DL-TR-2006-008

CCLRC

CAE and Computational Chemistry Application Performance I: Benchmark Studies Using Object-Based, InfiniBand-Connected Storage

Miles Deegan, Mark Kelly, Michael Lough, Sharon Shaw *and* **R. Kent Koeninger.**

October 2006

Council for the Central Laboratory of the Research Councils

Enquiries about copyright, reproduction and requests for additional copies of this report should be addressed to:

Library and Information Services
CCLRC Daresbury Laboratory
Daresbury Warrington
Cheshire WA4 4AD
UK
Tel: +44 (0)1925 603397
Fax: +44 (0)1925 603779
Email: library@dl.ac.uk

ISSN 1362-0207

CAE and Computational Chemistry Application Performance I: Benchmark Studies Using Object-Based, InfiniBand-Connected Storage

Miles Deegan[1], Mark Kelly[2], Michael Lough[2], Sharon Shaw[2] and R. Kent Koeninger[2].

Abstract

The I/O performance of internal disks in thin servers used in typical HPC Linux clusters can be inadequate for scratch I/O intensive applications in fields such as Computer Aided Engineering (CAE) and Computational Chemistry (CC). Examples of such codes are: MSC.Nastran and ABAQUS (CAE); NWChem (CC). A common approach to providing I/O facilities for applications such as these has been to add direct-attached external disk arrays on distributed servers to meet their requirements. Distributed direct-attached storage increases complexity, adds cost, reduces reliability, and creates specialized node types that are more difficult to schedule.

New object-based storage techniques using high-speed cluster interconnects may be efficient for both permanent-shared *and* temporary-scratch storage requirements, thus improving the cost and ease of use of the compute clusters.

This initial study – a collaboration between CCLRC Daresbury Laboratory and HP's HPC Division - will measure the performance and throughput of object-based storage with an InfiniBand fabric to ascertain whether such an architecture is a viable alternative to distributed-internal or external-direct-attached storage in thin-client Linux clusters.

[1] Distributed Computing Group, CCLRC Daresbury Laboratory, Warrington, UK.
[2] High Performance Computing Division, Hewlett-Packard Corp.

1. Introduction

Previous I/O benchmark studies have tended to measure the maximum bandwidth one can demonstrate with synthetic I/O benchmarks such as IOzone[3] . This study will revisit IOzone performance and in addition use a mixture of commercial ISV and academic/government-developed applications for which users tend to purchase thin-client Linux clusters. We present data which will demonstrate how various I/O architectures affect the run time and throughput of these applications, and whether or not Object-based Storage Device (OSD) solutions (such as Lustre™) can improve the throughput and ease of use of Linux clusters and reduce the total cost of ownership (TCO).

The CAE and CC applications we have benchmarked are: MSC.Nastran[4] and ABAQUS[5] (CAE), plus NWChem[6] (CC).

To date many HPC sites have deployed DAS-JBOD (direct attached storage to just a bunch of disks) on each server to meet scratch I/O performance requirements. They may augment this DAS-JBOD approach with centralized RAID-protected NFS or object storage for shared and permanent data. We investigate the replacement of such DAS-JBOD and NFS services with InfiniBand-connected object storage while maintaining equivalent performance for a widely used set of applications.

The Infiniband-connected object storage used in this study is HP SFS. HP SFS is largely based on Lustre™, an open source standard protocol and reference code initially written by Cluster File Systems, Inc. (CFS)[7]. HP SFS represents a transformation of the open source Linux distribution into an appliance that is easy to install, manage and use. The major components of SFS are data servers (each server is an Object Storage Server, or OSS), a metadata server (MDS), an LDAP (Light-weight Directory Access Protocol) server, and one or more network fabrics. The network fabric integrates the other hardware components and any clients (cluster compute nodes) that will use SFS. LDAP is an open and commonly used

---

[3] IOzone is a popular filesystem benchmark that generates and measures a variety of file operations. For more information see http://www.iozone.org.

[4] MSC.Nastran is a powerful general purpose finite element analysis solution. For more information see http://www.mscsoftware.com.

[5] ABAQUS, Inc. is one of the world's leading providers of software for advanced finite element analysis. For more information see http://www.ABAQUS.com.

[6] NWChem 4.7 Production version: Aprà, E.; Windus, T.L.; Straatsma, T.P.; Bylaska, E.J.; de Jong, W.; Hirata, S.; Valiev, M.; Hackler, M.; Pollack, L.; Kowalski, K.; Harrison, R.; Dupuis, M.; Smith, D.M.A; Nieplocha, J.; Tipparaju V.; Krishnan, M.; Auer, A.A.; Brown, E.; Cisneros, G.; Fann, G.; Fruchtl, H.; Garza, J.; Hirao, K.; Kendall, R.; Nichols, J.; Tsemekhman, K.; Wolinski, K.; Anchell, J.; Bernholdt, D.; Borowski, P.; Clark, T.; Clerc, D.; Dachsel, H.; Deegan, M.; Dyall, K.; Elwood, D.; Glendening, E.; Gutowski, M.; Hess, A.; Jaffe, J.; Johnson, B.; Ju, J.; Kobayashi, R.; Kutteh, R.; Lin, Z.; Littlefield, R.; Long, X.; Meng, B.; Nakajima, T.; Niu, S.; Rosing, M.; Sandrone, G.; Stave, M.; Taylor, H.; Thomas, G.; van Lenthe, J.; Wong, A.; Zhang, Z.; "NWChem, A Computational Chemistry Package for Parallel Computers, Version 4.7" (2005), Pacific Northwest National Laboratory, Richland, Washington 99352-0999, USA.

"High Performance Computational Chemistry: An Overview of NWChem a Distributed Parallel Application," Kendall, R.A.; Apra, E.; Bernholdt, D.E.; Bylaska, E.J.; Dupuis, M.; Fann, G.I.; Harrison, R.J.; Ju, J.; Nichols, J.A.; Nieplocha, J.; Straatsma, T.P.; Windus, T.L.; Wong, A.T.; Computer Phys. Comm. 2000, 128, 260-283.

[7] For more information see http://www.clusterfs.com.

standard for managing configuration and security information. The SFS MDS and OSS components store the separated file metadata and data, respectively. Disk arrays comprise the underlying storage medium for both OSS and MDS nodes. With the OSS nodes, there is a further level of abstraction with disk arrays being arranged into groups called Object Storage Targets (OSTs) – actual file data will be stored on one or more OSTs. On SFS, from a user's perspective, file data is striped across a number of OSTs using a fixed stripe size and by using a round-robin algorithm for picking the first OST to use (the default number of stripes and stripe size are configured during SFS installation). User-level tools are provided for modifying the striping configuration on file or sub-directory levels.

This investigation compares run times using internal disks, an external striped (RAID 0) storage array (DAS-JBOD), and HP SFS storage connected over InfiniBand (IB). We will determine which of these I/O-intensive applications run with similar run times using HP SFS in place of the DAS-JBOD and local scratch disks, and test throughput limits with multiple instances of the same applications running in parallel on a shared HP SFS storage appliance.

2. Systems' Details: Configurations, data and storage layout, and software stacks

Benchmarks were conducted on two AMD Opteron-based Linux clusters located within HP facilities in the United States. The thin client nodes (128 of them) were HP ProLiant DL145 G2 servers – 4 core SMPs (two dual-core 2.2 GHz Opteron sockets), 8 GB of RAM. Local storage on the servers comprised either two internal, two-way striped, 10K RPM 146 GB SCSI disks (2 jobs per host MSC.Nastran and 4-way parallel ABAQUS), or one internal 7.2 K RPM 80 GB SATA disks (NWChem).

The DAS-JBOD storage used consisted of the same ProLiant servers with a StorageWorks MSA30 array, with I/O striped across ten 15K RPM 72 GB SCSI JBOD disks (RAID 0) attached with dual StorageWorks Smart Array SA6402 SCSI controllers.

(Note: the benchmarks for the internal-disk and DAS-JBOD test were timed on individual servers. The results are assumed to scale linearly across multiple servers, given no shared components to limit the scalability.)

Details of the HP SFS connected storage configuration are as follows. The InfiniBand network was provided by a single core switch (Voltaire SDR) with 288 ports. The HP SFS server consisted of eight object-storage servers (OSSes) and one active metadata server (MDS). Each OSS was a single ProLiant DL380 with 4 StorageWorks SFS20 disk arrays. Each SFS20 included two 320 MB SCSI cables (one active, one passive), twelve 250 GB, 7200 RPM SATA drives and one StorageWorks (RAID 5) controller – nine data disks, one parity disk and two spares. The SFS20 arrays were striped using Lustre™ OSTs. The MDS was a single ProLiant DL380 server with two StorageWorks SFS20 disk arrays, and each SFS20 array included two 320 MB SCSI cables (one active, one passive), twelve 250 GB, 7200 RPM SATA drives and one StorageWorks controller. On the MDS, the SFS20 arrays were mirrored (RAID 1) for maximum resilience.

The cluster software was HP XC system software version 3.0 (RC3 20051218), which is binary compatible with Red Hat Enterprise Linux 4 (RH EL4). The InfiniBand firmware and driver were from Voltaire release v3.4.5_24. EXT2 and EXT3 filesystems were tested on the internal and DAS-JBOD disks. The HP SFS software was version 2.1, based on Lustre™ version 1.4.2 from Cluster File Systems Inc., with selected modules from releases 1.4.3, 1.4.5, and 1.4.6.

In terms of applications, we used version 2005r2 of MSC. Nastran, ABAQUS 6.5-5, and NWChem version 4.7.

3. Benchmarks

i. Introduction

For background information on the storage hardware performance we used the IOzone synthetic benchmarks to measure well-behaved, large-block I/O bandwidth on the three architectures in question: internal disks, DAS-JBOD, and HP SFS. We measured the HP SFS bandwidth with parallel I/O from one to thirty-two servers with one instance of IOzone running on each server.

A starting point for each application's performance was established with up to two-way striped internal SCSI disks on an individual ProLiant server (with the exception of NWChem – see below). We compared these timings to run times using DAS-JBOD on a similar server and selected applications where the run times improved significantly.

We then compared these timings to run-times using HP SFS storage. We increased the number of parallel applications running until we saturated the given HP SFS storage performance capacity.

For MSC.Nastran, we report runs with two serial (non-parallel) instances per compute server (per SMP). We saw similar run times with one and two instances per server. Running four instances delivered slower run times, regardless of the storage configuration, potentially masking the storage-performance differences. We assume interactions in the SMP other than storage speed contributed to this non-linear scalability for four-instances per SMP. To avoid the effects of this extraneous variable we only reported runs with two instances per SMP.

For ABAQUS we used one instance running 4-way parallel per server across multiple servers simultaneously.

NWChem is parallelized with the Global Arrays toolkit[8] which gives a portable and efficient "shared memory" interface for distributed memory parallel computers. The underlying communications are carried out using HP-MPI, but I/O is not done via MPI-IO, but rather PNNL's EAF libraries[9]. We benchmarked the code's semi-direct (i.e. out of core) MP2 gradient module and produced benchmark data for 2 and 4 processes per 4-core server.

ii. Synthetic Benchmark I/O Bandwidth

The goal of this study is to measure how different I/O hardware effects real application performance. As an initial step it is useful to establish the theoretical performance of the I/O hardware for well-behaved I/O. These are high-watermark benchmarks that are not necessarily achievable by CAE and CC applications.

We used the IOzone benchmark to measure high water marks for bandwidth for well-behaved I/O patterns. For the two-way striped internal SCSI disk configuration, the disks could be read at 91 megabytes-per-second (MB/s) and written at 84 MB/s, per server. For the single internal SATA disk configuration, the disk could be read at 48 MB/s and written at 23 MB/s, per server. The DAS-JBOD disks could be read at 323 MB/s and written at 240 MB/s, per server.

The I/O rates on this distributed hardware are assumed to scale linearly with the number of servers given no shared components among the servers. This would be an aggregate of roughly 2.8 GB/s writing to the two-way striped internal disks in 32 servers and roughly 7.5

[8] http://www.emsl.pnl.gov/docs/global/ga.html
[9] http://www.emsl.pnl.gov/docs/parsoft/chemio/EAFapi.html

GB/s aggregate bandwidth writing to DAS-JBOD external disks on 32 servers with one disk array per server (32 disk arrays total). A comparable HP SFS server with 32 disk arrays (8 OSSes with four SFS20 arrays per OSS) sustained roughly half the DAS-JBOD rate: 3.2 GB/s aggregate write performance.

Note the DAS-JBOD array use simple striping without RAID-parity protection which tends to deliver substantially higher bandwidth than a similar set of RAID 5 protected disks. This is true when comparing these DAS-JBOD arrays (323 MB/s for read and 240 MB/s for writes) with the RAID 5 SFS20 arrays rated at 190 MB/s for reads and 130 MB/s for writes.

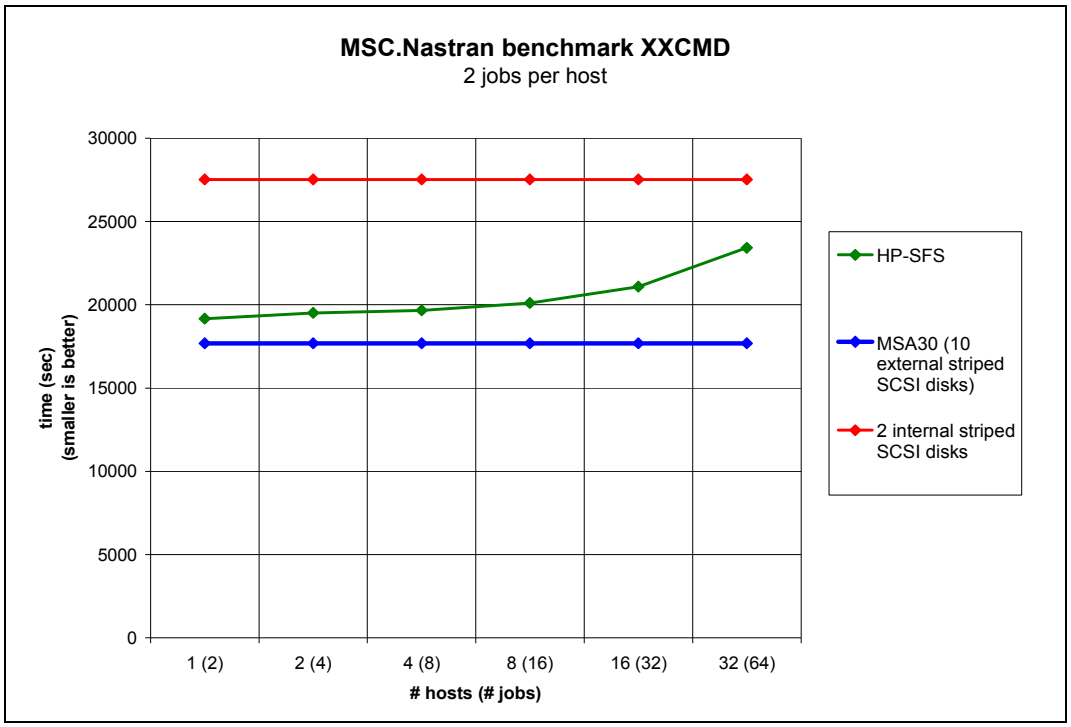
As expected, the aggregate performance and performance per server of the shared HP SFS storage varied with the number of servers simultaneously accessing the shared storage. The following table shows the *aggregate* bandwidths in MB/s for the HP SFS storage appliance ("agg" column). The *per-host* bandwidths, also in MB/s, are also shown in the table ("avg" column). As expected the aggregate bandwidth increased as more servers accessed the storage appliance simultaneously. The theoretical hardware write bandwidth limit was near 3.2 GB/s and using IOzone we were able to meet this limit with 32 hosts, each simultaneously running one instance of IOzone. We also note that with up to 8 simultaneous hosts, the per-host HP SFS write bandwidth was faster than the 240 MB/s measured for the DAS-JBOD write bandwidth.

hosts	1		2		4		8		16		32	
	agg	avg	agg	avg	agg	avg	agg	avg	agg	avg	agg	avg
read	225	225	456	228	896	224	1770	221	3610	226	4580	143
write	295	295	600	300	1120	280	2092	261	2402	150	3284	103

iii. MSC.Nastran

The MSC.Nastran standard benchmark XXCMD is a eigenvalue solution of the natural frequencies of an automotive body up to 200 Hz. It uses less than 1 GB of memory but reads and writes over 2.4 TB of I/O in a 43 GB file footprint. The Lanczos method used a block size of 7 and calculated 1073 roots in 384 solves of 10 decompositions (shifts).

The performance ratios quoted in the following paragraphs are for comparisons to timings using internal disks. All timings are graphed in the following figure:



With two jobs running on a single-server, both external DAS-JBOD and HP SFS demonstrated performance improvements over running with two-way striped internal disks: 1.56 and 1.44 times faster, respectively. The HP SFS performance ratios remained fairly constant up to 8 hosts running 16 jobs concurrently: 1.37 times faster than using two internal disks. With 16 and 32 hosts the run times began to slow down but continued to deliver improvements over using two internal disks: 1.31 and 1.17 times faster, respectively.

(The ratio using DAS-JBOD was assumed to be constant at 1.56 times faster given its shared-nothing storage hardware and filesystems.) In the shared nothing tests no additional time was considered for transferring any input data from separate permanent storage to node-local storage or transferring any output results data from node-local storage to separate permanent storage. In some cases this data and transfer time could be significant and could affect the shared nothing solution times negatively.

Additional tests were conducted on the same system using a four-way internal SCSI stripe with 8 GB of memory and also with only 4 GB of memory installed. Performance differences (with 8 GB of memory) to the two-way internal disk configuration were noticeable (approximately 1.12 times faster) but since four drives can not fit into a thin client system this configuration was not given as much consideration. It is interesting to note that repeating the two jobs per host with four internal SCSI disks test on a system with only 4 GB of memory shows a dramatic slowdown in runtime to over 120,000 seconds, which is 4.8 times slower than the same test with 8 GB of memory. Two jobs, each with a memory footprint of approximately 1 GB, will not cause paging on a 4 GB system. Consequently, this observed slowdown illustrates how effective additional memory can be when used as a file cache by the operating system.

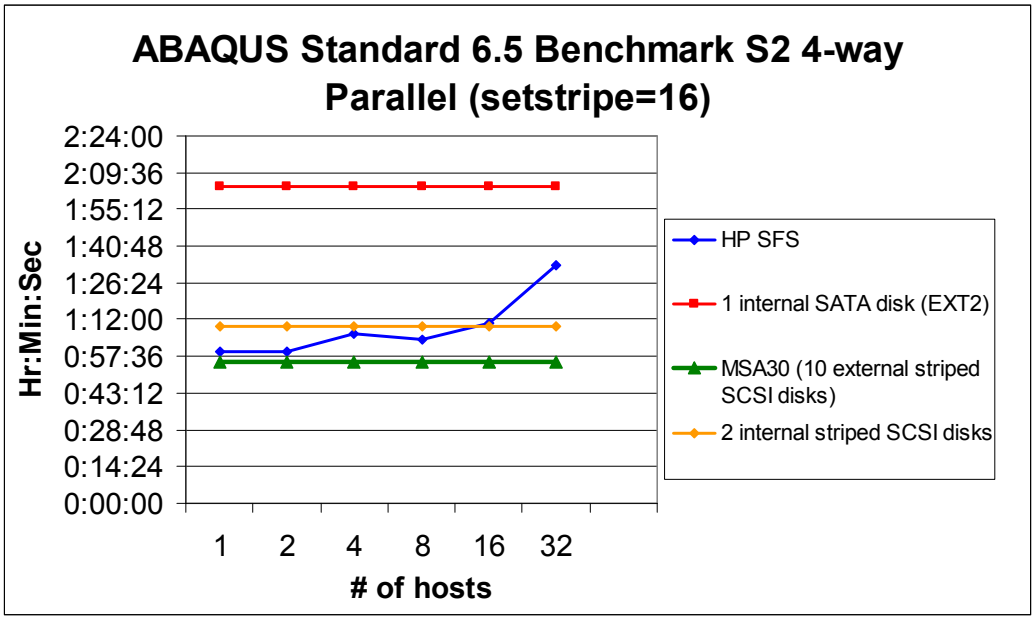
We conclude that HP SFS central storage was able to sustain reasonable throughput for a reasonable number of simultaneous I/O intensive MSC.Nastran runs with run times close to those of the external DAS-JBOD disks.

iv. ABAQUS

The ABAQUS standard benchmark, ABAQUS/Standard S2, is an eigenvalue analysis of a stiffened plate that is solved using the Lanczos eigensolver. This problem is dominated by I/O, since each Lanczos iteration requires backward passes, which are I/O bound, on the decomposed and shifted stiffness matrix. All timings discussed for S2 are graphed in the figure below.

The 4-way SMP-parallel ABAQUS benchmark was I/O intensive and, when compared to running on a single internal SATA disk, it ran 1.8 times faster on the 2-way striped internal SCSI disks and 2.2 times faster on the 10-way striped MSA30. In each of these scenarios, both the permanent job and scratch I/O data resides on the node-local file system. It should be noted that the permanent job data for a single instance of this ABAQUS benchmark amounts to over 3.5 GB. In the figure below, we assume these single server results scale linearly across multiple servers. This may be an optimistic assumption for this ABAQUS benchmark since all permanent job data remains on node-local storage and some of that may need to be copied off onto some shared file system for subsequent processing.

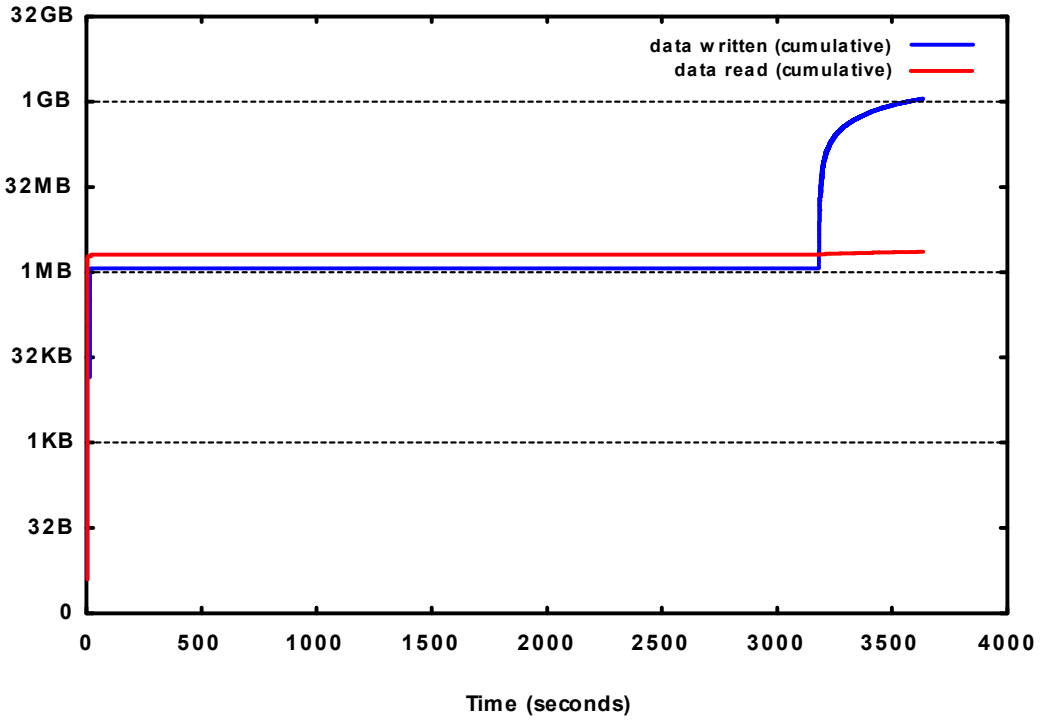
With both the permanent job and scratch I/O on HP SFS, the runs were between 2.1 and 1.8 times faster than the run on a single internal SATA disk, for between 1 and 16 simultaneous jobs, respectively. HP SFS runs were faster, or equally as fast, as the 2-way striped internal SCSI disks for up to 16 simultaneous jobs. Thus an eight-OSS HP SFS delivered good I/O performance for all ABAQUS I/O for ABAQUS running up to 64 cores (four per SMP). In contrast to the jobs run with node-local storage, after these SFS jobs complete, all permanent job data is already on a shared file system (HP SFS).



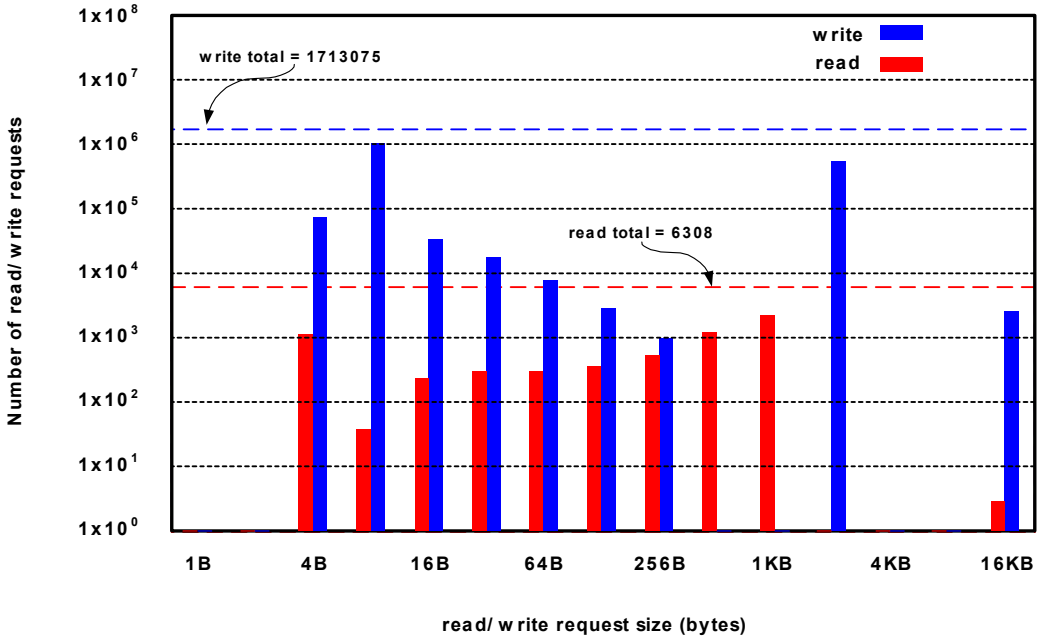
During installation, the default settings for SFS are to configure it with stripe width of 1, i.e., without further user interaction any file created will reside completely on a single OST. This configuration setting is optimal for a cluster, as it maximizes aggregate bandwidth for the file system. On the other hand, an application that reads or writes data in a single stream may

benefit from striping its files across a number of OSTs. For the ABAQUS S2 benchmark it was found that striping the output files (both permanent job and scratch data files) across 16 OSTs was optimal. This setting was used for all the ABAQUS throughput runs on SFS. We now believe that using this setting may have contributed to the relatively poor scaling performance for 32 hosts. When running in throughput mode, with this setting, there will be high contention for resources at the OST level – a single OST may have to service an I/O request from every host. I/O requests that are small and/or random will make matters worse.

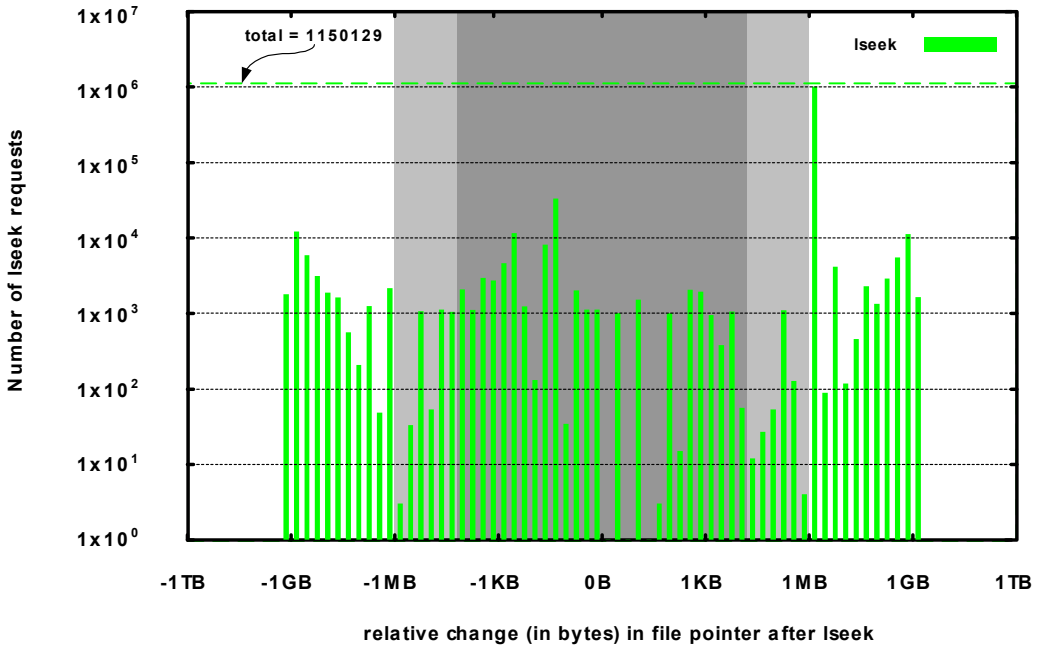
To investigate this a bit further, we reran a single instance of the ABAQUS job under the Linux "strace" tool to monitor all system calls during a normal run (including all I/O requests). We found that one of the permanent job files, the so-called ".odb" file, had the creation features that suffer significantly from contention issues, as we shall illustrate now. In the following figure we illustrate the volume of data that is written to and read from the ".odb" file during a run.



As can be seen, this file exists for the duration of the run, but it is in the final stages of the run that most of the data (over 1 GB) is written to the file. In our next figure we plot a histogram, showing the frequency of read and write requests versus request size, for the same "odb" file:



This shows that there are about a million write requests of size 8 bytes, a further half a million write requests of size 2 KB, and all requests are small in size (none that exceed 16 KB in size). Finally, it is worth pointing out that most read and write requests appear after an `lseek` command and illustrates that, for this ".odb" file, access is quite random (i.e., not in large sequential blocks). In the final figure of this section we plot a histogram of frequency of `lseek` requests versus request size:



There are a substantial number of requests to `lseek` forward and backward by more than 1 GB. However, most of the requests (~1 million) are to `lseek` forward by more than 1 MB. The SFS is configured with a stripe size of 1 MB, so it should be clear that most of the `lseek` requests will force the file pointer to switch continuously between different file system stripes (OSTs).

v. NWChem

We chose the code's semi-direct MP2 gradient module to benchmark since calculations on even modest sized molecular systems will result in considerable I/O requirements due to the inherent scaling properties of the algorithm.

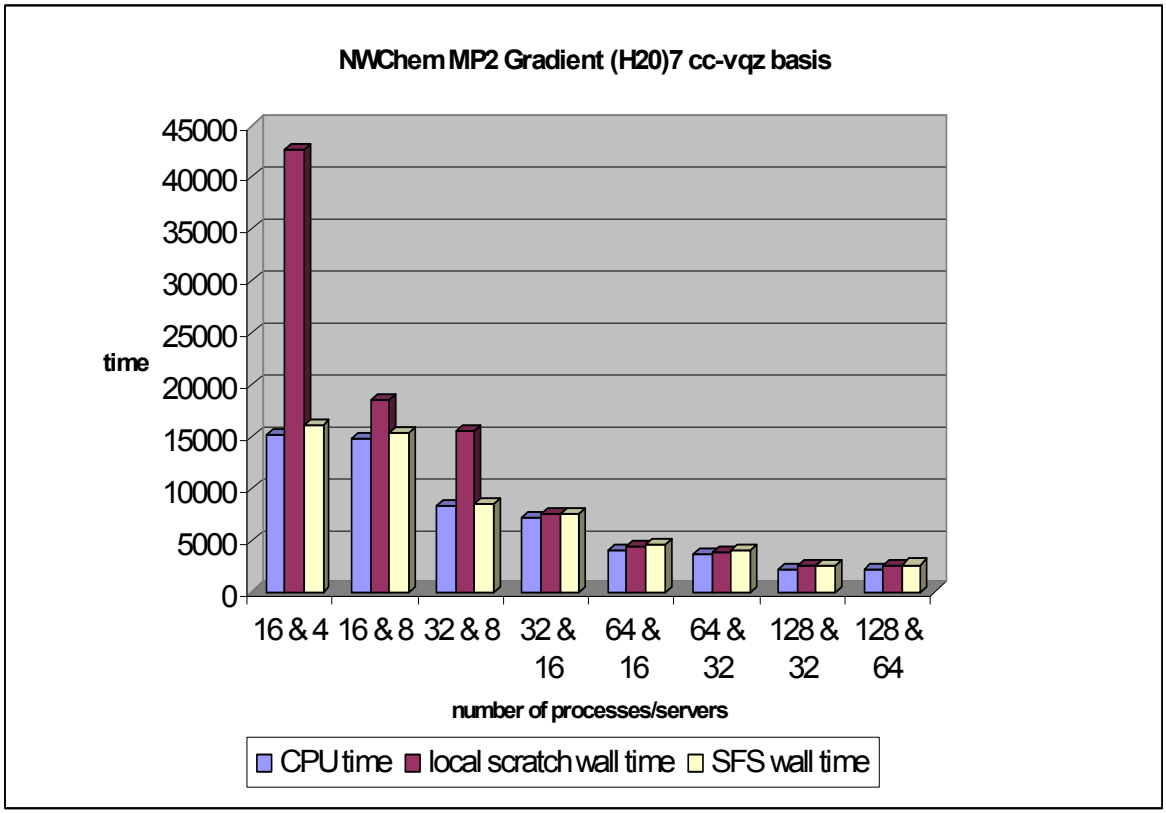
We have adapted PNNL's benchmark for a cluster of 7 water molecules[10] by increasing the size of the basis set cf. the original in order to ensure more exacting tests of the I/O options available to us. The original basis (Dunning's aug-cc-pvdz) results in a total of 287 basis functions; our choice (Dunning's cc-vqz) results in 805 basis functions. The amount of scratch data generated scales roughly as the fourth power of the number of basis functions (and compute requirements scale as the fifth power).

The MP2 gradient code was originally designed with distributed memory MPPs (with local scratch disks available to each compute node) in particular in mind, e.g. early IBM SP systems. Each process reads/writes its own exclusive access scratch files. These exclusive access files are essentially the same size across the cluster due to the dynamic load balancing within the algorithm.

These design considerations, coupled with the size of our adapted benchmark resulting in the inability to either fit the run into a small number of nodes and/or complete the job within a reasonable length of time (note the scaling properties of the algorithm above), meant we have not so far carried out measurement of performance on a single server with two-way parallel internal disks or an MSA30 DAS-JBOD array.

Performance figures from our initial investigations are encouraging. The table below summarizes the CPU time, and wall times for both local scratch and HP SFS for runs using between 16 and 128 processes. In addition we are able to compare the effect of running the code with two and four processes per DL145 G2 server. Little difference in the CPU times for *n* processes with 2 and 4 processes per server suggests memory bandwidth contention is not an issue. Contention for disk resources is a different matter. We observe the superior overall scaling of HP SFS with a considerable difference for lower process counts and for fully populated runs (all cores on a server in use).

[10] http://www.emsl.pnl.gov/docs/nwchem/benchmarks/index.html



Resrouce constraints have resulted in CCLRC/HP being unable to provide data for larger test cases at this time. In follow-on papers we intend to rectify this and extend the current study to include data for the $(H_2O)_7$ benchmark with larger basis sets, and for other molecular systems, for the following reasons:

- As the amount of scratch data increases, and therefore the size of each process's scratch files increases, it is likely that the poor scaling of the local scratch approach cf. SFS for lower process/node counts will extend to higher process/node counts. This is due to written files becoming too large to be cached, with the result that performance of subsequent reads is likely to degrade considerably. We expect to observe less degradation in performance, and therefore less difference between CPU and wall times, for runs using SFS;

- Cutting-edge problems of real scientific interest are likely to involve larger basis sets (and/or more atoms) than our test case, and therefore further, larger tests are likely to be more informative and relevant for the user community.

Discussion and Conclusions

This study demonstrates Lustre™-based storage accessed via a fast InfiniBand fabric is not only useful for rapid access to permanent and shared files but can also be used by certain I/O intensive CAE and CC applications for scratch I/O purposes. In these cases the additional costs incurred when deploying an InfiniBand-based Lustre™ infrastructure are offset by the avoidance of installing and maintaining additional directly attached disk arrays and/or additional internal disks for each compute server. (Moreover, an architecture featuring DAS-JBOD arrays will tend to be underutilized if some of its workload features applications with

no scratch I/O needs, whereas as the HP SFS will see higher utilization as it provides for permanent, shared storage requirements as well.)

Further benchmarking studies for these and other applications from areas outside of CAE and CC are required before we are in a position to carry out a thorough, quantitative total cost of ownership (TCO) analysis, but these initial studies do suggest that an InfiniBand-based implementation of Lustre™, in this case HP SFS, is capable of providing a high performance, cost-effective alternative to the storage architectures often deployed at HPC sites today.

Of course the number of simultaneous I/O intensive processes/jobs is limited by the speed of the centralized storage server with this approach. However, such object-based storage generally scales well as one increases the number of OSSes; the architecture is expandable and can be sized to sustain throughput for a larger number of simultaneous CAE and CC runs – performance can be scaled up to tens of GB/s compared to the moderate eight-OSS HP SFS server studied in this paper (but additional capital outlay is required obviously). In future papers we intend to investigate the scalability of runs using smaller and larger HP SFS servers for these and other applications.

Our initial investigations point to the following conclusions, some of which could be strengthened with additional data (including investigations of numerous striping strategies), and analysis (ABAQUS in particular):

- Compared to using internal disks in the servers, HP SFS has the potential to dramatically increase the throughput of I/O intensive applications, such as MSC.Nastran, ABAQUS, and NWChem;

- Compared to using external DAS-JBOD arrays, HP SFS can sometimes match the performance with good throughput for a reasonable number of simultaneous jobs.

Prior to this study, vendors such as HP have typically recommended the OSD approach for large and fast storage in general but not necessarily as a replacement for local scratch and/or DAS-JBOD in the context of high-demand scratch I/O production application environments. For example, to our knowledge PNNL's HP SFS Lustre™ server is not used for such purposes and NWChem jobs similar to the ones presented above are only ever run using local scratch disks. We expect this study will help change perceptions and lead to adoption of the OSD approach for these purposes by HPC sites which deploy thin-client Linux clusters for production work in areas such as CAE and CC.

Acknowledgements

We wish to thank MSC Software Corporation for the use of MSC.Nastran software, ABAQUS, Inc. for the use of their software, and Pacific Northwest National Laboratory (PNNL) for the use of NWChem.

In addition we thank colleagues at HP – Eamonn O'Toole, Gavin Brebner and Mark Koenig – for helpful discussions and guidance.

Council for the Central Laboratory of the Research Councils
Chilton, Didcot, Oxfordshire OX11 0QX, UK
Tel: +44 (0)1235 445000 Fax: +44 (0)1235 445808

CCLRC Rutherford Appleton Laboratory
Chilton, Didcot,
Oxfordshire OX11 0QX
UK
Tel: +44 (0)1235 445000
Fax: +44 (0)1235 44580

CCLRC Daresbury Laboratory
Keckwick Lane
Daresbury, Warrington
Cheshire WA4 4AD
UK
Tel: +44 (0)1925 603000
Fax: +44 (0)1925 603100

CCLRC Chilbolton Observatory
Drove Road
Chilbolton, Stockbridge
Hampshire SO20 6BJ
UK
Tel: +44 (0)1264 860391
Fax: +44 (0)1264 860142