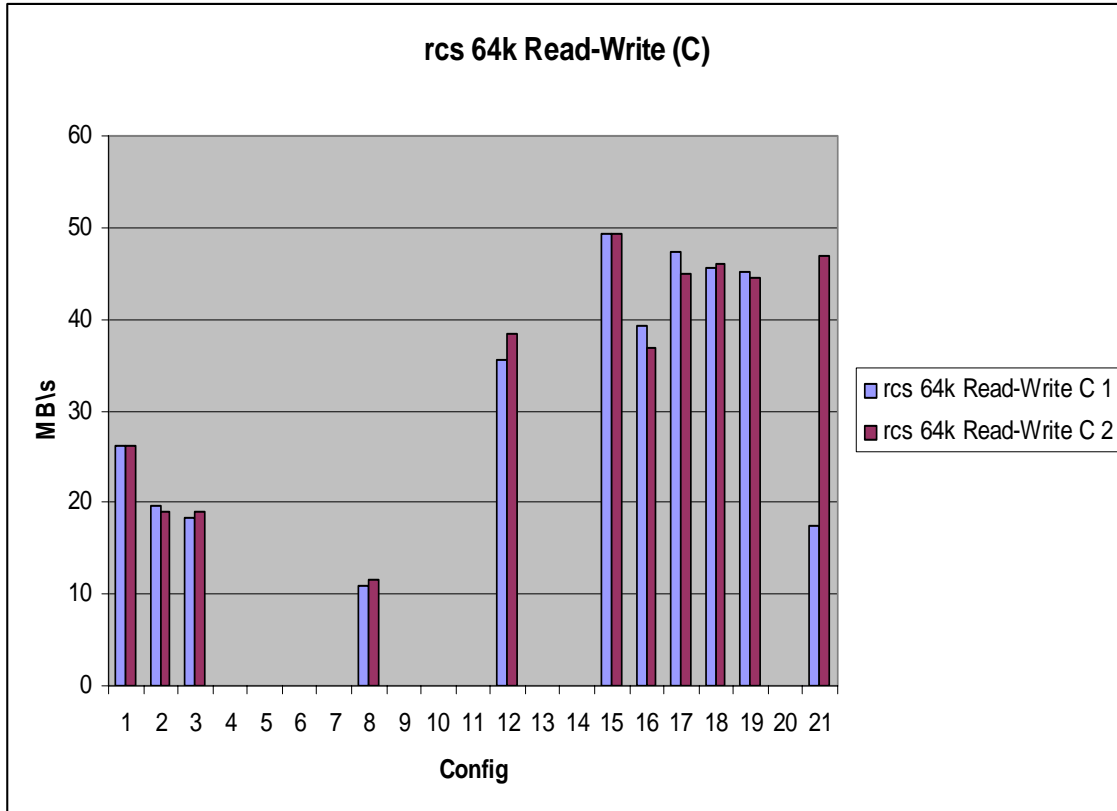


Analysis: XFS again outperforms ReiserFS. Both XFS and ReiserFS tended to be over 2x faster than NTFS or ext3.

### Test #10: RCS 64k Read-Write (Commit)

This test shows read-write throughput with commit on the versioned files volume with 64k block size.

Higher is better

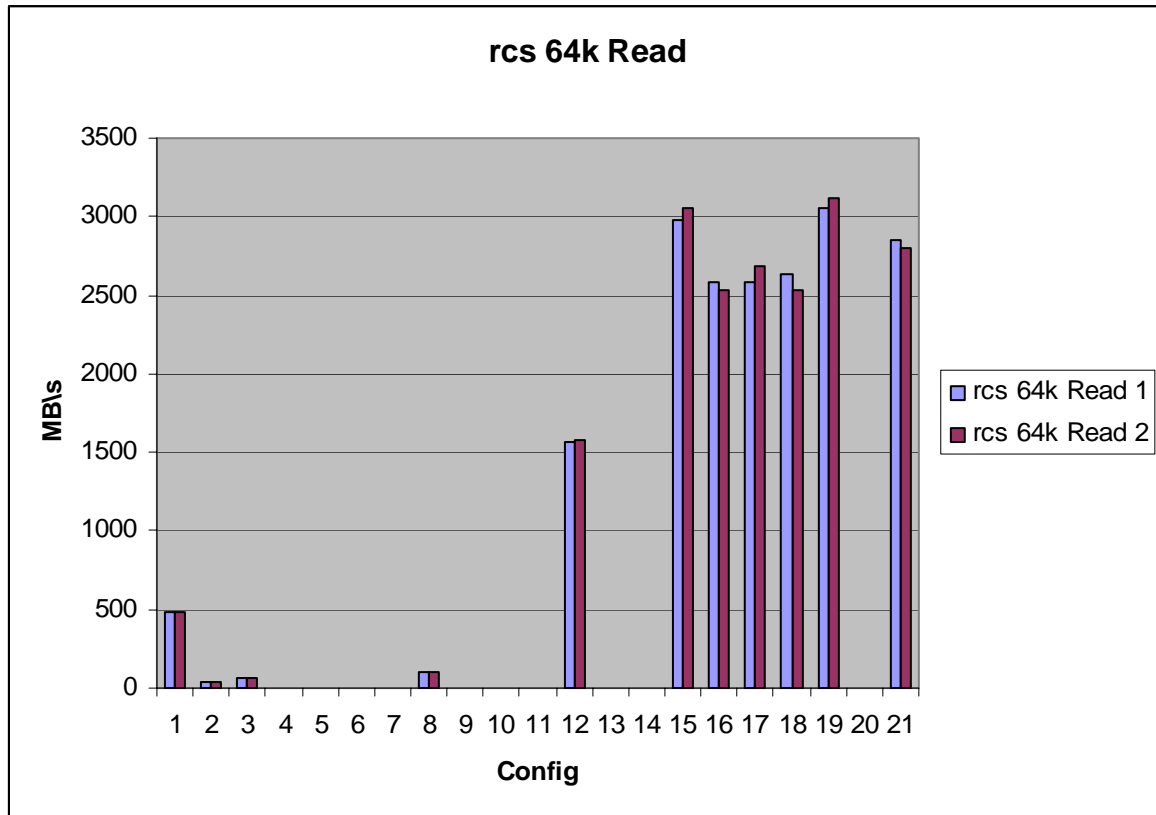


Analysis: Once again, the Linux 64-bit configs win. Surprisingly, we haven't seen much difference in performance between the 10k and 15k drives on this metric. The filesystems and RAID levels seem to have a larger effect on this particular performance metric than the drive speed.

### Test #11: RCS 64k Read

This test shows read throughput on the versioned files volume with 64k block size.

Higher is better

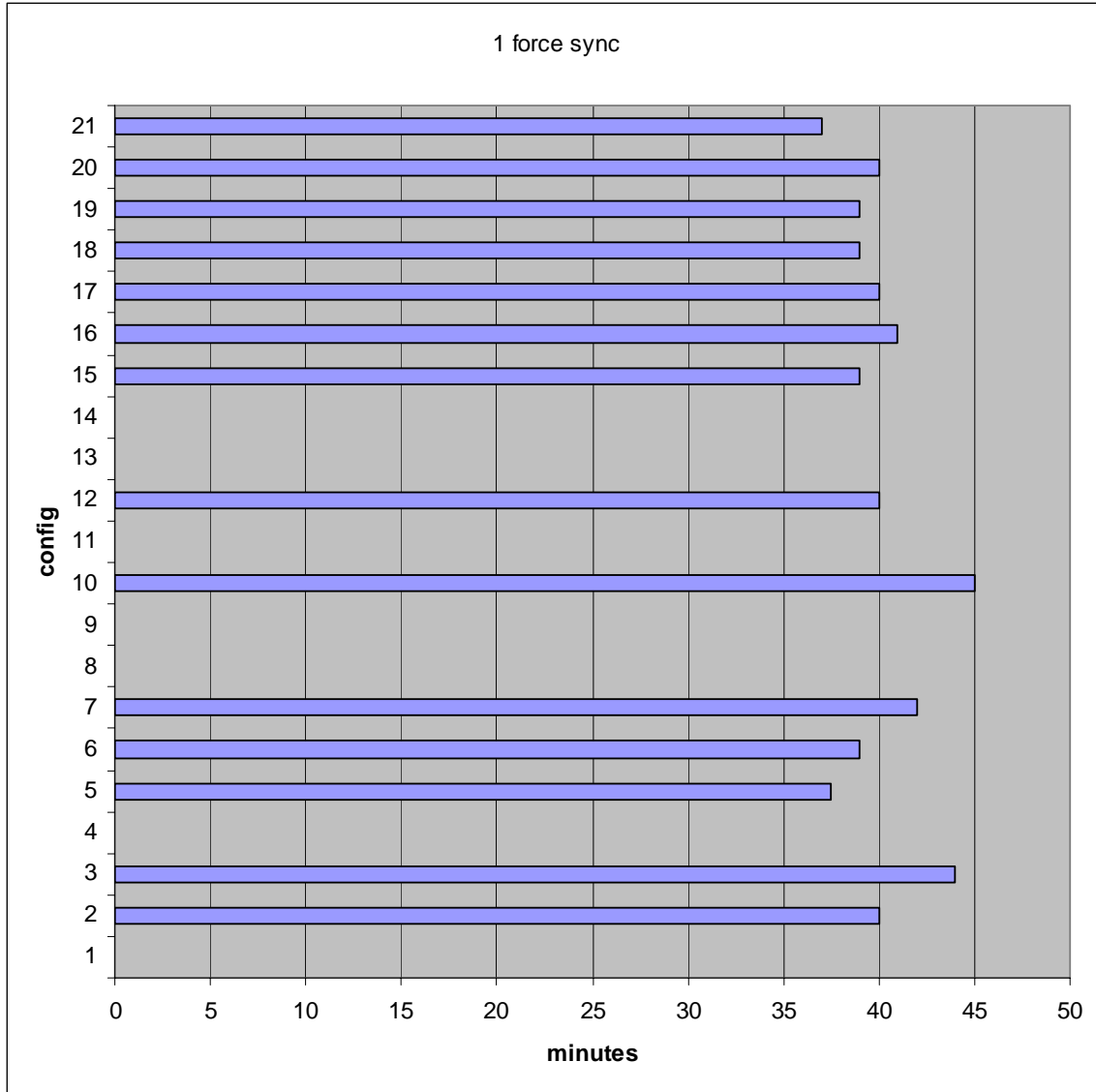


Analysis: Yes, 64-bit Linux crushes all contenders.

**Test #12: Single forced sync of 39 GB of Godfather P4 data**

This test shows the average elapsed time for a single forced sync of 39 GB of Godfather Perforce data (179,000+ files).

Lower is better

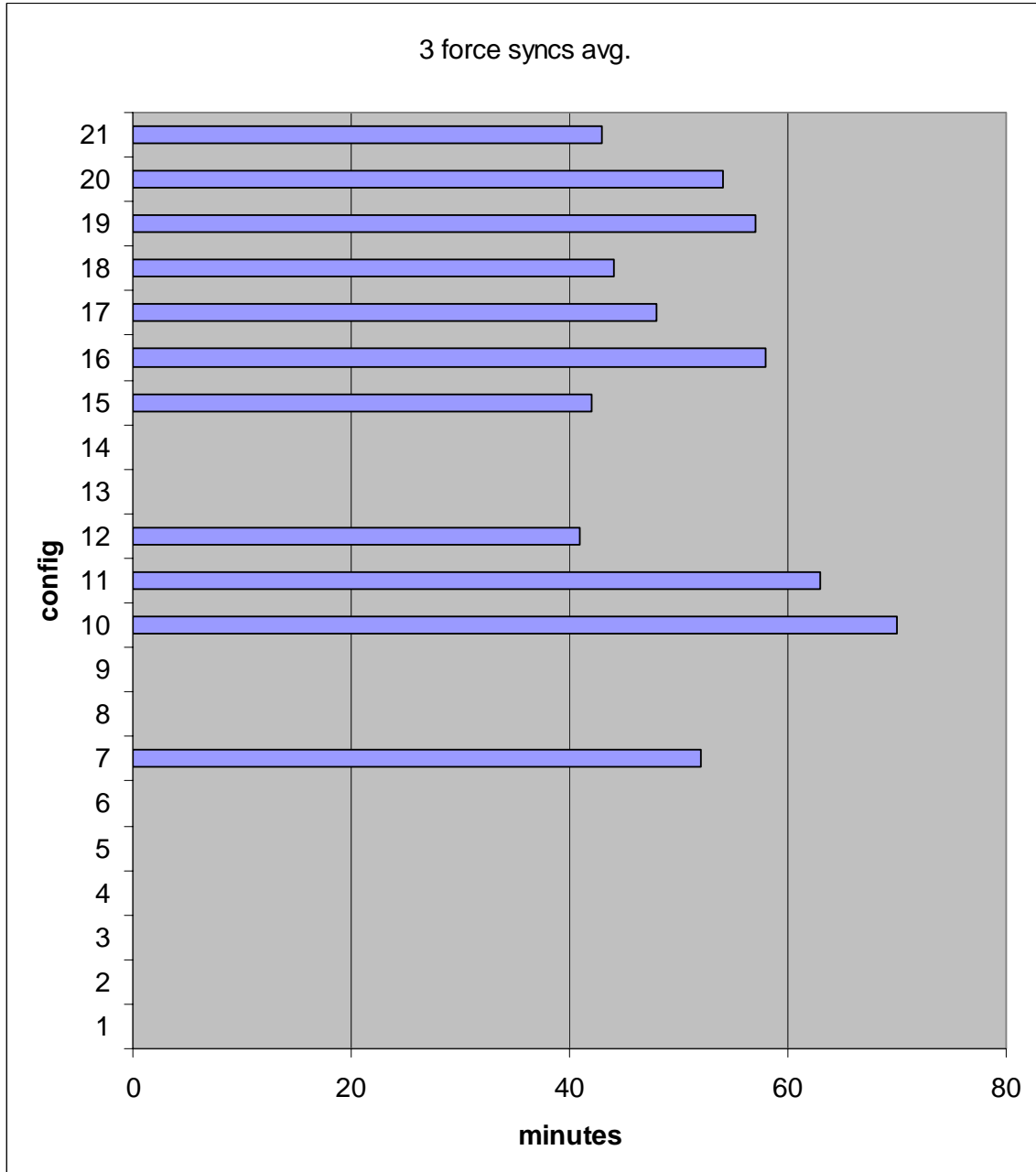


Analysis: The Windows clients averaged around 40 minutes for a single forced sync to pull down 39 GB to the desktop. This metric is more important as a baseline to see how much performance degrades when running concurrent syncs. Windows 64-bit (10) was actually the slowest here.

**Test #13: Average elapsed time for three concurrent forced syncs**

This test shows the average elapsed time for three concurrent forced syncs of 39 GB of Godfather Perforce data (179,000+ files). This was a key metric for our project since it was a close approximation of the issues we faced in production. Sync starts were staggered by 5 minutes.

Lower is better

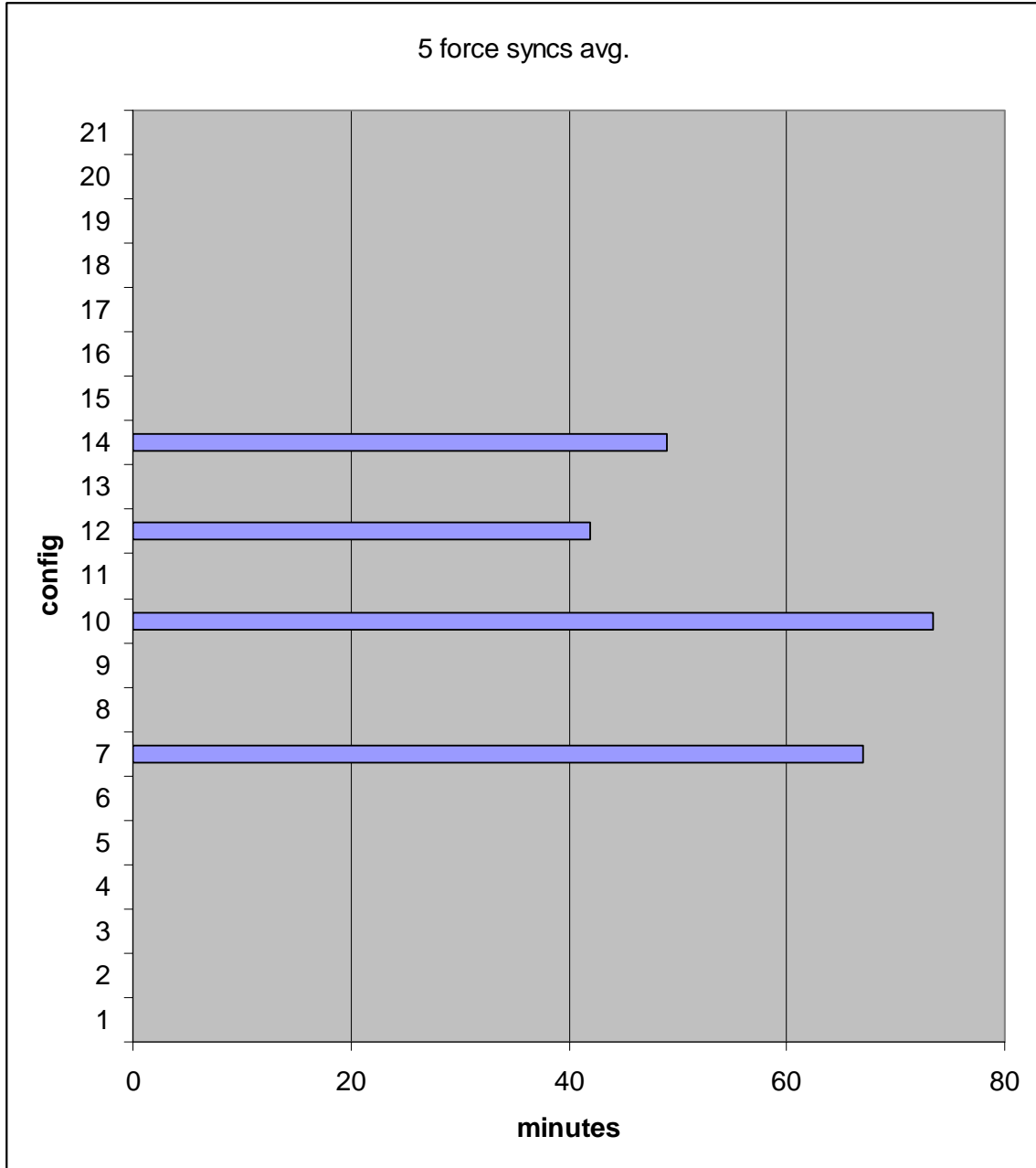


Analysis: Windows 2003 64-bit (10) on the exact same hardware, memory, and RAID configuration as 64-bit Linux (9), shows significant degradation, and the only difference is the OS and filesystem. This metric really shows how Linux makes better use of available hardware resources than Windows does and performs better under load.

**Test #14: Average elapsed time for five concurrent forced syncs**

This test shows the average elapsed time for five concurrent forced syncs of 39 GB of Godfather Perforce data (179,000+ files).

Lower is better

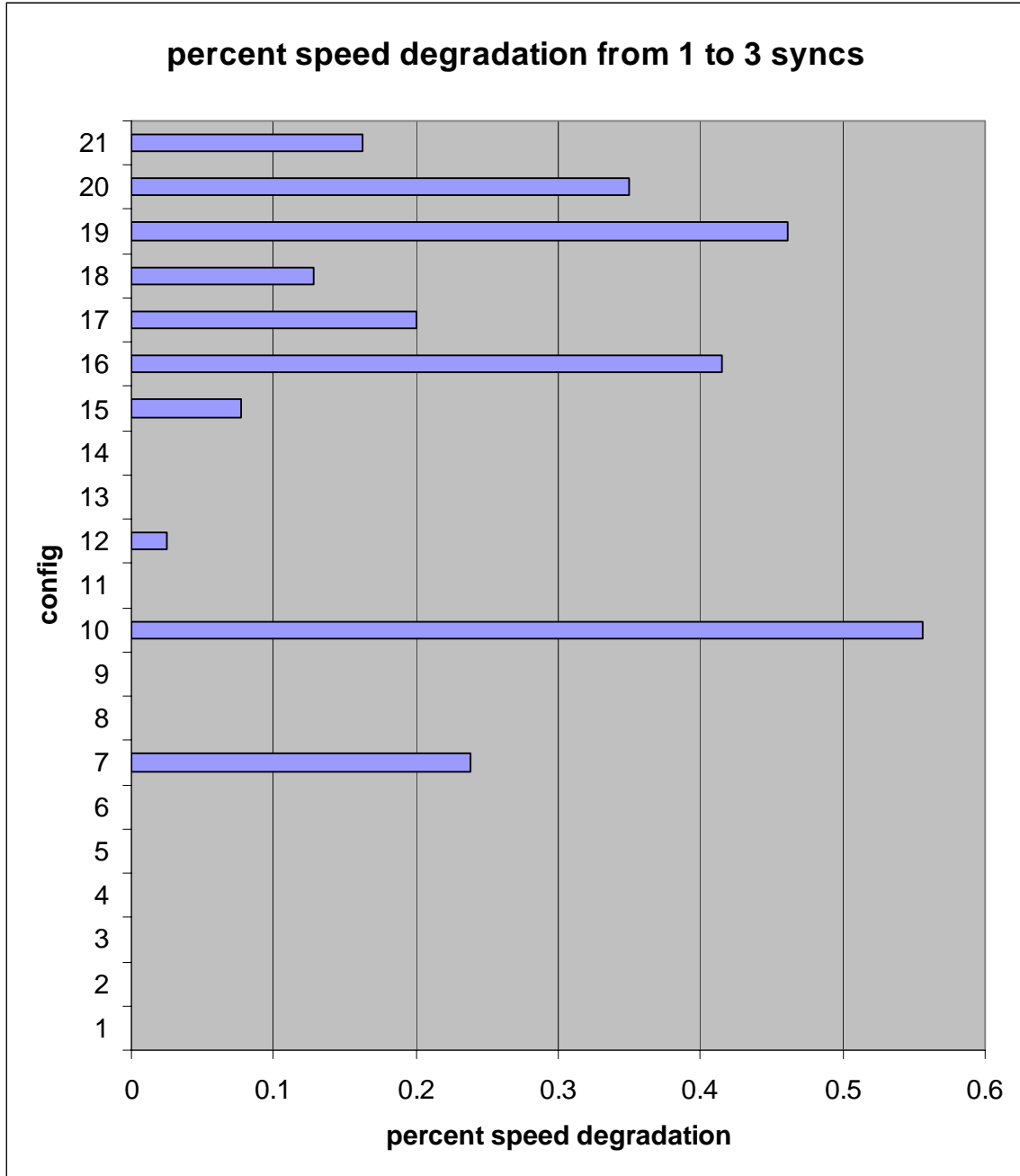


Analysis: Config 12 (Linux 64-bit) still shows virtually no degradation, while the Windows 64-bit server is nearly twice as slow from one sync to five syncs.

**Test #15: Percent increase in sync elapsed time under load from 1 to 3 forced syncs**

This test shows the percent increase in elapsed sync time from running 1 forced sync to running 3 concurrent forced syncs.

Lower is better

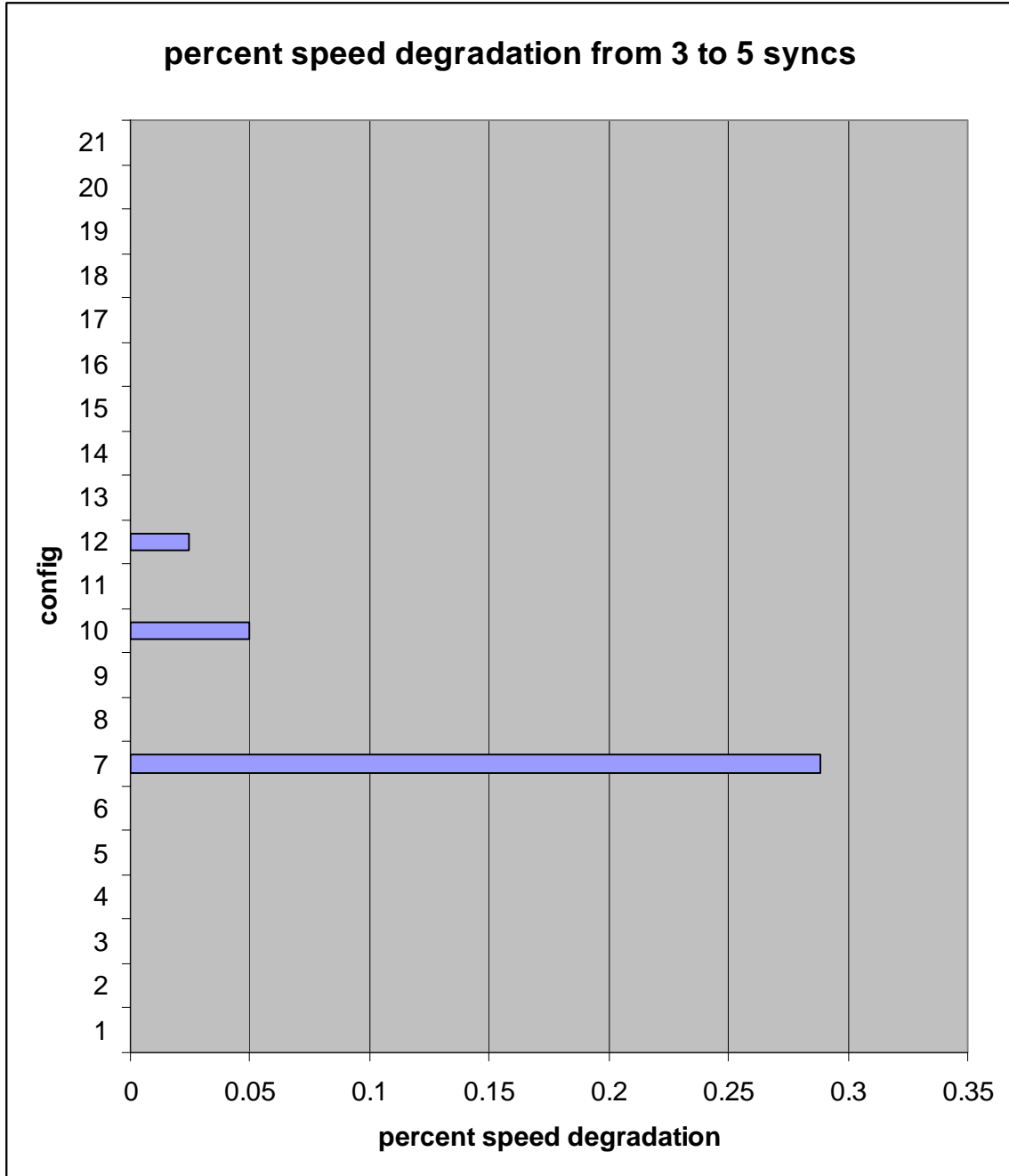


Analysis: Config 12 (Linux 64-bit) wins here, with virtually no degradation.

**Test #16: Percent increase in sync elapsed time under load from 3 to 5 forced syncs**

This test shows the percent increase in elapsed sync time from running 3 concurrent forced syncs to running 5 concurrent forced syncs.

Lower is better

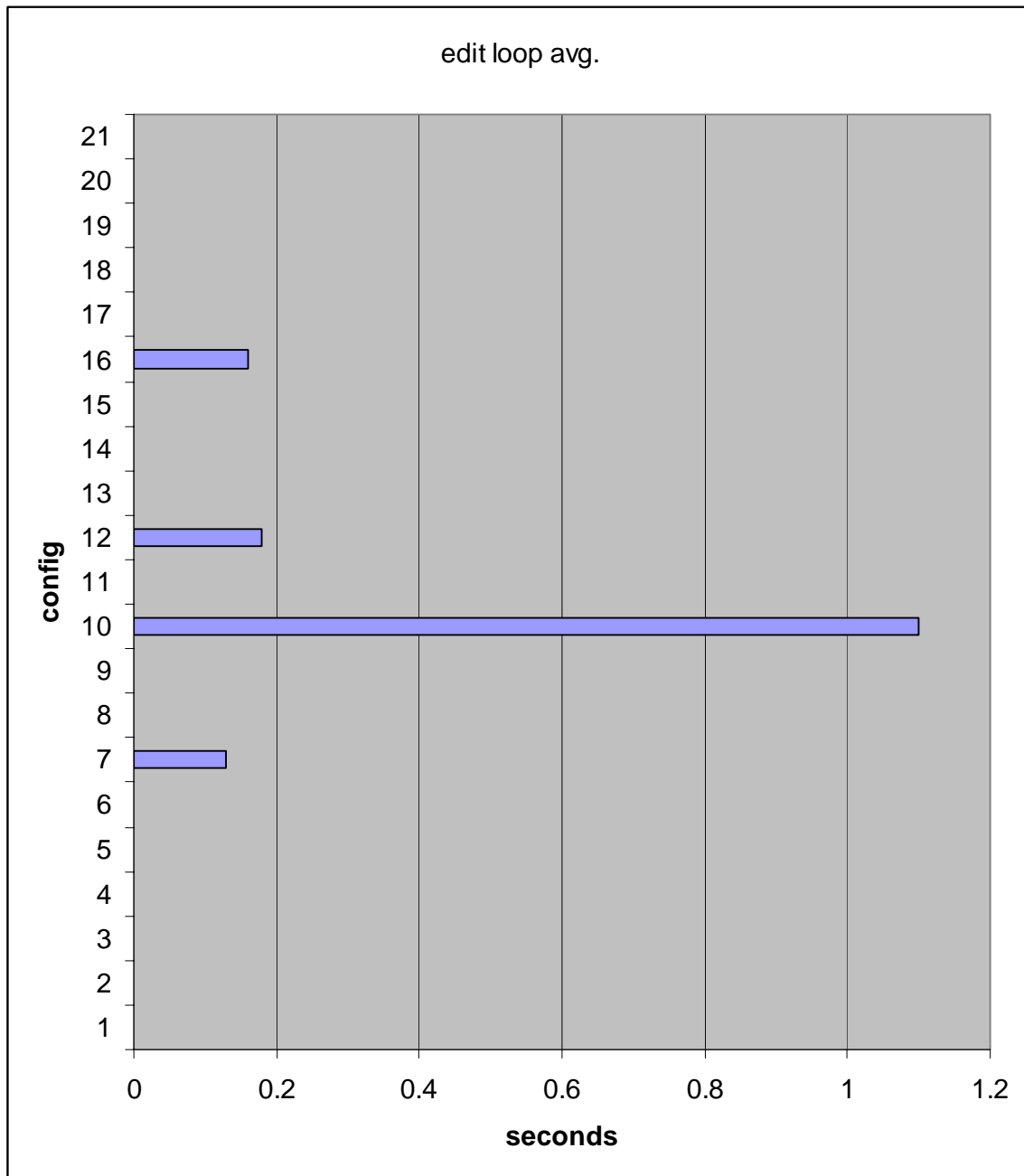


Analysis: Windows 64-bit (10) degrades twice as much as Linux 64-bit (12).

**Test #17: Average edit loop for 176 files**

This test shows the average elapsed time for a 'p4 edit' of 176 files

Lower is better



Analysis: Windows 64-bit (10) is 5-8x slower on opening files for edit than the 64-bit Linux configs.

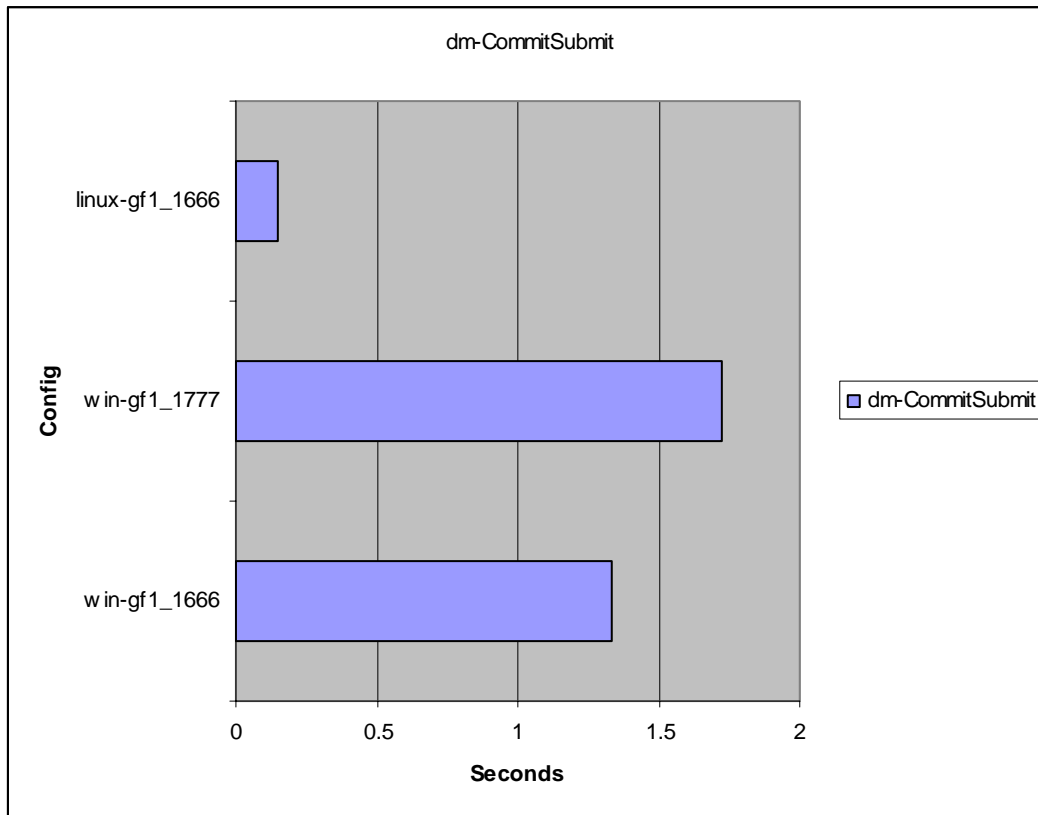
### Production Results

Here's how the production server config we came up with (Config 12), stacked up against the Windows production Godfather box (Config 2). These metrics were taken from actual user operations a week before the upgrade and a week after. This was an opportunity to see how well our new config performed in production instead of just in a test environment. Due to the 2 GB memory limitation on Win32, we had to have two separate user-facing Perforce instances (1666 & 1777) pointing to the same databases. Linux doesn't have that memory restriction, so we were able to just use a single instance (1666).

### Production Metric #1: dm-CommitSubmit Elapsed Time

This metric shows the average elapsed time the Perforce databases were locked when submitting a changelist. This shows up as a hang to users.

lower is better

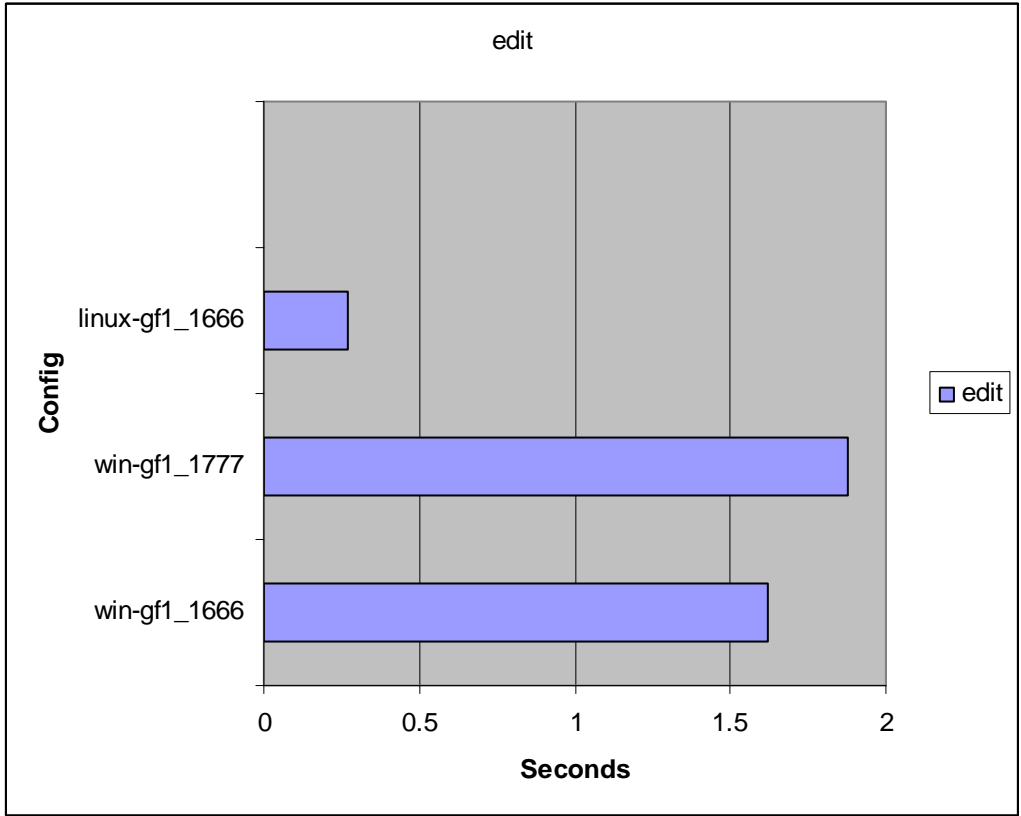


Analysis: The Linux 64-bit machine with upgraded hardware was over 10x faster than Win32 on database commits during submits.

**Production Metric #2: p4 edit Elapsed Time**

This metric shows how the average elapsed time to open a file for edit in Perforce. This metric is visible to users since the Perforce databases are locked when doing a 'p4 edit' so users can notice a hang in their operations if this takes a while to run.

lower is better

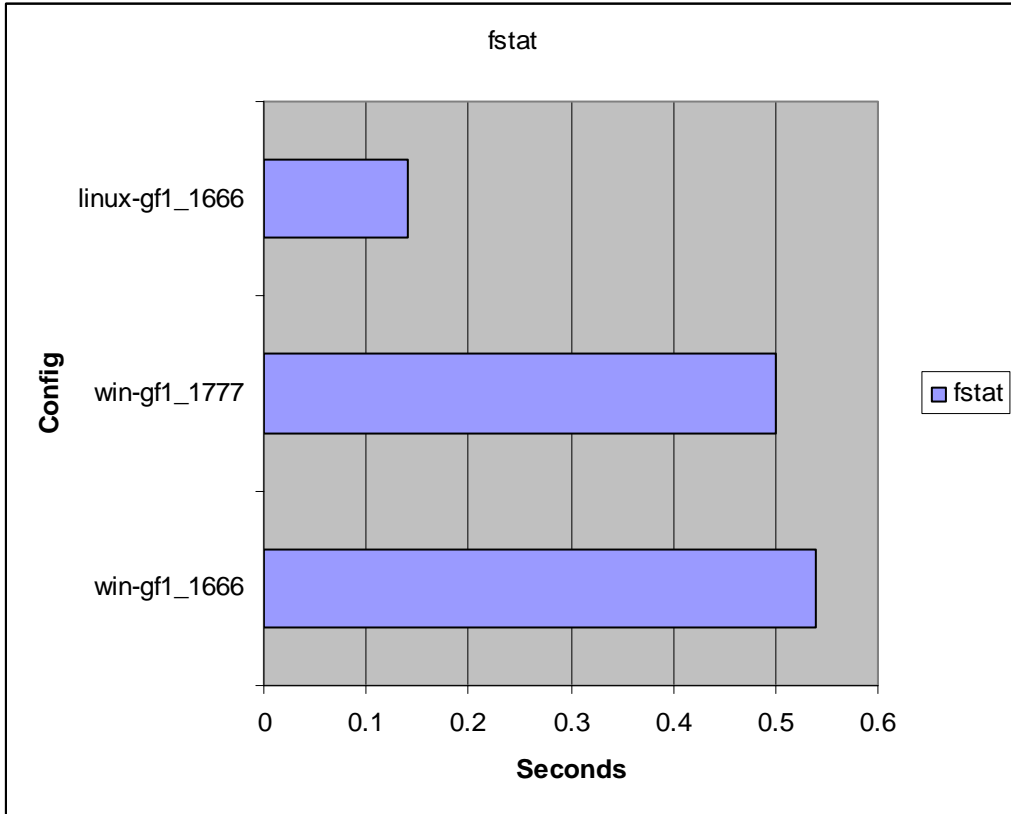


Analysis: The Linux 64-bit machine with upgraded hardware was over 8x faster than Win32 on opening files for edit.

### Production Metric #3: p4 fstat Elapsed Time

This metric shows how the average elapsed time for getting file information (such as when browsing a depot). This is a very visible metric to users since it affects how quickly they can browse in Perforce.

lower is better

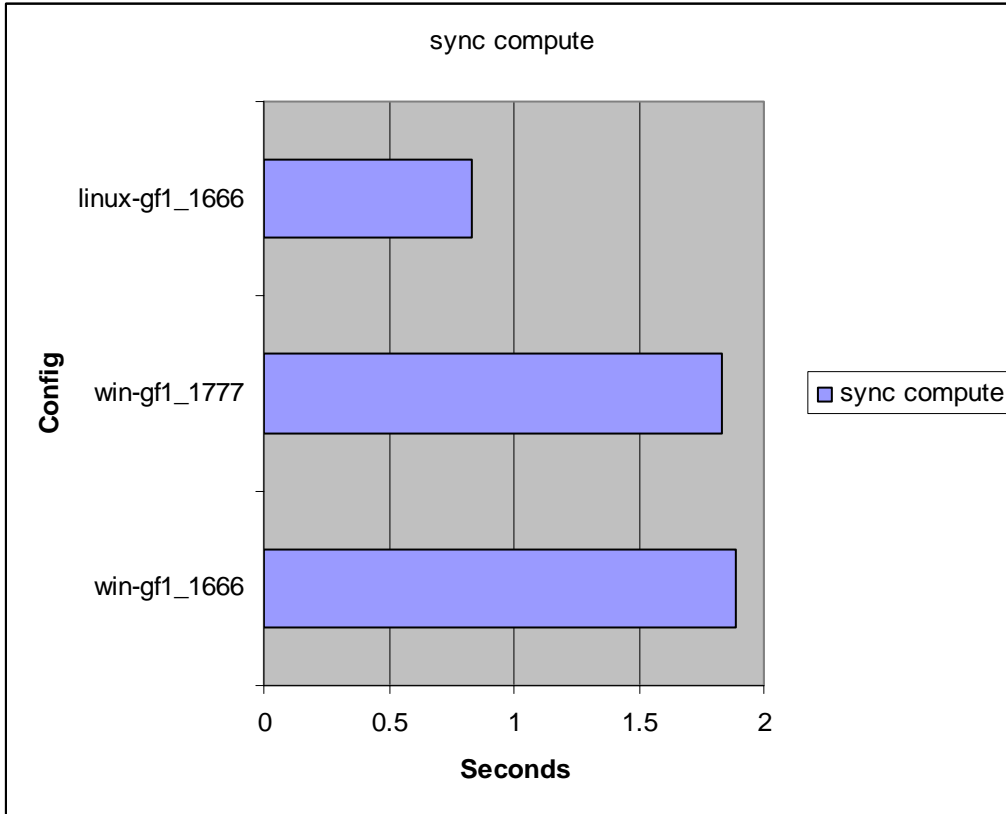


Analysis: The Linux 64-bit machine with upgraded hardware was around 4x faster than Win32 on getting file information.

**Production Metric #4: p4 sync Compute Phase Elapsed Time**

This metric shows how the quickly the server can compute the list of files to deliver to a client when they do a sync. The compute phase of a sync locks the databases so other Perforce operations will hang if this operation takes a long time to complete.

lower is better



Analysis: The Linux 64-bit machine with upgraded hardware was over 2x faster than Win32 at calculating the list of files to deliver during a sync.

## Performance Improvements

This new server platform solved our stability issues. Each thread gets its own memory space, the POSIX file descriptor limit is much higher and can be manually tweaked even further, and the scalability is significantly improved.

Linux sync speed was **25% better under concurrent force sync load** on the exact same hardware as Windows. On Windows, there was some database performance hit while running multiple concurrent forced syncs. On Linux, **database operations were 5-10x faster and showed virtually zero speed degradation** during multiple concurrent forced syncs. Once again, this is on the exact same hardware.

Later, we further improved the concurrent sync speed by having proxies for the build farms. Concurrent forced syncs can cause I/O wait and sync delays, so offloading that churn to separate disks helped both our build farm and the teams.

## Case-Sensitivity

There is a 'case-insensitivity' flag that we have tested and are using in production. This allows the Perforce server to run on case-sensitive OS's but function as a case-insensitive server. There is no performance hit here like there was with the old 'checkcase' triggers. The case issues were one of the primary roadblocks to moving our teams to Unix-based servers in the past. We have run eight Perforce servers with this case-insensitivity flag for nearly two years without any issues. See 'p4 help undoc' for more info.

## Production

We successfully migrated our first production server to our new standard server platform on September 10, 2005. The database performance is 5-10x faster than our old config. Sync performance under load degrades only 5% total instead of the previous 450%. The memory and scalability limitations on Win32 which caused service interrupts are gone.

## Scalability

The new server platform has scaled well for teams of over 1000 users, even with an average clientspec size of 500,000+ files. This platform could potentially scale much larger, but we don't have any teams over 1,000 users (yet).

## Lessons Learned

We learned several lessons from Perforce and from our own testing.

- Even if you are on Windows, keep your Perforce database files on their own disks. This keeps sync load separate from database operations. This is one of the main things you can do to improve your Perforce performance, no matter what OS you are running.
  - See the "EA Perforce Server Standard Configuration" appendix in the online version of this whitepaper for more detail on setting up a high-performance Perforce server.
- Windows has historically had certain scalability limitations which make it difficult for large sites to use Windows Perforce servers.
- The majority of Perforce's large customers (1,000+ users) run a Unix-based OS for their Perforce servers.
- Linux is much more efficient in its use of available hardware than Windows.
- The XFS filesystem outperforms ext3 and even ReiserFS for data syncs from the repository volume.

## Appendix A: EA Perforce Standard Server Configuration

### Overview

This appendix is an overview of our standard next generation server platform configuration we use at EA Redwood Shores for servers supporting between 200-1000 users as of March 2007.

### Hardware

Server: Dell 2950

RAM: 16 GB

CPU: Dual-core or Quad-core Xeon EM64T

Drives: 6 x 146GB internal 15k RPM drives

OS: RedHat AS 4.0 EM64T

Versioned files storage: 15k drives on SAN (StorageTEK or NetApp SAN)

QLogic FibreChannel card(s) (for SAN)

### Hardware Selection

As we are not usually bottlenecked by the CPU in a Perforce server, the most critical hardware components are amount of RAM and speed of the disk drives. In a Linux environment all available RAM is used to help performance: any unused RAM is used by the kernel to provide buffers and cache of the file systems. In addition to 15k RPM drives, 15k RPM Fibre Channel drives are used in the SAN to allow 20% faster throughput of versioned data.

### RAID Configuration

Server component	RAID level	Min. # disks	disk location	filesystem
OS	1	2 (shared)	internal	ext3
checkpoints/journal/log	1	2 (shared)	internal	XFS
p4 databases	10	4	internal	XFS
versioned files	5	3	SAN	XFS

### RAID Selection

The testing has shown that the RAID levels chosen during the initial setup of the server have a significant effect on the performance of Perforce transactions. A critical part of the configuration is to separate the Perforce databases onto their own physical disks so that heavy disk activity from other operations does not starve the Perforce databases out of disk I/O. The following is the suggested RAID configurations for the system mentioned above:

1. RAID1 (Disks 1 and 2)
  - a. LUN0
    - i. OS and checkpoints LUN, standard stripe size for OS (64k)
  - b. LUN1
    - i. P4 Journal/Log LUN (remaining space), stripe size set to 4k
2. RAID10 (Disks 3 – 6)
  - a. LUN0
    - i. P4 DB LUN (all space), stripe size set to 8k
3. RAID5 (SAN Disks 1-3)
  - a. LUN0
    - i. P4 RCS LUN (all space), standard stripe size for RCS (64k)

### OS Selection

While the new hardware configuration certainly helps performance another critical performance improvement was realized through the use of the Linux operating system. Our requirements were that the Linux version had to support: Dell hardware; Intel EM64T Xeon processors for increased addressable memory support; the Linux 2.6 kernel due to improvements in its handling of virtual memory, CPU (threading and scheduling), and I/O performance; and the XFS file system as suggested by Perforce. Both SuSE Enterprise Linux and RedHat are able to be configured to support the above requirements, however currently the EA standard is RedHat.

## Filesystem Selection

Our testing results proved that in a Perforce environment the XFS file system outperformed ext3 and ReiserFS. Unfortunately, RedHat's default enterprise kernel doesn't support XFS out of the box and therefore needs to be recompiled with the applicable modules.

1. Our testing results proved that in a Perforce environment the XFS file system outperformed ext3 and reiserfs. Unfortunately, RedHat's default enterprise kernel doesn't support XFS out of the box and therefore needs to be recompiled with the applicable modules. If you want to customize the kernel further for your environment, please ensure to compile "XFS" and "Probe all SCSI LUNs" into the kernel.
2. As with the default RedHat enterprise kernel, XFS tools (i.e. mkfs.xfs) are missing from the RedHat distribution. Therefore these tools must be built from the SGI source.
3. XFS Volume Creation (assuming internal RAID config as stated above and one SAN LUN)
  - a. Once all of the partitions have been created, create XFS file systems with 64meg logs by issuing the following commands:
    - i. Run: `mkfs.xfs -l size=64m /dev/sdb1`
    - ii. Run: `mkfs.xfs -l size=64m /dev/sdc1`
    - iii. Run: `mkfs.xfs -l size=64m /dev/sdd1`

**Tuning Linux OS Parameters** - During the testing of various system configurations Perforce gave us some OS level customizations which should be made for the best performance possible.

1. This parameter is the optimal TCP Window to use for Windows-based clients accessing a Linux Perforce server. It also helps with latent connections.
  - i. Edit the `/etc/rc.local` file and append the following lines
    1. `sysctl -w net.ipv4.tcp_wmem="4096 32768 131072"`
    2. `echo 10 > /proc/sys/vm/swappiness`
2. 'MALLOC\_MMAP\_MAX' and 'MALLOC\_TRIM\_THRESHOLD' tweaks should be added to the Perforce environment startup script. These tweaks fix some garbage collection issues on Linux which have caused sync problems in a production EA environment.
  - i. `MALLOC_MMAP_MAX=0`
  - ii. `MALLOC_TRIM_THRESHOLD=-1`
  - iii. `export MALLOC_MMAP_MAX MALLOC_TRIM_THRESHOLD`
3. If you don't run any scripts which require file access time, then you can disable atime by running 'noatime' on the appropriate Perforce volumes before mounting them.

## Case-Insensitivity

If you desire to run your Perforce server on Linux in case-insensitive mode, it is recommended that you protect yourself against accidentally running p4d without the case-insensitivity flag, as doing so could lead to database corruption. We do this by renaming the 'p4d' executable to P4D\_EXE and creating a shell script called 'p4d' which looks like this:

```
/usr/local/perforce/P4D_EXE -C1 $*
```

This wrapper script should be used instead of the actual Perforce server binary.

## Perforce Server Database Configuration

By default, the Perforce versioned files will automatically be put in the P4ROOT directory on submit (/p4db where the Perforce databases live.) With the above configuration, we want to ensure that any versioned files get put on the /p4rcs volume instead of the /p4db volume. To ensure this:

1. Update any existing depots to point to /p4rcs
  - a. 'p4 depots' to list all depots.
  - b. 'p4 depot depot' and then change the 'Map:' field for depot 'depot' to '/p4rcs/depot/...'. Repeat this step for all depots.