Comparing Different File Systems' NFS Performance
A Cluster File System and a Couple of NAS Servers Thrown In

Cary Whitney[1]

National Energy Research Scientific Center (NERSC)[2][3]
whitney@nersc.gov
http://pdsf.nersc.gov/

Abstract

With the continual growth of the Parallel Distributed System Facility (PDSF) Linux cluster. we have started to explore different storage solutions to meet our users' needs. This search started by increasing the hard drive size in our current systems, then by looking at expensive Network Attached Storage (NAS) solutions when performance started to be an issue. Currently we have started to investigate how to use our existing hardware more efficiently, because of the continual drivers and software improvements for Linux. Our Questions: Could our low cost solutions stand up to the demands of a clustered environment? Could they offer comparable performance and more important could they scale to large numbers of connections? We hope to answer these questions here.

1. Introduction

Since PDSF is a production cluster, the setup of these tests had to be scheduled around the production jobs. This file system work is in support of our user groups.

PDSF is a cluster of about 200 dual processor systems. Most systems are connected to the network via fast Ethernet, but there is a class of machines which we call 'high-bandwidth nodes' which have a GigE connection. All of the cluster machines made up the testbed for this work.

Most of the science done on PDSF involves high energy physics, whose problems normally have a dataset that is 1-5 GB in size with file sizes ranging from 80-200 MB. This data resides on what we term 'disk vaults'. PDSF currently has about 35 disk vaults, with an average exported file system size of 500 GB. We are looking at having to increase this storage by 50 TB for the coming year.

We first tried to solve our growing data requirement by increasing the size of the hard drives. Instead of 0.5 TB we would have 1 TB for the file system. This proved to be unworkable since more data could be stored on the node than the node was able to serve to the researchers.

We then thought about mirroring NFS servers. This would increase

1 Many thanks to all those who answered my questions: Matthew Andrews (Mass Storage), Shane Canon (PDSF), Tom Davis (ASG) and for generally helping out: Tom Langley (PDSF), Iwona Sekrejda (USG).

2 This work was supported by the Director, Office of Science, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy under contract number DE-AC03-76SF00098.

3 This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy.

performance of accessing the data but at the cost of doubling our storage requirements. This could not be done because of the lack of funds and the users' desire to use 'all' the disk space.

Next we looked at a NAS solution. These provided the scalability that we wanted and looked to be a good fit, but the users then changed the amount of storage required for the year, thus placing the NAS out of financial reach.

Finally we started to look more at what we could do to leverage current hardware and software. This became more feasible with improvements in drivers, more file systems choices and time to evaluate these options.

To start with, we limited our tests to the following parameters:

- Users will have largish files (in relation to Linux's 2 GB limit.)
- Users will mostly read/write these files sequentially.
- Jobs will be bursty. Many of the same type of jobs will be executing at the same time against the same datasets.

Thus, we chose to look at a large number of clients accessing a single NFS partition, sequentially reading and writing individual files.

2. Overview of the conclusion

We have concluded that the XFS file system offered the best solution for our needs. This was because of its performance and support for Data Management API (DMAPI). But we should also mention ReiserFS for making a strong showing as an alternative.

As this paper shows, there are really no overall winners or losers. The application has a lot to do with performance, and a study of the I/O characteristics of the applications should be performed before a storage solution is considered.

Both the NAS's and the cluster file system performed very well, but in our situation we do not have the money to fully implement either system. However, we have purchased a smaller NAS to serve as our 'home' and cluster-critical server. Our goal has always been to provide an economical clustering environment for our users, and this guided our testing.

Below you will find our results so far. Hopefully, testing will continue when new developments are made available and those results will also be shared when completed. All related work will be linked off the PDSF web site at http://pdsf.nersc.gov.

3. Testing method and setup

PSDF uses two connection methods for its compute nodes. The GigE-connected machines were used for tests under 40 clients, while 40 and above clients also used the fast Ethernet nodes. All nodes were running a Linux 2.2.19 kernel. The networking switches used were all Extreme 7i's with Cisco 3548s connecting the fast Ethernet nodes. The Cisco switches had a GigE uplink to a head 7i.

Iozone[IOZ] was selected as the benchmark program. Scripts were written to log into the client nodes and submit an 'at' job to start the test. Since ntpd was running cluster-wide, this was judged to be a good synchronization mechanism.

One of two problems encountered was client-side caching. If Iozone was run first with a write test, which created the temporary file to be used for the read test, the file would then be cached on the client side creating misleading results. This problem was overcome by using the -i flag and we specified each test to run in this mode. First a write test was run specifying an output file and requesting it not to be deleted. When all the write tests were done, the read tests would start and request a differently numbered temporary file; one that was created on a different client. This helped to avoid client-side caching.

A 1 GB file was selected for all the tests. This size may be a little larger than the average size of a user file, but it was large enough that when the number of clients increased, the tests would stress the disks and not run for days. This would not fully eliminate the server-side cache, especially for the low number of clients, but the aggregate from all the clients tested would fill the server cache.

The Iozone commands that were used for the tests are shown below. TESTDIR was the NFS mount point for the test and the $2 was used to select the appropriate temporary file to use. All output was placed into run.txt which was a temporary file stored locally. This file was later copied to a central server for processing.

```
iozone -c -t 1 -i 1 -s 1000m -e -w -f
${TESTDIR}/iozone.$2 >> /tmp/run.txt
iozone -c -t 1 -i 0 -s 1000m -e -w -f
${TESTDIR}/iozone.$2 >> /tmp/run.txt
```

Tests were limited to sequential reads '-i 1' and sequential writes '-i 0'. There has been little testing done with random access reads/writes or profiling.

Only a single Iozone process ran on each client system. We wished to test throughput and server connectivity so we ran a few tests to see if running multiple Iozone tasks on the clients would give us better throughput and more connections. Table 1 showed us that running multiple instances or multiple threads on a client gave us mixed results. This may be investigated further to provide more connections for scalability testing.

Reiser	Read	Write
1 Stream	41823.08	44592.68
2 Streams	28586.15	38451.68
2 Processes	24770.57	37046.32
2 Machines	27943.13	51877.45

Table 1: Stream performance number (KB/s)

The server side consisted of dual 1.6 GHz Athlons with an LSI 2 Gb fibre channel card. The disk system was a 7 drive IDE-RAID system in a RAID 5 configuration. A dual port SysKonnect card provided the network connection. (Only one port was active.) This server was running a 2.4.19-pre10 kernel with the NFS_ALL[TMNB24] patches and the XFS extensions. The NFS all patches also contained Neil Brown's NFS over TCP patch which allowed for up to 32K read/write block size. (It was not necessary to patch the clients to use 32K read/write blocks.) A further note should be made that UDP was the transport used for all test - NFS over TCP will be looked at later. All other drivers were stock with the kernel. The RAID 5 system had one partition and the file system was placed on that. There where 35 nfsd threads started on the server. The effects of using a larger number of threads, especially with regards to scaling may be investigated at a future time.

4. Ext3 tests

We started with the default Linux file system ext2 journaling variant, ext3.[OLS00] The ext3 drivers are distributed with the kernel so no patches were required. The file system was created with this command:

```
mkfs.ext2 -j -b 4096 -i 262144 /dev/sda1
```

This creates a journaled file system with 4k blocks and a decreased number of inodes. The default journaling method that EXT3 uses is:

"data=ordered" Only journals metadata changes, but data updates are flushed to disk before any transactions commit. Data writes are not atomic but this mode still guarantees that after a crash, files will never contain stale data blocks from old files.[EFAQ]

Local tests showed:

Clients	1	6	13	40
Reads	47917.35	8592.97	7272.20	8179.06
Writes	47556.69	35722.87	37727.60	41864.59

Table 2: EXT3 Local Performance (KB/s)

While the NFS tests showed:

Clients	1	6	13	40
Read 8k	8028.14	8578.72	8259.68	5700.14
Read 32k	9716.24	13944.02	14718.30	14977.66
Write 8k	15424.91	12939.76	40389.36	41815.93
Write 32k	21692.16	12885.18	13742.93	41587.48

Table 3: EXT3 NFS Performance (KB/s)

Reading from the EXT3 file system seems to peek at 8.5 MB/sec for anything greater than one process. There is a 'big kernel lock' (BKL) patch that can be applied to EXT2 and this should improve EXT3 reads, and may be investigated later. This BKL could also account for the odd read and write performance.

Advantages:
- Included in the kernel.
- Default file system, thus well tested.

Disadvantages:
- Little optimization for high performance.

Future:
- New H-Tree patches for better meta-data access.[HTR]
- Big kernel lock removal in file system code.[LSE]

5. JFS tests

JFS[JFS] version 1.0.20 from the IBM site was used. Installation, configuration and setup was very clean. It installs into its own directory and only modified the configuration menu.

Local performance numbers were:

Clients	1
Reads	47035.60
Writes	46345.12

Table 4: JFS Local Performance (KB/s)

We were not able to gather any other local performance numbers for JFS since running the six simultaneous tasks caused the system to hang repeatably. No oops code was generated.

As for the NFS performance, here are the numbers:

Clients	1	6	13	40
Read 8k	7797.43	7595.22	6801.22	4746.73
Read 32k	8360.69	13377.27	15295.13	13226.02
Write 8k	6495.13	11676.33	17546.88	9599.22
Write 32k	6889.07	11760.76	29741.40	14516.31

Table 5: JFS NFS Performance (KB/s)

Here we see that JFS seems to suffer form the same read performance limitation as EXT3. While write performance does improve, it seems that there is some contention still in the code.

One other problem was encountered with JFS. When performing a 40-client test with an Qlogic card, the kernel generated an oops in jfs_metapage.c. This was repeated twice.

Advantages:
- A very clean installation process.

Disadvantages:
- Unstable

6. ReiserFS tests

The ReiserFS[REIFS] tests where done with the default code found in the 2.4.19-pre10 kernel. Also we created the default file system with the command:

```
mkfs.reiser /dev/sda1
```

Reiser's local performance was measured at:

Clients	1	6	13	40
Reads	46014.53	24714.50	19602.21	18965.23
Writes	47401.78	43957.83	44230.47	41886.47

Table 6: ReiserFS Local Stats (KB/s)

While its NFS performance as seen in Table 7 was:

Clients	1	6	13	40
Read 8k	8403.05	9568.47	7967.47	5865.89
Read 32k	42537.24	21830.05	17845.33	14415.63
Write 8k	30784.17	36629.59	36594.95	41512.78
Write 32k	44201.83	51213.73	50429.35	41727.31

Table 7: ReiserFS NFS Stats (KB/s)

As can be seen, Reiser does very well even when scaling. Reiser's other strength is in its meta-data performance, (testing just started on this aspect, but meta-data is not as critical to PDSF.)

Advantages:

- Good data and meta-data performance and scalability.
- Included in the kernel.

Disadvantages:

- None really.

7. XFS tests

Here are both the local and NFS performance numbers for the XFS[XFS11] file system:

XFS Local	1	6	13	40
Reads	48321.17	74022.53	48234.87	35648.13
Writes	49136.25	54591.74	50239.26	40686.25

XFS NFS	1	6	13	40
Read 8k	18946.82	47670.20	34236.26	19247.73
Read 32k	29336.81	58523.86	35676.95	26239.88
Write 8k	26392.25	53153.81	37025.32	44936.48
Write 32k	54936.93	53115.75	48487.28	44277.17

Table 8: XFS Local and NFS Performance (KB/s)

Installation here was also straightforward - Just install the tar ball from the web site. Although the installation is clean, XFS does modify other Linux files, such as parts of Linux's VFS layer.

XFS's performance stays pretty high up to 40 clients and the write performance does not drop much. The modification to the VFS layer seems to be justified. But how do these changes effect the other file systems tested? We will talk about that in Section 10.

Advantages:

- Fastest and more scalable Linux file system tested.
- Promising other add-ons (ie DMAPI).

Disadvantages:

- Modified of the VFS layer.

8. GPFS tests

We included the GPFS tests here to see what a current cluster file system can do. The tests were performed on the Alvarez Linux cluster, which consists of 87 dual PIII 866 nodes.

The GPFS setup on Alvarez consists of one volume which is shared across two I/O nodes. Each I/O node has 2 GB of memory and the storage network is across Myrinet. The tests were done by setting up a PBS job to schedule the Iozone tasks across the desired number of nodes. As above, there where 40 test files which were created earlier, and the test started by doing a read test and then a write test. The test script also cycled through the available test files so the file read was different than the file written.

Installation and setup of GPFS was easy. RPMs were used to distribute and install files on the servers and clients. At this point the file system was built and configured by the supplied utilities. This all went very well.

From Table 9 you will see that GPFS has a good single stream performance and tends to level out nicely. There was a slight dip as the number of concurrent files increased, but no serious performance problems where seen during the testing.

Clients	1	6	13	40
Gpfs-read	57570.93	83449.06	82461.17	81942.55
Gpfs-write	65291.69	101991.99	101413.85	100797.08

Table 9: GPFS Performance numbers (KB/s)

Even though we did not experience any problems during our testing, there have been a few problems. One of the most frustrating was with server fail over. The fail over from one server to the other worked without a problem, but when the down server was brought back up and tried to join GPFS, the original server would not relinquish control and the whole GPFS volume needed to be taken down and restarted. If you would like more information about GPFS on Intel, please contact Thomas Davis (TADavis@lbl.gov).

GPFS does provide a lot of the functionality that PDSF is looking for in a file system, but in order for PDSF to implement a GPFS solution we may have to make too many changes. PDSF could not use GPFS as tested since we only have a fast Ethernet network. This could be done, but we fear the performance would not be very good. Also we could treat GPFS as a NAS. Basically NFS export from the I/O servers the GPFS volume. At this point we would lose the benefits of a cluster file system.

More testing will be done with GPFS. We will be looking at trying to export Alvarez's GPFS volume to PDSF and measure its performance. If things work out, we would also like to do the same with the SP2's GPFS volume. In any case, other testing to be done with GPFS is in the low end to see when the maximum throughput is reached.

Advantages:

- Very nice performance and scaling.

Disadvantages:

- Tied to RedHat kernels (2.4.9 currently)
- Stability could be improved.
- Not released yet.

9. Other tests for comparison

These next two tests were performed on NAS appliances. BlueArc[BA] is a system based on FPGA's. Embedded within the FPGA's are the protocols (NFS, CIFS) and the file system code. BlueArc uses fibre channel disks and RAID controllers to do all the back end work. Our system was an Si7500 with three volumes of 24x73 GB disks, see Fig 1. The disks were arranged in a RAID 5 configuration with each volume seen as a separate mount point.

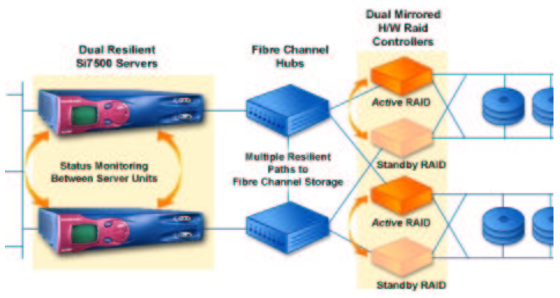

Fig 1: BlueArc Architecture

The second system was a Zambeel[Z] beta system, which takes a more modular approach to design. It uses more off-the-shelf components and can be expanded more easily. Our system had a total of 4.5

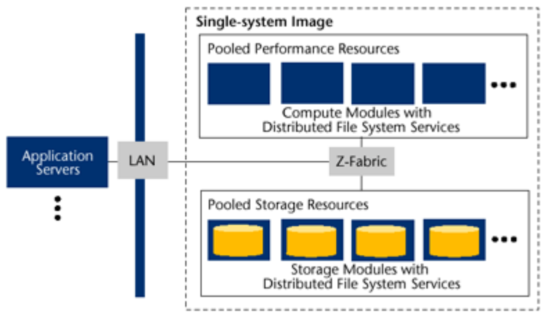

Fig 2: Zambeel Architecture

TB of disk space in a RAID 0 + 1 configuration. The entire NAS volume is seen as one mount point. See Fig 2 for an abstract view of the system.

The measured performance of both systems can be seen in the Table 10. Both systems showed good scaling and better performance than our old 3ware setup.

Reads	1	6	13	40
bluearc-8k	40575.87	53236.93	60174.36	96260.70
bluearc-32k	44569.88	113487.58	103863.53	101062.00
zambeel-8k	12420.33	55096.81	83153.35	93430.13
zambeel-32k	14003.56	69839.84	146948.82	179036.76

Writes	1	6	13	40
bluearc-8k	51556.87	58458.49	58182.25	57421.55
bluearc-32k	38374.12	68768.49	70493.14	65259.10
zambeel-8k	6076.13	21436.48	29228.35	40337.94
zambeel-32k	9052.22	39046.55	51213.14	64013.78

Table 10: NAS System Performance (KB/s)

10.Combined outlook

To start with, we should revisit for a minute the XFS modifications to the kernel. Since the VFS layer was changed we ran some tests of the different file systems with and without the XFS changes. Tables 11 and 12 show the read and write test differences for each of the other file systems.

Reads	1	6	13
Reiser 8k XFS	8403.05	9568.47	7967.47
Reiser 8k Non	15459.91	9830.54	8719.35
Reiser 32k XFS	42537.24	21830.05	17845.33
Reiser 32k Non	12641.68	9815.06	7380.44
JFS 8k XFS	7797.43	7595.22	6801.22
JFS 8k Non	12849.63	12247.40	8444.66
JFS 32k XFS	8360.69	13377.27	15295.13
JFS 32k Non	10314.70	10714.07	8478.87
EXT3 8k XFS	8028.14	8578.72	8259.68
EXT3 8k Non	7228.80	10105.08	7893.10
EXT3 32k XFS	9716.24	13944.02	14718.30
EXT3 32k Non	7582.21	10223.15	7707.76

Table 11: Read performance between an XFS and Non XFS kernel (KB/s)

Since PDSF was wanting to use XFS's DMAPI support, we looked primarily at kernels with the XFS modifications. This was an administrative decision to help keep the differences in configuration of the cluster to a minimum.

In general, the 8k read tests showed the file systems performed better with a non-XFS modified kernel, while the 32k tests showed them working better with the XFS modifications. On write performance, the improvement was not seen, and most file systems actually showed degradation in performance. Overall, JFS seem to be affected more in the negative, while Reiser seemed to enjoy a boost in performance. Further investigation is needed to see if the XFS modifications address problems that are being looked at in the LSE[LSE] project.

Writes	1	6	13
Reiser 8k XFS	30784.17	36629.59	36594.95
Reiser 8k Non	31787.37	36181.90	37100.72
Reiser 32k XFS	44201.83	51213.73	50429.35
Reiser 32k Non	31530.96	36188.12	35364.90
JFS 8k XFS	6495.13	11676.33	17546.88
JFS 8k Non	11285.75	12008.36	13395.40
JFS 32k XFS	6889.07	11760.76	29741.40
JFS 32k Non	9313.03	11092.02	14850.98
EXT3 8k XFS	15424.91	12939.76	40389.36
EXT3 8k Non	14752.48	43210.31	32651.24
EXT3 32k XFS	21692.16	12885.18	13742.93
EXT3 32k Non	14762.58	40937.51	39806.24

Table 12: Write performance between an XFS and Non XFS kernel (KB/s)

Figs 3 and 4 are graphs to show the differences between the tested file systems. While Figs 5 and 6 show how the best Linux file system compared to other solutions.

11.Summary of results

Where to go from here? There are still a lot of areas to look at. But with the 32k performance of most of the file systems, a look at NFS over TCP or Jumbo frames may not buy us any performance increases. NFS over TCP may provide some increased measure of reliability, but that may be secondary to looking at other performance enhancements. The areas next on the list are the 'bounce buffer' patches. This only affects machines with more than 1 GB of memory and has to do with DMA transfers. A group at Berkeley Lab has seen marked improvements with these patches, but they were doing raw I/O and we will be investigating to see if the improvements are passed on to the file system. The next area to look at is lock reduction (BKL). It is our hope that these changes will increase local performance and will translate into improved NFS performance. If at this point we see a growing gap between the local performance and the 32k numbers, then we will look at NFS over TCP and Jumbo frames. Lastly, we would like to look at the 2.5 kernel with its rewritten block layer.

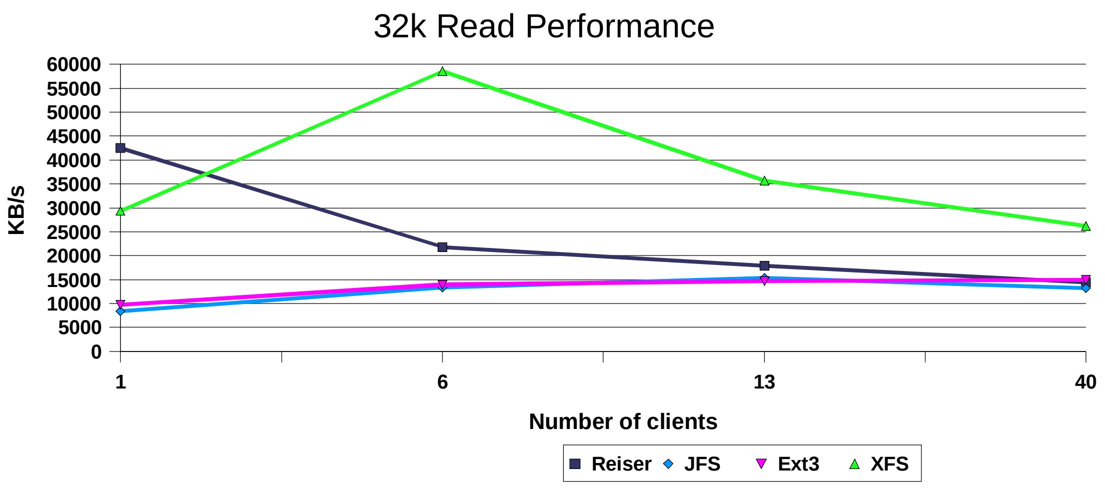

Fig 3: File Systems Read

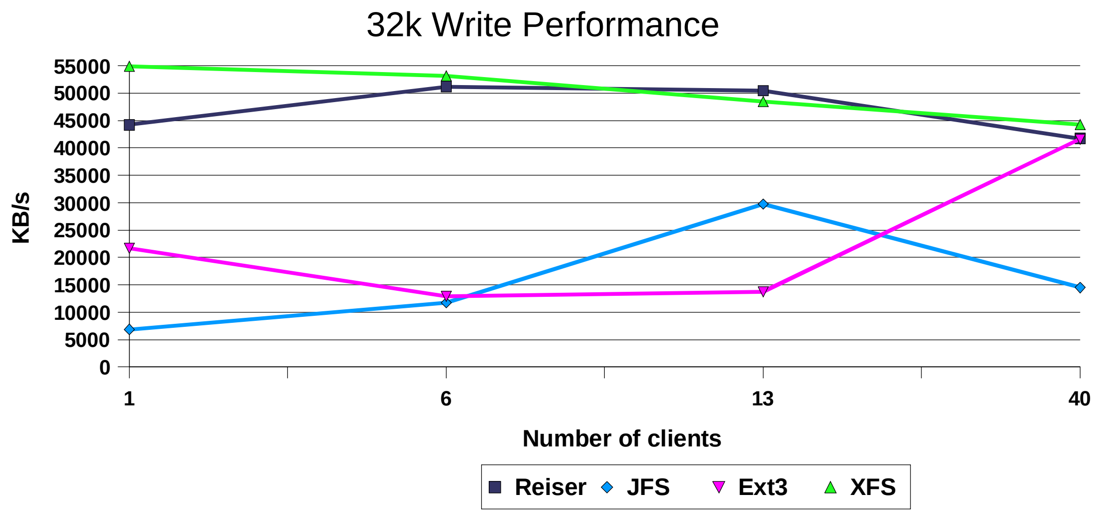

Fig 4: File Systems Writes

And finally, more testing will be done with increasing the numbers of clients accessing the file systems. We have some tests up to 190 clients, but with the build-out this year we are hoping to increase that number to about 260.

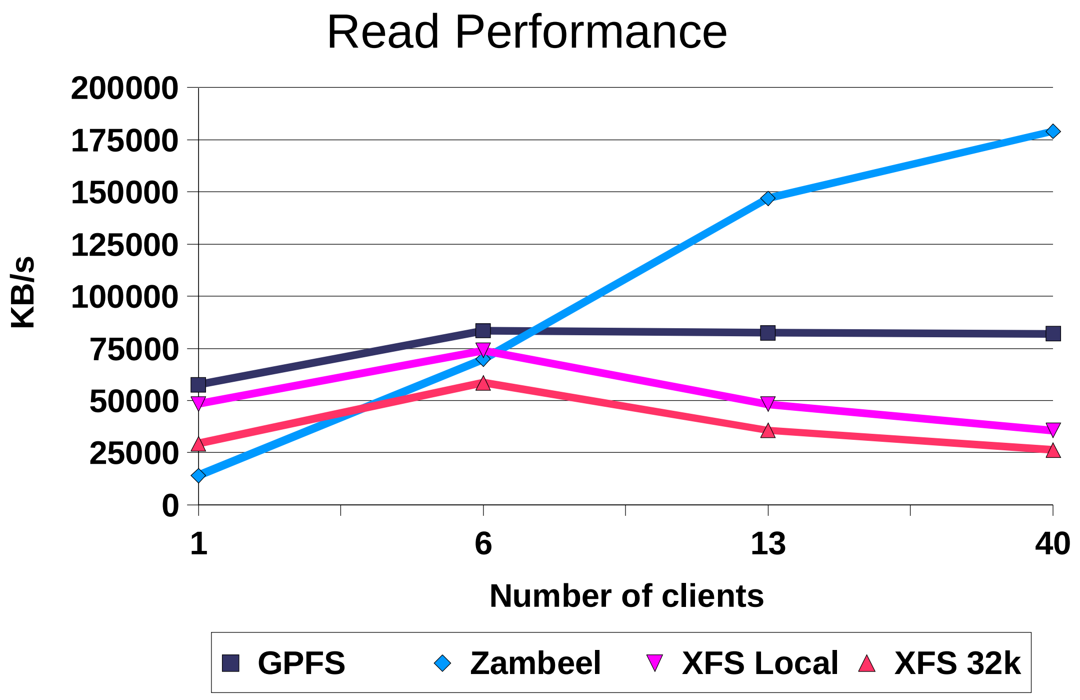

Fig 5: Best Systems Read (KB/s)

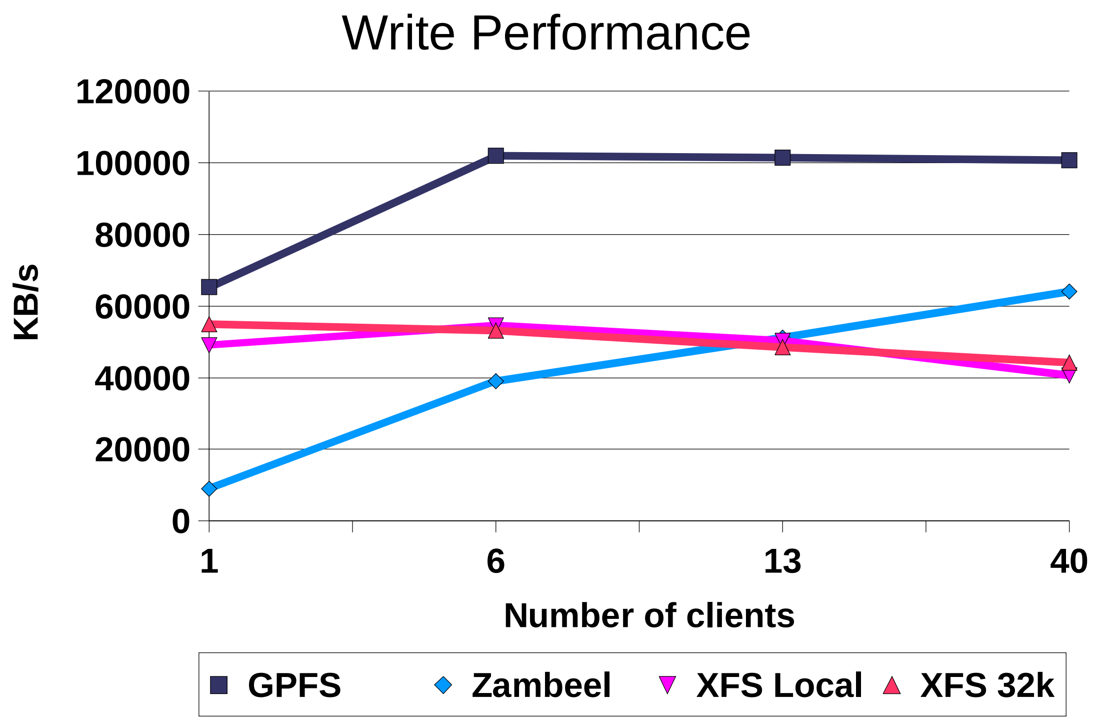

Fig 6: Best Systems Write (KB/s)

References

IOZ: Don Capps, IOzone Benchmark, ,http://www.iozone.org

TMNB24: Trond Myklebust, Neil Brown, NFS All Patches, , http://www.fys.uio.no/~trondmy/src/2.4.19-pre8/

OLS00: Dr Stephen Tweedie, EXT3, Journaling Filesystem, 2000,http://olstrans.sourceforge.net/release/OLS2000-ext3/OLS2000-ext3.html

EFAQ: Juri Haberland , EXT3 FAQ, , http://people.spoiled.org/jha/ext3-faq.html

HTR: Daniel Phillips, EXT3 Htree patches, , http://people.nl.linux.org/~phillips/htree/

LSE: , Linux Scalability Effort, , http://lse.sourceforge.net/

JFS: IBM, JFS Web site, , http://www-124.ibm.com/jfs/

REIFS: Hans Reiser, ReiserFS, , http://www.reiserfs.org/

XFS11: SGI, XFS 1.1 File system, ,http://linux-xfs.sgi.com/projects/xfs/

BA: , BlueArc Company, , http://www.bluearc.com/

Z: , Zambeel Company, , http://www.zambeel.com/