PaN-data ODI

Deliverable D8.2

D8.2: Evaluation of Parallel file systems and MPI I/O implementations (M9) - Report

Grant Agreement Number	RI-283556
Project Title	PaN-data Open Data Infrastructure
Title of Deliverable	D8.2: Evaluation of Parallel file systems and MPI I/O implementations (M9) - Report
Deliverable Number	D8.2
Lead Beneficiary	STFC
Deliverable Dissemination Level	Public
Deliverable Nature	Report
Contractual Delivery Date	01 Jul 2012 (Month 6)
Actual Delivery Date	23 Aug 2012

The PaN-data ODI project is partly funded by the European Commission under the 7th Framework Programme, Information Society Technologies, Research Infrastructures.

Abstract

This document D8.2: contains and evaluation of Parallel filesystems and MPI I/O implementations (M9) - Report

Keyword list

PaN-data ODI, Scalability

Document approval

Approved for submission to EC by all partners on 23.08.2012

Revision history

Issue	Author(s)	Date	Description
1.0	Bill Pulford	17 July 2012	Complete version for discussion
1.1	Nick Rees, Ulrik Pedersen	10 Aug 2012	Updates to sections 1-4
1.2	Frank Schluenzen	22 Aug	DESY additions

Acknowledgements:

Mark Basham, Nick Rees, Tobias Richter, Ulrik Pedersen (Diamond Light Source)
Heiner Billich (PSI/SLS)
Hedges, Keith Fitzgerald, Mark Gary, D. Marc Stearman **(**Lawrence Livermore National Laboratory) see text

Table of contents

Page

1 Introduction

The processes of Data Acquisition and Analysis at large facilities, particularly synchrotrons, often involves the reading and writing of many extremely large data files potentially occupying up to petabytes of storage space. Depending on the experimental technique detectors may be used that can provide high data rates in terms of both numbers of files and file sizes.

Specification	PCO 4000	PCO-Edge	Pilatus 6M	Excalibur
Frame	2D	2D	2D	2-3D
Scan Size	1D	1D	1-3D	1-2D
Data Rate	100MB/s	700MB/s	160MB/s	600MB/s
Status	Complete	In development	complete	In development

Table 1 – Examples of the detectors used at the Diamond Light Source indicating the challenge for the underlying file system. Some detector systems are being built, to utilize multiple, stacked PCO detectors synchronously at full data rate

The above table provides examples of the higher point to point data rates expected to be supported by the file system at Diamond Light Source. Additionally many other experiments that have modest data rates can produce very large numbers of small files and experience has shown that this can also provide very significant challenges for the supporting file system. Most large facilities operate many experiments in parallel thereby multiplying the bulk input and output requirements. Data processing and analysis frequently is done while data are being taken and being transferred to storage and archive, which results in additional concurrent read and write operations. Concurrency of the read and write processes should never impair an on-going experiment.

Clearly speed of reading and writing files (both in terms of throughput and fixed overheads) is of key importance but there are many additional properties that are important to the overall performance of the file system used: These include:

a) Speed of access to any file.
b) Directory and file creation and removal time.
c) File metadata access time.

References in table 1:

1. PCO 4000 and PCO Edge are high speed cameras developed and supplied by PCO AG,Donaupark 11, 93309 Kelheim, Germany
2. Pilatus 6M is a very high capability detector developed and supplied by DECTRIS Ltd. Neuenhoferstrasse 107,5400 Baden, Switzerland
3. Excalibur is an ongoing detector development collaboration between the Science and Technology Facilities Council and Diamond Light Source

2 Meetings and workshops

There have been a number of workshops and solution evaluations addressing the capabilities of file systems and effective solutions to implement parallel processing. The most pertinent and recent were:

a) 27-28 Feb 2012 - Joint HDRI & PaNdata workshop at DESY, Hamburg

Abstract – *"The joint PaNdata ODI and PNI-HDRI workshop at DESY covers a number of topics of common interest, like GPU acceleration for the analysis of specific experiments, MPI i/o, analysis frameworks and data management related services and policies. This was followed by an informal visit by Diamond staff to the Swiss Light Source on the 29 Feb to start the process of reviewing the performance of the technical solutions."*

b) 29-30 May 2012 - Efficient use of HDF5 with high data rate x-ray detectors at the Swiss Light Source.

Abstract – *"Current and future x-ray detectors - like the Eiger detector developed and build at PSI - deliver data at sustained rates of several times 10Gbit/s. The HDF5 library and Nexus data format are widely accepted solutions for data storage at synchrotron and x-ray free electron light sources.*

The two day workshop and course will focus on the efficient storage of x-ray detector data at rates of 10-100Gbit/s and beyond in HDF5/Nexus files. The workshop will give a detailed and accurate presentation of the HDF5 library's current capabilities and limitations to write detector data at high rates."

The scope of the meetings was wider that than file systems and parallel processing and extended to the consideration of file formats that they should support in the most efficient way; this is reported on in D8.1 - Definition of pHDF5 capable Nexus implementation (M9).

3 Technical Solutions

The current economic disk technologies although increasingly advanced are not able to provide appropriate characteristics to fulfil the above requirements when employed in simple modes. Solutions to fulfilling the requirements are generally based on configuring the disks in parallel so that file access, reading and writing is distributed across many devices simultaneously.

One conclusion of the meetings 2.1a and 2.1b supported the widely held view that there are a relatively small number of potential technologies. At present two clear candidates provide well supported leading edge parallel file systems suitable for supporting very high speed data streams from current and future detectors:

a) GPFS (http://www-03.ibm.com/systems/software/gpfs)
b) Lustre (http://wiki.whamcloud.com/display/PUB/Wiki+Front+Page)

Both 3a) and 3b) have been shown to provide appropriate support for data acquisition and analysis at large synchrotron facilities. Furthermore, they are complementary in that the former is a leading commercial solution and the latter a leading open source one. There was an additional requirement of D8.2 to provide a more detailed comparison between the two systems and thus to provide guidance for collaborating facilities where technical choices remain to be made. It emerged that much of this work has however been performed already by Lawrence Livermore (LLNL) in the following documents:

- http://www.pdsi-scidac.org/events/PDSW10/resources/posters/parallelNASFSs.pdf
- https://e-reports-ext.llnl.gov/pdf/457620.pdf

Whilst these evaluations were conducted around 18 months ago, they have largely been reproduced on the production versions of these file systems running at our facilities. The reader is encouraged to read the papers but the conclusions are reproduced here to provide some continuity to the discussion.

Though Lustre and GPFS are particular prominent implementations, quite a few filesystems are currently rapidly emerging or being developed. The *Fraunhofer Global Filesystem* (fhgfs), *Parallel Virtual File System* (pvfs), *Write Anywhere File Layout* (wafl; which is more an abstraction of a filesystem than a file system per se), *hadoop* or *glusterfs* are just a few examples of new implementations or implementations supporting new protocols. NFS 4.1 for example has several new features and enhancements compared to NFS 3. There are amazingly few file systems or vendors currently supporting NFS 4.1. One of the few systems implementing NFS 4.1 on a wide spectrum of clients is dCache, which in addition permits transparent migration of data from high to low speed media (i.e. tape) and vice versa. In the appendix a few of these systems and implementations or protocols are compared for meta-data operations and realistic applications.

3.1 Conclusions from Lawrence Livermore National Laboratory (LLNL)

Full acknowledgement to **Richard Hedges, Keith Fitzgerald, Mark Gary, D. Marc Stearman** of the LLNL

1) Both file systems are able to drive hardware at high rates.
2) GPFS has the advantage for throughput tests due to larger blocksize, and since Lustre has the additional overhead of internal check-summing.
3) Lustre is generally significant faster for the metadata operations that are most important in our workload: operations where data cannot be locally cached on the client.
4) GPFS is better able to spread data over multiple servers and to use multiple cores in the client.
5) Where metadata can be cached on the client, GPFS will have an advantage.
6) For stating a large number of files in a shared directory GPFS can beat Lustre, with the advantage increasing as the number of files increase.
7) File creations in a shared directory have been special optimized algorithm for GPFS with “fine grained directory locks”.

3.2 Tests performed between PSI and Diamond during the meeting 2b) above

Some tests of GPFS and Lustre were performed at the Swiss Light Source during the 29-30 May workshop, and subsequently. Both of these used similar hardware (S2A9900 disk arrays from Data Direct Networks). The throughput tests were conducted with the ior benchmark and showed that GPFS could easily saturate a 10 GbE link with a single writer (read and write performance over 1150 MB/sec) whilst Lustre was limited to around 400 MB/sec. In addition the utility mdtest, an MPI-coordinated metadata benchmark test that performs open/stat/close operations on files and directories, was also run and the results are shown in table 2 below.

Operation	Max GPFS/Lustre **Ops/sec**	Min GPFS/Lustre **Ops/sec**	Mean GPFS/Lustre **Ops/sec**
Directory creation	512.6/2742.0	137.9/1896.0	275.2/2254.1
Directory stat	1895.3/3690.9	1501.5/2877.2	1644.822/3296.0
Directory removal	181.8/1824.3	140.6/1578.0	156.6/1685.3
File creation	1338.4/1663.3	151.5/1113.262	612.5/1439.3
File stat	5765./906.6	2081.0/79.6	4243.8/331.6
File removal	1786.5/2338.5	1066.5/1487.0	1366.3/1766.2
Tree creation	1807.6/2683.3	156.2/2196.8	1070.9/2430.8
Tree removal	1049.6/1910.5	626.2/1455.6	809.5/1750.4

Table 2 – Comparison of file operation speed between GPFS and SLS and Lustre at Diamond
Commands used:

- mdtest -I 10 -z 5 -b 5 -i 5 -u -d /sls/XBL/da for SLS
- /home/bnh65367/code/mdtest/mdtest -I 10 -z 5 -b 5 -i

These overall results broadly support the more detailed analysis performed at LLNL. However, these tests were done with Lustre 1.8, and the LLNL tests used Lustre 1.6. We are planning on repeating the tests with a Lustre 2.1, since the Lustre client was largely re-written as part of the Lustre 2.0 development and is reported to have better performance.

3.3 Final comments

a) Both technologies are capable of fulfilling the requirements set out in the introduction.
b) GPFS does seem to have the edge in a number of areas in the general case. However more recent versions of Lustre are reported to have better performance.
c) GPFS is an IBM commercial product and it is potentially expensive to deploy on a large site.
    a. The pricing and maintenance structure is not clear for non profit making organizations.
d) Lustre is open source and offers the possibility to be more economical to deploy.
    a. Practically a maintenance contract has been found to be required to optimize stability of the system in operational mode. This can be nevertheless a significant cost.

4 Practical Details of Hierarchical Data Format Implementation

4.1 Introduction:

The data acquisition and analysis processes require experimental data to be written and read from files stored in a stable and well understood format. In this case these files and format must be optimized to be supported by the file systems discussed in the previous sections. Ideally the functionality should include low level file access software being able to exploit the highly parallel natures of the file systems. The Hierarchical Data Format version 5 (hdf5) has been adopted by the PAN-Data project to fulfill this role.

4.2 Advantages:

HDF5 has been available for some time within the High Performance Computing (HPC) community and has provides a stable overall file format sufficiently general to be used for most scientific applications. The application programming interface is well documented and supports a rich feature set including:

- Reading and writing a basic portable, hierarchically structured data file.
- Data compression and decompression cycles may be used at run time.
- Data types can be defined and changed within the stored files.
- Data can be filtered prior to reading and writing.
- The layout of the file on the storage media can be defined flexibly including non-linearly.
    - This is useful where data may be contributed from a number of asynchronous processes.
- It has a variety of file storage back ends, including one that allows coordinated parallel I/O between machines and processes using the MPI library.

4.3 HDF5 Problems:

HDF5 while benefitting from the advantages outlined in 4.2 does have a major disadvantage that limits the flexibility with which its functionality can be exploited. As a consequence of its long availability key parts of the core code are non-reentrant and single threaded, with a global lock to protect against multi-threaded use. HDF support report that changing this is very difficult and estimated to require 6 man years of a good programmer. The consequences are:

- The parallelization it currently offered must be process based (not thread based).
- MPI (Message Passing Interface) libraries are used for communication that is also process based.

An additional problem is that parallel writing and compression are currently mutually exclusive, since the parallel writing relies on the layout on disk being predictable.

- Compression usually needs a predictable storage location of the data on disk. This is rendered difficult by the flexibility of defining the storage layout.

Parallelizing the compression itself is not possible either, though a multi-threaded implementation could comparably easily enhance the HDF5 performance without major interference with the global HDF5 layout. Discussions about the particular implementation of generic mulit-threaded compression algorithms and supporting such developments are on-going between HDF5.org and PaNdata partners.

4.4 Parallel HDF5 (pHDF5) implementation

Having adopted HDF5 as a standard file format, significant work has been done at a number of facilities to extend the technology to be able to read and write within a parallel computing and network architecture. The following discussion is mainly based on work done at Diamond but should be complementary to that done at many collaborating facilities.

4.4.1 Targeted Computing Platforms and associated notes

a) Linux (Red Hat Enterprise Linux versions version 5 or greater) 64bit with native Lustre client support is the preferred platform where there are no constraints on hardware particularly imposed by the data source such as a detector. Modern CPU-architecture could impose additional restrictions for MPI or MPI I/O utilizing applications (e.g. optimal performance on AMD interlagos multicore system can only be achieved with most recent compiler and kernel versions).
b) Microsoft Windows XP, Server 2003 and Server 2008R2 are used when the supplier only provides Windows binary drivers for their hardware.
    a. Unfortunately several detectors only vendor supported with Windows binary drivers
    b. There are a number of examples such as PCO
    c. The combination of MPI and pHDF5 on the Windows platform is not officially supported although some have reported making it work. At DLS it has been decided to use Linux as the platform for running the pHDF5 writer as a service – and to transfer detector data from Windows based systems via TCP.
c) Diamond Light Source (DLS) use the EPICS controls system for accelerator and beamline system consequently the EPICS area detector (EPICSAD) is used as the default large detector interface.
    a. This has been implemented to run on all of the above mentioned platforms – see a) and b)
    b. The significant advantage of this strategy is that detailed access to the data source is abstracted as much as possible from the EPICS Area Detector API and in most cases the actual writing of the data files can be delegated to EPICSAD.
d) Notes on Windows support:
    a. Using the 32 bit architecture platform is limiting, providing poor network performance on a modern 10Gb network
    b. The software and drivers supplied can us obsolete platforms and tool-chains causing problems building and linking recent libraries
    c. A good current target tool chain is MSVC 2010 which supports the chosen standard Windows platform: Server 2008R2
    d. Current laboratory results with 64 bit Windows-Linux over a 10Gb network are promising giving round 95% of the available bandwidth on simple TCP data transfer between servers and enable up to 300MB/s HDF5 file writing (single process, samba share)

4.4.2 Some Examples of Data Acquisition Architectures for complex and high throughput detector systems at Diamond Light Source

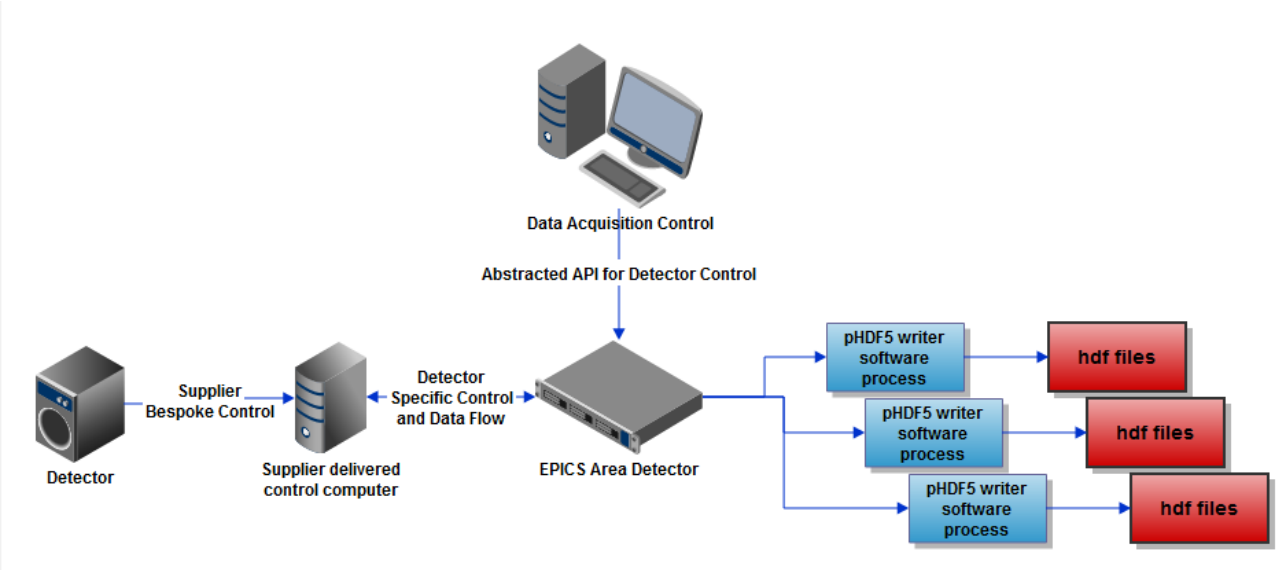


Fig 1 – A very high data rate detector needing to write successive files concurrently (eg. Pilatus 6M). The EPICS area detector is shown as a separate server for clarity of login but in practice the software is usually run on the Data Acquisition Control or even on the supplier delivered computer if the technology is compatible.

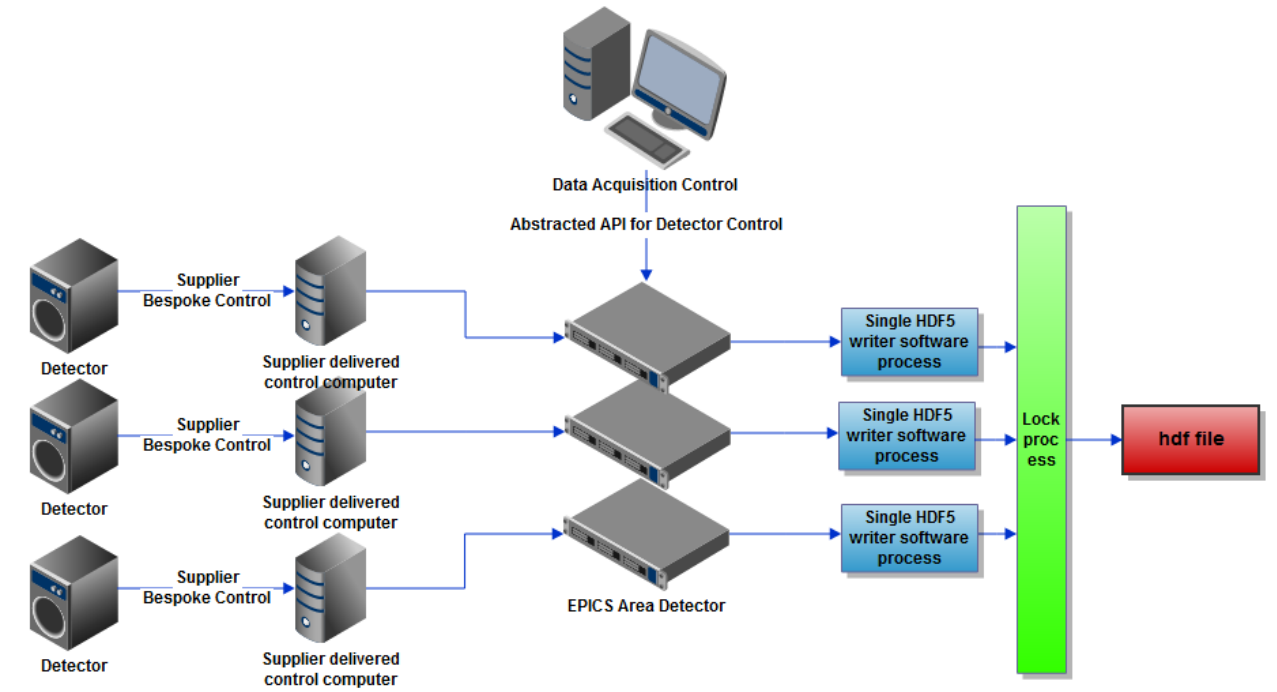


Fig 2 – A complex detector with multiple elements needing to concentrate the data collected into a single file. A lock process is required to arbitrate low level file access due to the need for process based parallelization

The EPICS area detectors is shown as separate servers. In practice the software is usually run on the Data Acquisition Control or detector control computer.

4.4.3 The initial requirements pHDF5 writing and reading

4.4.3.1 Writing pHDF5

Section 4.4.2 gives some examples of the architecture and strategy used to write high speed and data volume HDF files at large facilities. The overall concept is that the specific functionality of the detector is delegated to the suppliers' delivered drivers and the data acquisition control interacts with an abstraction layer (AL) responsible for both interfacing with the suppliers' software and directly writing the data files. The abstraction layers at many EPICS based facilities would be the EPICS Area Detector; at TANGO based facilities such as the ESRF it may be LIMA.

It is evident that this AL will be the primary user of libraries to write pHDF5 files.

4.4.3.2 Reading pHDF5

In many cases the normal HDF5 client software will be used for file access. The conventional use case for parallel reading is for computing cluster access to HDF5 files where processes on each node would use MPI to mediate the file access.

4.4.3.3 Perceived requirements for optimizing pHDF5 access

The following list is the result from the increasing pool of experience gained while using the HDF5 software version 1.8.7 which was the current version at the beginning of the project (work has begun to update to the most current release 1.8.9):

a) Generally the software should be scalable to support future detectors with increasing data rates
   - pHDF5 run as MPI jobs so should scale easily providing they are run as multiple processes
   - Each process write a slice of data into appropriate locations in the HDF5 file
b) Always use the documented best practices in the use of HDF5 libraries.
   - Example "Tuning HDF5 for Lustre File Systems"
c) Significant improvements may result from focusing on optimizing performance on parallel file system: Examples for Lustre:
   - Optimize the overall performance of the file system
   - Align chunk boundaries with Lustre boundaries. (A chunk is an n-dimensional subset of a dataset, and represents how the file is physically laid out on disk)
   - Select a suitable stripe count to match the number of writer nodes

- Ideally do write transactions of a size which is an even multiple of the selected Lustre stripe size

d) Implement user configurable chunking
   - Selecting a chunking scheme is often a balance between write and read performance and optimization is highly dependent on the data access patterns when writing and reading. Users need to be able to select an appropriate chunking scheme for their data – but probably require expert advice on what is efficient in terms of performance.

e) Set up the ability to store N-dimensional datasets
   - As an example, 2-3 Dimensions per frame + M scan dimensions

f) Implement the use of extendible datasets that grow as new data arrives
   - This is very useful but can have a negative impact on performance.

g) Particularly in the case of detectors insert a blank record (frame) to emulate a valid overall file to handle dropped frames resulting for the low level hardware.

h) Implement configurable definition of the file structure.
   - The provision of an XML configuration description would allow simple raw datasets or facilitate more complex layouts such as defined by NeXus.

i) Optimize the design Data Acquisition Control so that metadata are available to be written at the same time as the raw data to the HDF file. In some cases it might be favorable to separate data streams for images and metadata, since writing metadata is always a serial process, and join the streams after completion of the processes.

j) Despite the description in 4.4.3.1 ensure that the coupling between detector data and file is sufficient flexible to allow integration with other control frameworks (or ad hoc solutions if necessary)
   - The pHDF5 writer is designed as a stand-alone cluster application using MPI. It does not depend on any one control system. Data is transferred to the pHDF5 writer via TCP so it can integrate with any control system or stand-alone application. A small C++ client library is in development to aid integration with control systems

k) The pHDF5 software should include the following properties:
   - Compatible with Linux based MPI application
   - Must integrate well with Windows based detector systems. TCP is currently used at Diamond. European XFEL is investigating UDP based image recording to optimize point to point data transfer rates.
   - Ensure that it can run as network service where appropriate.

4.4.3.4 The current status of the implementation(s)

The items in the list below concentrate of the status at Diamond.

a) A Single process HDF5 writer already available as part of the EPICS areaDetector distribution since version 1-7.
   a. It runs natively on Linux and Windows as an areaDetector plug-in
   b. The performance is adequate for a number of detectors in use at DLS
   c. As outlined in D8.1, developments of NeXus APIs and interfaces between DAQ, controls and NeXus/HDF5 (Nexus Data Server) are under development.

b) A parallel HDF5 file writer application in active development
   a. The initial design and some development has been done

   b. An initial prototype is running but is lacking many features
c) Integration of Windows based detector system
   a. TCP transfer of N dimensional arrays with attached metadata functional.
   b. The prototype implementation uses the new EPICS CA4 protocol which has been implemented in a small stand-alone library and without any dependency on EPICS base.
d) HPC-testbeds with a wide spectrum of filesystems attached are availabe for tests and bencmarking. A number of HDF5 datasets in various formats and sizes are available for testing pHDF5 in analysis workflows.

4.4.3.5 Required Validation tools

1) Benchmark Systems

a) Must be able to measure pHDF5 writer application performance
   a.1. Timestamps are taken at each data transfer and stored locally the output HDF5 file for later analysis if required
b) Must be able to measure parallel file system performance
   b.1. The current proposal is IOR which should be able to match the I/O pattern used for detectors. i.e. Writing multiple 2D frames into 3D datasets.
   b.2. iozone and h5perf might be complementary benchmarks to pinpoint performance bottlenecks.
c) Must be simple for systems administrators to re-run benchmarks to verify performance when making updates or tweaks to file system or cluster

1) Simulate new detector systems

a) Must be able to produce simulated data at configurable resolution and rate
b) Simulated data should reflect real experimental data e.g. in terms of compressibility.
c) Must be available for decision support in determining requirements for computing resources when investigating or purchasing new detector systems
d) Verify file system performance related to a new detector system

4.4.4 Longer term efficiency improvements proposed for HDF5

A number of longer term improvements to the software and support of HDF5 were proposed in the “Efficient use of HDF5 with high data rate x-ray detectors” meeting held at the Swiss Light Source (meeting 2b).

a) Allow the user to create their own dynamically loadable filters to improve compression performance for special applications.

b) Allow the user to provide pre-compressed data chunks to be written by the library.

c) Implement a second approach to compressed parallel file writing using sparse files and pre-allocate the data chunks at their un-compressed size but only write the data at the compressed size. This would require sparse file support for all the file utilities.

d) Implement a meta-data server to reduce the constraints on parallel I/O. This would allow the disk layout to be more flexible and allow parallel writers to write compressed chunks without having to require sparse file support.

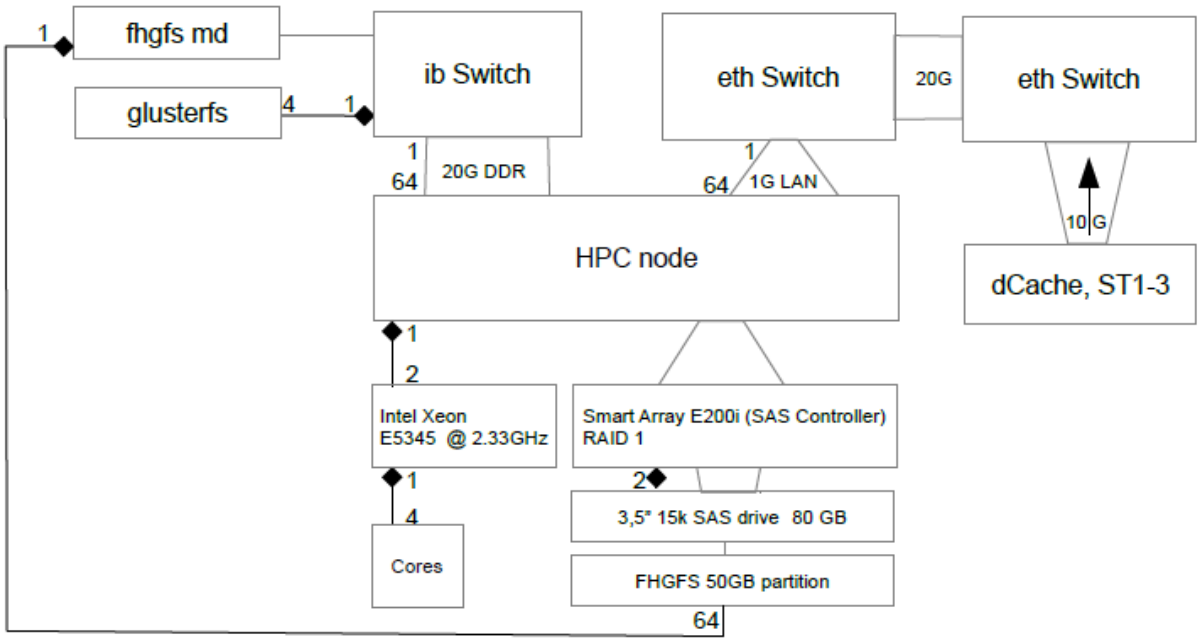
5 Appendix File system performance reported by DESY

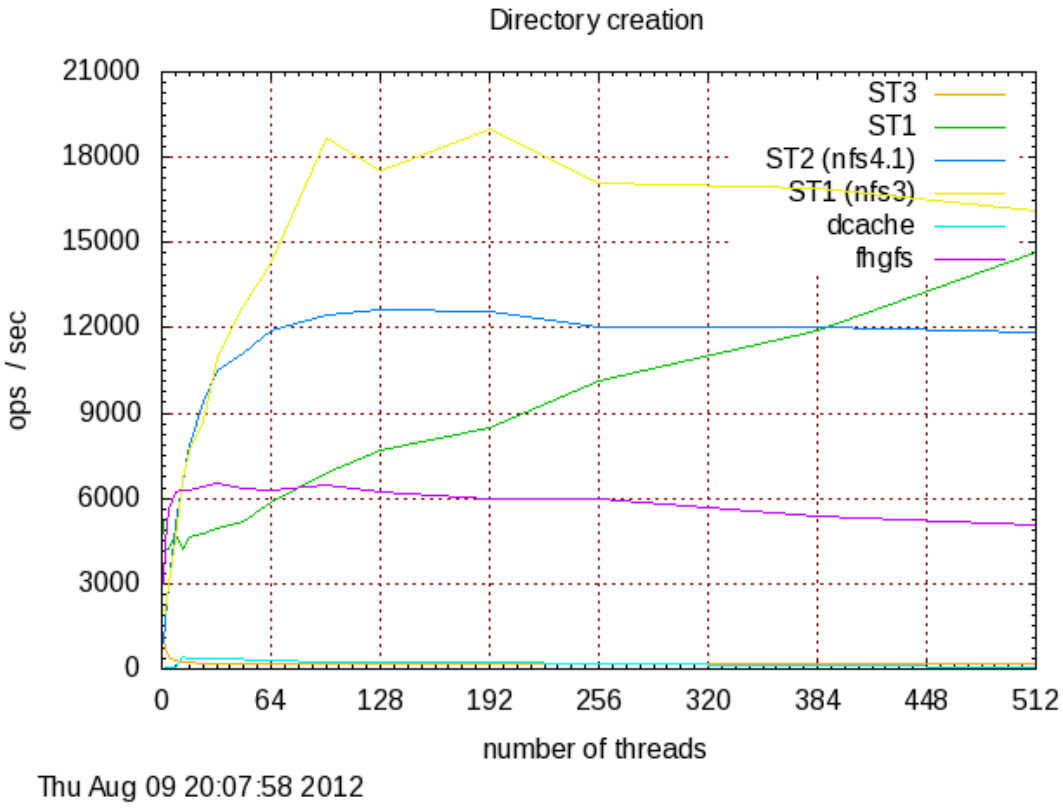
5.1 Name: mdtest

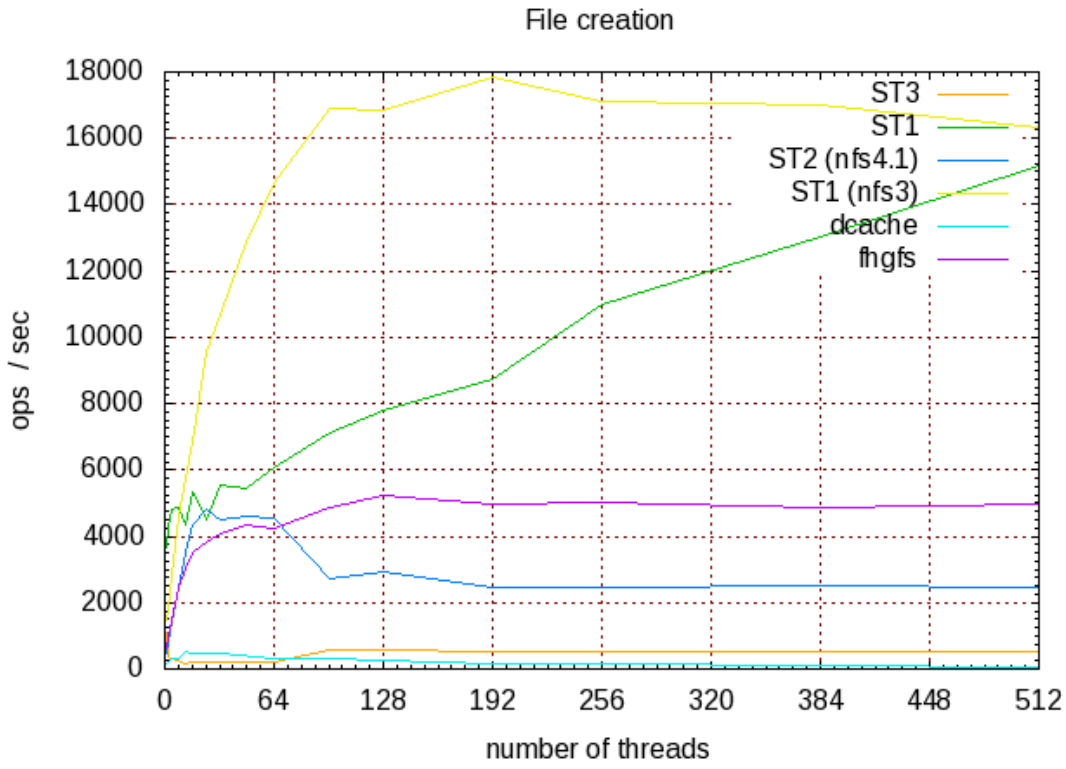
Version: 1.8.3
Url: http://sourceforge.net/projects/mdtest/
Description: mdtest is an MPI-coordinated metadata benchmark test that performs
OS: Scientific Linux 6.2
Kernel: 2.6.32-279.1.1.el6.x86_64
MPI: openMPI 1.5.3
Platform: DESY-HPC 2011

Description	Type	Capacity / TB	Protocol
dCache PS 2011	dCache	10.000	NFS 4.1
fhgfs 2011	FHGFS (ipoib)	3.2	FHGFS 2011.04.r16
ST1	WAFL	20	NFS 3
ST2	WAFL	40	NFS 3
ST2	WAFL	4*10	NFS 4.1
ST3	GPFS	443	NFS 3
Fhgfs	FHGFS (ipoib)	73	FHGFS 2011.04.r19
Glusterfs	Glusterfs (rdma)	73	3.2.6

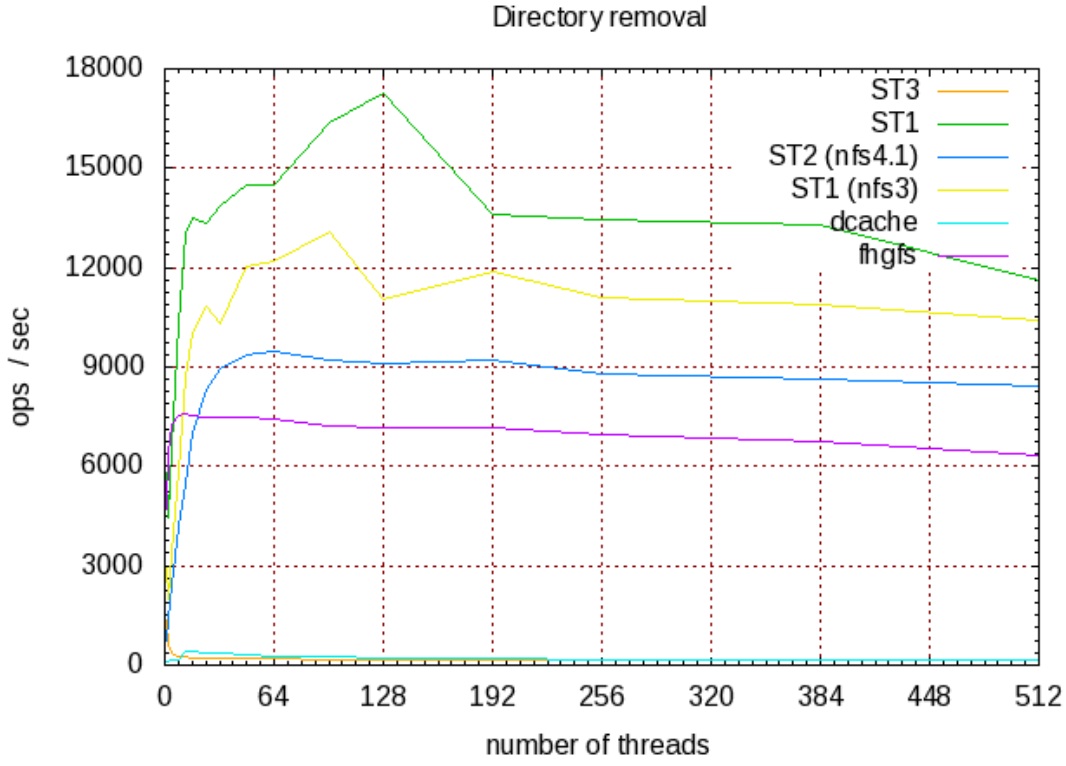
Topology:



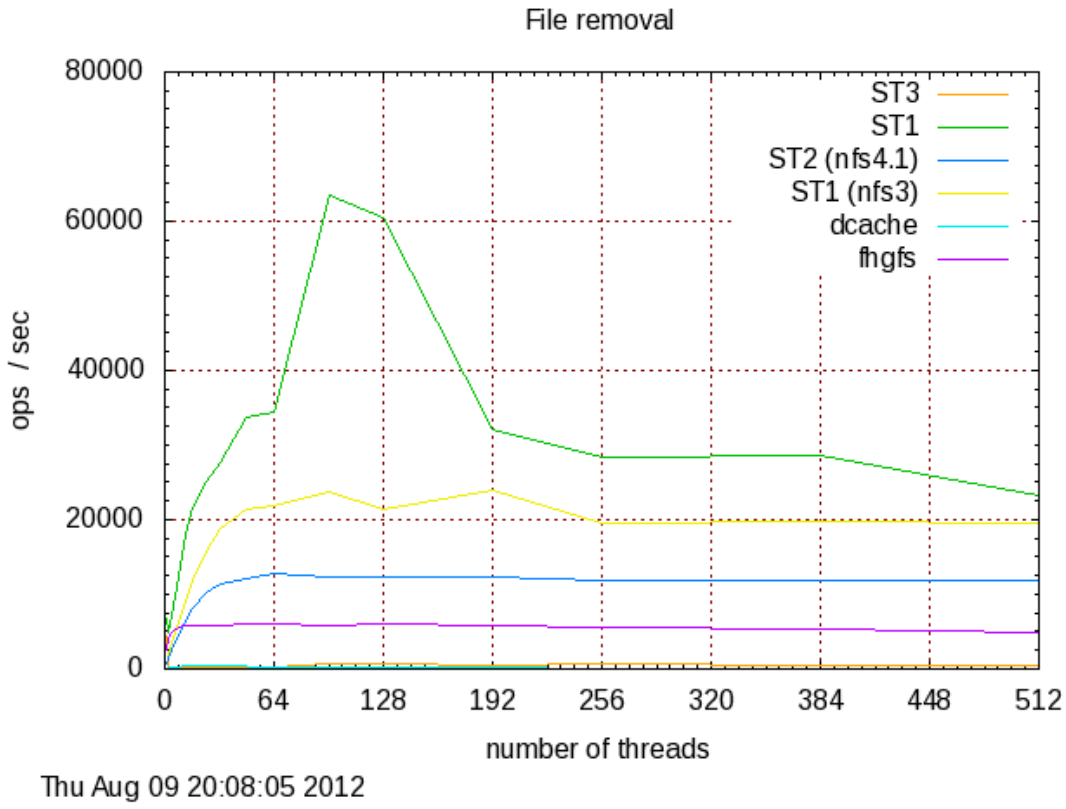

Directory creation
21000
18000
15000
12000
9000
6000
3000
0
ops / sec
0
64
128
192
256
320
384
448
512
number of threads
ST3
ST1
ST2 (nfs4.1)
ST1 (nfs3)
dcache
fhgfs
Thu Aug 09 20:07:58 2012

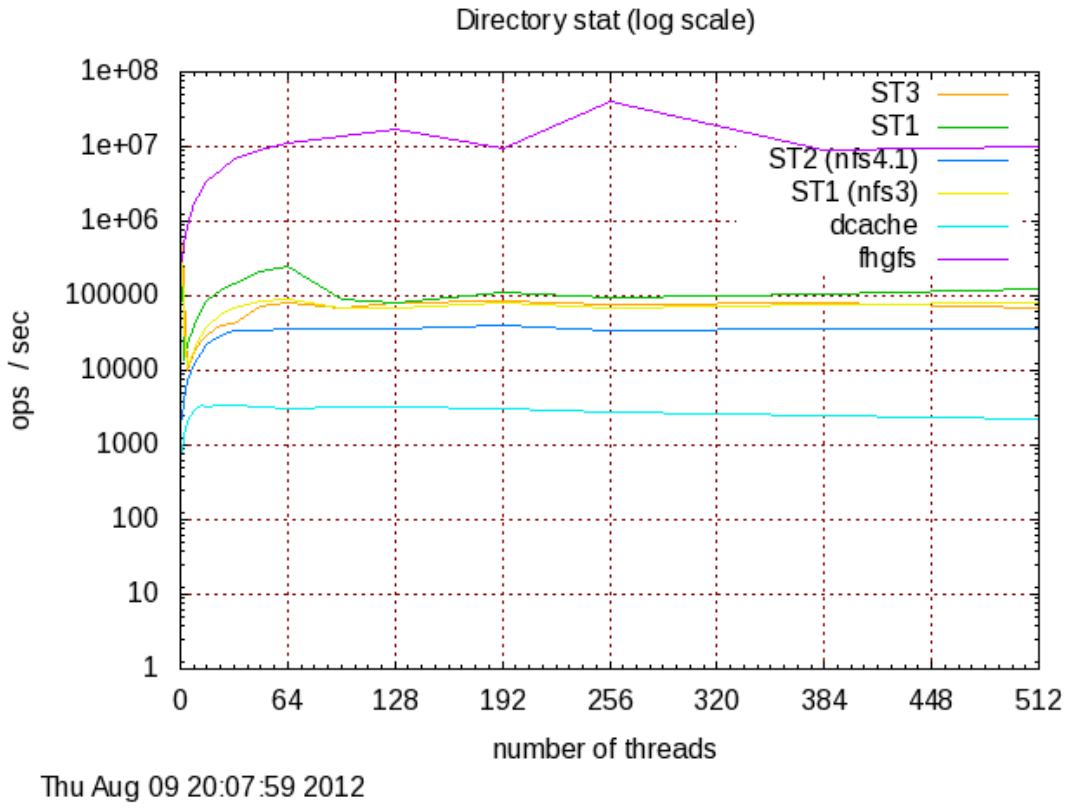


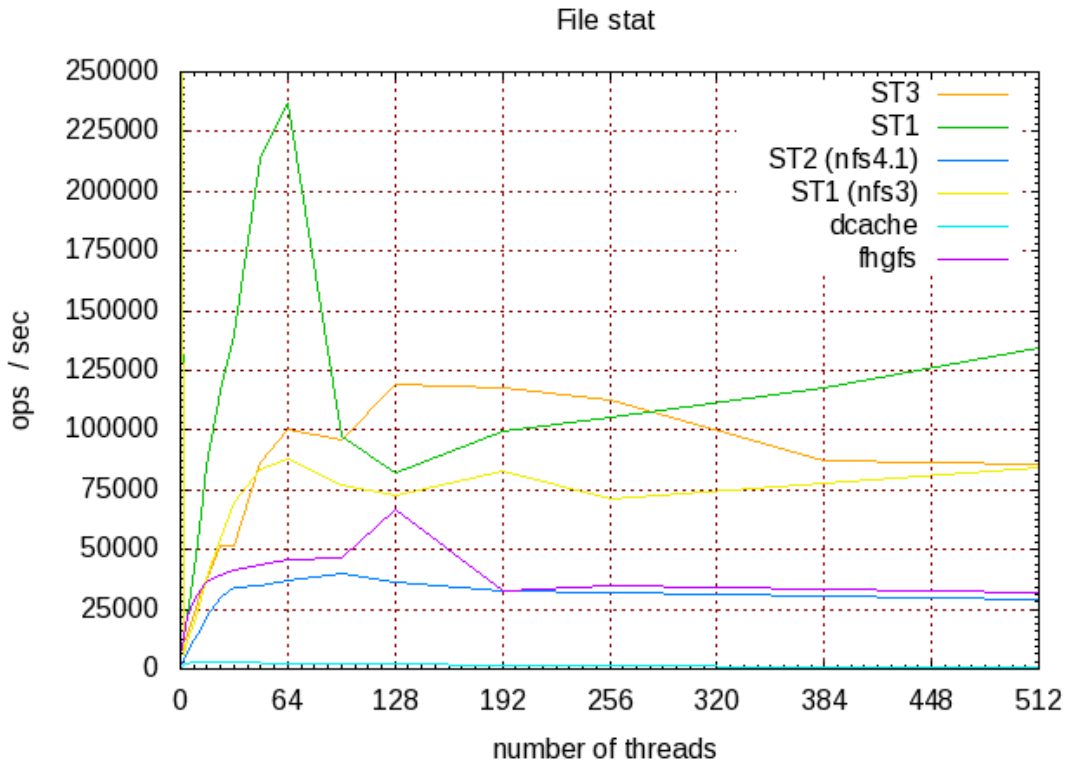
Thu Aug 09 20:08:02 2012



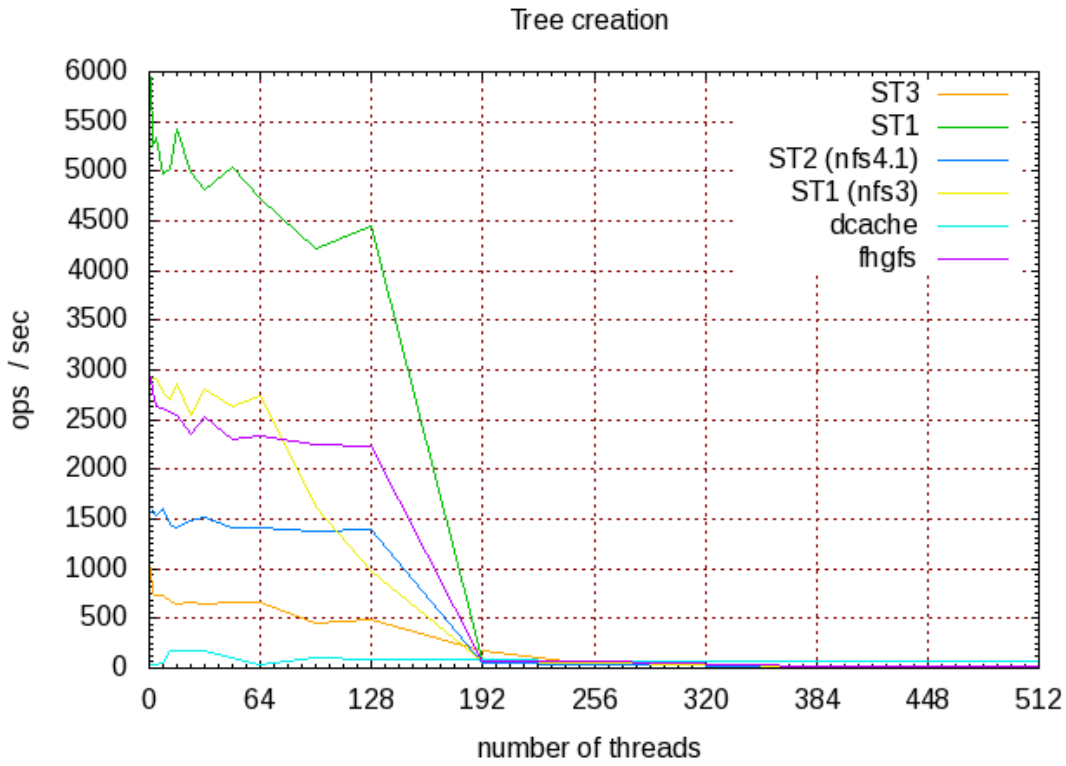
Thu Aug 09 20:08:01 2012


File removal
80000
60000
40000
20000
0
ops / sec
0
64
128
192
256
320
384
448
512
number of threads
ST3
ST1
ST2 (nfs4.1)
ST1 (nfs3)
dcache
fhgfs
Thu Aug 09 20:08:05 2012

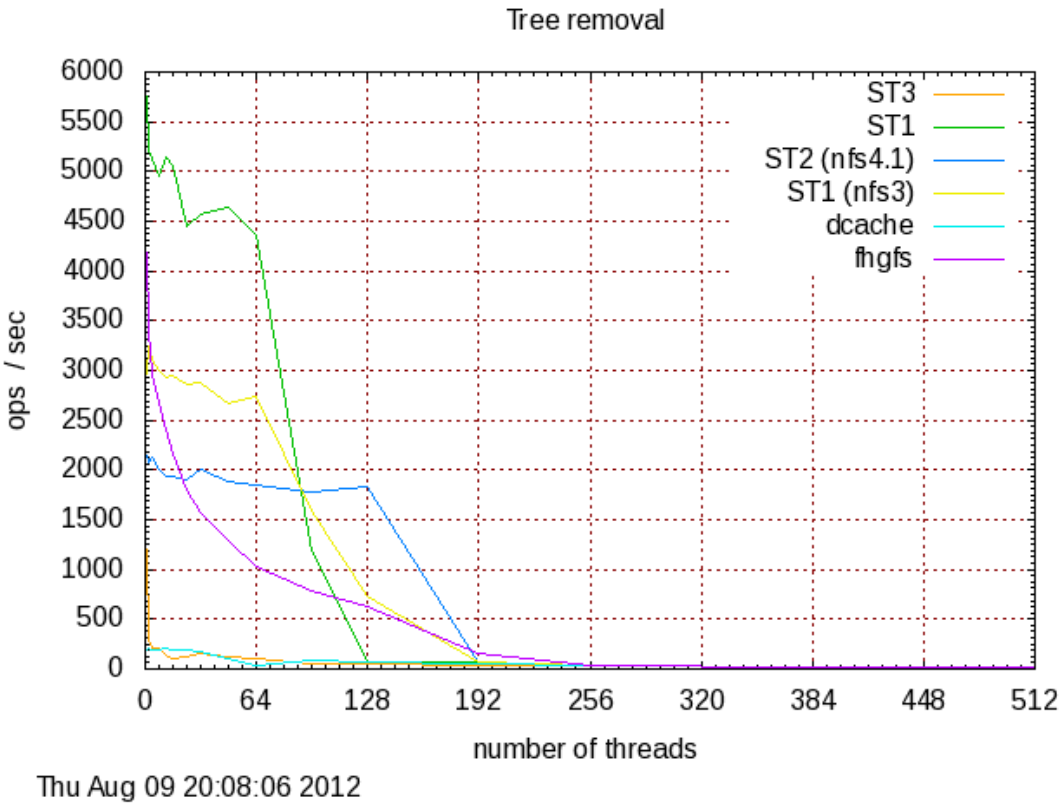

Directory stat (log scale)
1e+08
1e+07
1e+06
100000
10000
1000
100
10
1
ops / sec
0
64
128
192
256
320
384
448
512
number of threads
ST3
ST1
ST2 (nfs4.1)
ST1 (nfs3)
dcache
fhgfs
Thu Aug 09 20:07:59 2012



Thu Aug 09 20:08:04 2012



Thu Aug 09 20:08:08 2012



5.2 Name: pilatus 6M Simulation

Description: pilatus 6M detectors operate at framerates of up to 10-25Hz. This test simulates on 4000 real diffraction images the data transfer (total volume ~12GByte), at
Frame rates of 1-50Hz. A single data stream is utilized.
OS: Scientific Linux 6.2
Kernel: 2.6.32-279.1.1.el6.x86_64
Platform: DESY-HPC 2011
Img Format: CBF
Storage(s):

Description	Type	Capacity / TB	Protocol
dCache PS 2011	dCache	10.000	NFS 4.1
fhgfs 2011	FHGFS (ipoib)	3.2	FHGFS 2011.04.r16
ST1	WAFL	20	NFS 3
ST2	WAFL	40	NFS 3
ST2	WAFL	4*10	NFS 4.1
ST3	GPFS	443	NFS 3
Fhgfs	FHGFS (ipoib)	73	FHGFS 2011.04.r19
Glusterfs	Glusterfs (rdma)	73	3.2.6

Results:

- Single stream is limited to 1G bandwidth on this platform, except for fhgfs (20G ib).
- All file systems can easily cope with frame rates of 20Hz, beyond could not be tested due to the limitations.
- Most file systems can saturate the 1G link, e.g. meta-data handling is not an issue.

- Tests should be repeated on a 10G infrastructure.