# PaNdata ODI

## Deliverable D8.4

## **Draft: D8.4** Examination of Distributed parallel file systems (Month 21)

| | |
|---|---|
| Grant Agreement Number | RI-283556 |
| Project Title | PaN-data Open Data Infrastructure |
| Title of Deliverable | D8.4: Examination of Distributed parallel file system (Month 21) - Report |
| Deliverable Number | D8.4 |
| Lead Beneficiary | STFC |
| Deliverable Dissemination Level | Public |
| Deliverable Nature | Report |
| Contractual Delivery Date | 01 Jul 2013 (Month 21) |
| Actual Delivery Date | 17 November 2013 |

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

## Abstract

Implementation of pNexus and MPI I/O on parallel file systems (Month 21)

**Keyword list**

PaN-data ODI, Scalability

## Document approval

Approved for submission to EC by all partners on 07.10..2013

## Revision history

| Issue | Author(s) | Date | Description |
|---|---|---|---|
| 01.0 | Bill Pulford | 31 Aug 2013 | Complete version for discussion |
| 01.1 | Diamond co-workers | 02 Sep 2013 | |
| 01.2 | Frank Schluenzen - DESY | 17 Sep 2013 | Comment and suggestions |
| 01.3 | Bill Pulford | 7 October 2013 | Reformat and update |
| 02 | Juan Bicarregui | 20 November 2013 | Changes to deliverable numbering and title to make consistent with DoW |

**Acknowledgements:**
Jon Thompson (DLS), Ulrik Pedersen (DLS), Mark Basham (DLS), Frederik Ferner (DLS), Nick Rees (DLS), Heiner Billich (PSI/SLS), Frank Schluenzen (DESY)

## Table of contents

Page

## 1. Introduction

The PANData ODI project sets out to optimize coordination between research groups working at one or more different large experimental facilities across Europe and with the potential of expanding its scope across the scientific world. There are a number of components to the project such as common authentication, application software and federated searchable data storage systems. This report relates to a joint research activity, Work Package 8 **Scalability,** which concerns standardization of file formats and research to identify supporting data storage architectures to optimize speeds and data storage capacity.

The timeline for this workpackage:

- D8.1: Definition of pHDF5 capable Nexus implementation – Software – Report Delivered Aug 2012
- D8.2: Evaluation of Parallel file systems and MPI I/O implementations - Report Delivered Aug 2012
- D8.3: Implementation of pNexus and MPI I/O on parallel file systems – Report Delivered Oct. 2013
    - (Note that in the WP description there is no D8.4, so deliverables 8.4,8.5,8.6 are numbered 8.5,8.6,8.7 in the WP description.)
- D8.4 Examination of Distributed parallel file system
- D8.5: Demonstrate capabilities on selected applications (Month 21 June 2013) – This report
    - A demonstration application is distributed and is in daily use by many users and at a number of European facilities see DAWNScience.
- D8.6: Evaluation of coupling of prototype to multi-core architectures (Month 27 Dec 2014) - Report - Work continuing in the community.

## 1.1. Meetings and workshops

- NOBUGS Meetings and NOBUGS Workshop – Diamond Light Source, 24-28 September 2012
- Eiger Detector Workshop – SLS, 23-25 January 2013
- Workshop – DESY, 4-6 March 2013
- PANData – LUND, 12-14 March 2013

## 1.2. Scope of the report

The following topics are covered in this report.

1. A case study of the commissioning of an advanced parallel detector
2. Highly parallel detector and file systems Lustre and GPFS
3. Practical Details of Hierarchical Data Format Implementation
4. Implementation details of the file systems architecture to support fast parallel writing of HDF files.

## 2. The challenges provided by current detectors

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.

### 2.1. Examples of detectors at DLS

| Specification | PCO 4000[1] | PCO-Edge | Pilatus 6M[2] | Excalibur[3] | Percival[4] |
|---|---|---|---|---|---|
| **Frame** | 2D | 2D | 2D | 2 - 3D | 2D |
| **Scan Size** | 1D | 1D | 1 - 3D | 1 - 2D | 1D |
| **Frame rate** | 5Hz | 100Hz | 100Hz | 100Hz | 120Hz |
| **Data Rate** | 100MB/s | 700MB/s | ~640MB/s | ~600MB/s | ~5-6 GB/sec |
| **Status** | Complete | In development | Complete | Commissioning | In development |

**Table 1: Examples of the more demanding detectors used at the Diamond Light Source indicating the challenge for the underlying file system.**

The above table provides examples of most demanding detectors from the point of view of data rates and volumes needing to be supported by the file systems at Diamond Light Source. Additionally there are many other experiments that have modest data rates but 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 will run experiments that involve the use of these resource intensive detectors in parallel and it is the challenge to the infrastructure and file systems to support the multiplication in bulk input and output requirements. Moreover data processing and analysis frequently is done while data are being taken and being transferred to storage and archive; this results in additional concurrent read and write operations.

The principal requirement is that on-going experiments should never be compromised by inadequacies of the file system. This implies that the speed of reading and writing files (both in terms of throughput and fixed overheads) and the number of I/O operations per second (IOPS) that the underlying system is capable to perform are of key importance. The detailed properties that are important to the overall performance of the file system used include:

[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 a detector development collaboration between the Science and Technology Facilities Council and Diamond Light Source - Journal of Physics: Conference Series Volume 425 Part 6 J Marchal *et al* 2013 *J. Phys.: Conf. Ser.* **425** 062003 doi:10.1088/1742-6596/425/6/062003

[4] PERCIVAL (Pixelated Energy Resolving CMOS Imager, Versatile and Large) is an Ongoing development project between DESY and RAL / STFC

1. File access, creation and removal performance.
2. Directory creation, removal and traversal performance
3. Metadata access time.

From the experiment point of view, the file creation and bare write performance are of prime importance to secure immediate data storage, in particular since several instruments capture and compose multi-nodal images in non-persistent memory.

Many of these points were discussed in report D8.2 but since then detectors and experimental processes have evolved considerably:

- There has been a 4 fold increase in the Pilatus 6M performance and the Excalibur detector is now in commissioning use.
- The file system must be able to support a number of these enhanced detector systems in parallel use.
- Some detector systems are in active development that will require very advanced technology to exploit their capabilities. One example is the use of multiple stacked PCO detectors that produce data synchronously at full data rate. Even more challenging:
    - The developments by Dectris for a single photon counting detector (EIGER) for X-ray applications. Dectris EIGER[5] is composed of a varying number of modules. EIGER 16M is a 32 module detector with a readout rate of up to 22kHz producing several (asynchronous) data streams with a data rate of up to 200GByte/s.
    - A Pixelated Energy Resolving CMOS Imager – PERCIVAL that is capable of data rates in excess of 5GB/sec. X-ray CMOS imager for synchrotron and FEL applications.

## 2.2. Highly parallel detectors and file systems Lustre and GPFS

The PCO Edge, PCO 4000 and many of the available Pilatus detectors including the 100Hz 6M are now in routine use at Diamond (see report D8.2) with the acquisition and analysis file storage architecture being illustrated in this report below. The commissioning of the Excalibur detector provides an interesting opportunity to examine the use of file systems and optimizing their performance. See section 3.

The current Lustre system in use at Diamond successfully supports the high throughputs required by such beamlines as Macromolecular Crystallography and Tomography where demands approach data volumes of greater than 2TB per day and data write speeds of 600MB/s. Newer advanced experimental techniques coupled with the increasing use of even higher speed detectors such as the Pilatus 6M running at 100Hz frame rate, PCO Edge and possibly Dectris EIGER in the future, have caused a review of our current file system technology. At present we are evaluating in both run and test modes the Lustre (http://www.whamcloud.com/lustre/) and GPFS (http://www-03.ibm.com/systems/software/gpfs) file systems with the intention of optimizing our future strategy in this area.

[5]For a comprehensive overview see for example:
http://indico.cern.ch/getFile.py/access?contribId=36&resId=1&materialId=slides&confId=48618

During the same time frame work is being done to commission the Diamond and STFC developed advanced and highly parallel Excalibur detector. This has presented an opportunity to combine the two activities.

## 2.3. Lustre and GPFS test performance comparison, a DLS example

### 2.3.1. Benchmark criteria

In order to be suitable as data storage area for the new detectors (PCO Edge and new fast Pilatus 6M 100Hz), any new high performance file system must be able to accept data at the maximum rate of 900MB/s from a single 10GigE connected machine writing to a single file.

The performance requirements are:

1. 900MB/s write throughput for single process to single file over 10GigE
2. 3x 900MB/s write throughput for 3x 10GigE clients
3. 1500 MB/s write throughput for a 2x 10GigE LACP bonded client
4. 15GB/s aggregated throughput for a suitable large number of clients connected over Ethernet

Functional requirements:

1. provide a POSIX interface to users
2. fully support POSIX draft 1.e style ACLs
3. can be exported via NFSv3
4. can be exported via CIFS/SMB

Notes for 2.3.2 and 2.3.3:

- IOR - http://sourceforge.net/projects/ior-sio/- is the widely accepted tool for comparing the performance of parallel file systems using POSIX, MPIIO, or HDF5 interfaces

- Link Aggregation Control Protocol (LACP) mentioned below provides a method to control the bundling of several physical ports together to form a single logical channel. See Chapter 4 in http://standards.ieee.org/getieee802/download/802.1AX-2008.pdf for more information.

### 2.3.2. Lustre Test

Using Lustre, writing a 20GB file using IOR over 10GigE or 2x10GigE LACP, the best write throughput achieved was about 650MB/s, which fails to meet the write throughput requirements even for a single 10GigE client. Tests with 2 clients have shown that this can be achieved on two clients in parallel, each writing at about 650MB/s. Writing to multiple files from multiple processes on a single client allowed writes at network line rate of about 1150MB/s for a single 10GigE link or about 2300MB/s for a 2x10GigE LACP link. 2 processes writing into one file each were sufficient to provide more than 900MB/s write throughput.

Tests using Lustre's network test tool lnet selftest also demonstrated that Lustre can use the full network bandwidth

### 2.3.3. GPFS Test

The GPFS version used in this test was GPFS 3.5.0-6 provided as part of the DDN GridScaler setup. The server nodes have been installed using DDNs GridScaler DVDs and have been configured by DDN with DLS system administrators closely observing the initial setup but not all subsequent tuning attempts have been as closely monitored. The GridScaler install is based on Red Hat Enterprise Linux 6.2.The maximum write performance for a single 10GigE attached client for GPFS was 1220 MB/s and for a single 2x10GigE LACP attached client 2400MB/s. In the same configuration writing with two 2x10GigE LACP attached clients achieved up to 4600 MB/s. For details IOR outputs see section

## 3. Excalibur Overview

Imaging experiments at X-ray sources like Diamond Light Source require a 2D position sensitive detector like the Excalibur Detector, a joint development of the Science and Technology Facilities Council and Diamond Light Source. The primary application of the Excalibur detector will be in Coherent Diffraction Imaging (CDI) but will also find applications in X-ray Photon-correlation Spectroscopy (XPCS), full-field microscopy and holo-tomography. Please see Figure 1.

| Parameter | Medipix3 system | Alternative systems (e.g.) |
|---|---|---|
| Max. frame rate | • 1kHz AT 12 bit,<br>• 30kHz at 1bit | Pilatus II < 300Hz, Pilatus XFS > 10kHz |
| Readout time | <1ms per frame, with 12bits per pixel,continuous with no dead time | • In Pilatus II, the system is dead during readout.<br>• Pilatus XFS offers continuous readout |
| Pixel Size | 55x55 micron | • Pilatus II: 172 micron<br>• Pilatus XFS: 75 micron |
| Dynamic range | Single 12 bit, greater by summation of frames. | Pilatus XFS is 12bits |
| Quantum efficiency | ~65% at 15keV | 15% at 15keV |

**Overall parameters of the Excalibur detector.**

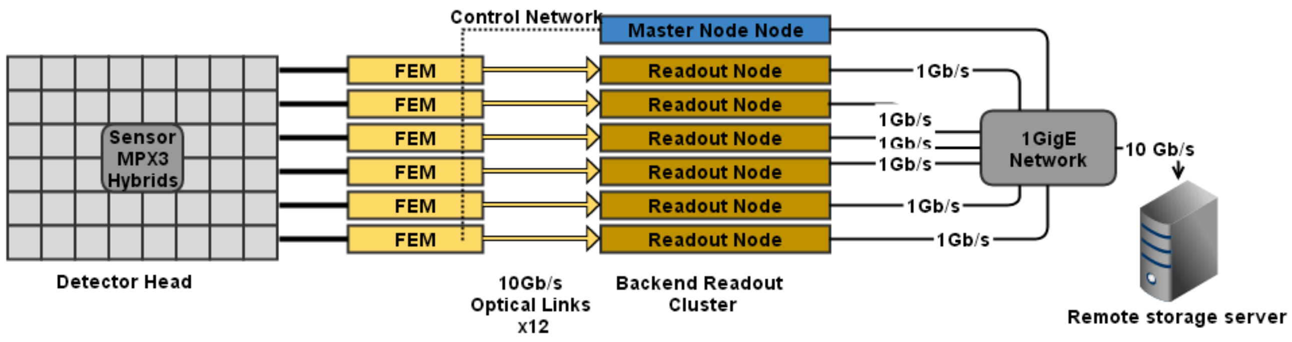


**Figure 1 - A schematic representation of the Excalibur detector.**

The EXCALIBUR detector consists of three modules, each with 16 MEDIPIX3 chips which can be read-out at 100 frames per second in continuous mode or 1000 frames per second in burst mode. In each module, the sensor is a large single silicon die covering 2 rows of 8 individual MEDIPIX3 read-out chips and provides a continuous active detection region within a module. The detection area of the 3-module EXCALIBUR detector is 115 mm x 100 mm with a small inactive region between modules. Each detector module is connected to 2 FPGA read-out boards via a flexi-rigid circuit to allow a fully parallel read-out of the 16 MEDIPIX3 chips. The 6 FPGA read-out boards used in the EXCALIBUR detector are interfaced to 6 computing nodes via 10Gbit/s fibre-optic links to maintain the very high frame-rate capability. Each computing node is connected separately by a 1Gbit/s link to a 10Gbit/s switch; the latter leading directly to the storage server

A major function of the computing nodes is to field the data provided by the read-out boards and write HDF files to the storage facility. The nature of this transfer provides a model to examine the highly parallel characteristics of the file systems used. For the purposes on this document it was useful that two file systems Lustre and GPFS were being evaluated for wider deployment.

## 3.1. Comparison of Lustre and GPFS to support Excalibur

As already stated, Lustre and GPFS are highly complex with many tunable parameters to optimize their performance for different operational requirements. It is not currently possible to provide a detailed analysis the configurations with respect to differing requirements due to the limited practicability of deploying real world testing environments; these tests will depend on the properties of detectors and, moreover, on the changing physical storage and networks employed at the scientific facilities.

Notes:

- Report PaNdata D8.2 provides details of the tuning of the GPFS and Lustre file systems and this report will be restricted to highlighting additional issues emerging during the commissioning of a highly parallel detector.

Notwithstanding the above, we used the ab initio commissioning of the Excalibur detector to perform an analysis of the suitability of similarly resourced Lustre and GPFS file systems. The commissioning rig for the Excalibur detector has been deliberately equipped with equivalent mounts for Lustre and GPFS. These have been used for evaluating the detector and the performance of the file systems under similar conditions to those expected under normal operating workload. The relevant software and version details are noted in the tables of sections 3.2.2 and 3.2.3 below.

### 3.1.1. Write test to the Lustre and GPFS under similar conditions

The file system must be capable of supporting the maximum data rate emerging from the detector consequently the write to file tests is of key importance. The throughput plot of the file writing plug-ins under the same conditions

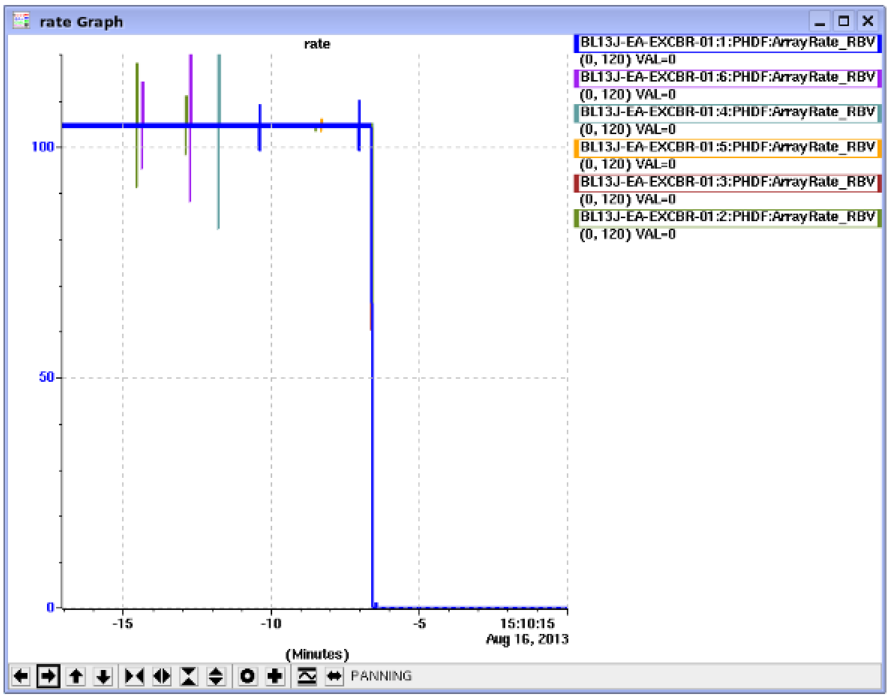
### 3.1.2. 104Hz, 100000 frames to Lustre

**Notes:**

A key architectural concept of Lustre is the availability of Object Storage Targets (OSTs) which are frequently high performance disks, one of the main factors leading to the high performance of Lustre file systems is the ability to stripe data over multiple OSTs. The stripe count can be set on a file system, directory, or file level. The stripe size is a configurable parameter and experience has indicated that that writing to the Lustre03 using a stripe size of 4MB was particularly successful.

The throughput plot of the file writing plug-ins:

| **Module** | **Config** |
|---|---|
| | configure.py --basic --fix --phdf5 --lustre3 |
| fix | No Vertical Gaps = Enabled |
| phdf5 | Lazy Open = No |
| GPFS | Mounted |
| File path | /mnt/lustre03/testdir/tmp/stripe4MB |
| Ave, Data rate | 586 Mb/s ~ Close to the theoretical maximum of the 6 parallel 1Gb links to a 10Gb/s switch and then the file system. |
| Software | Operation system of readout cluster nodes – RED Hat Enterprise Server 6 |
| File System | Server Lustre 1.88 (Patched)/Client1.88 |

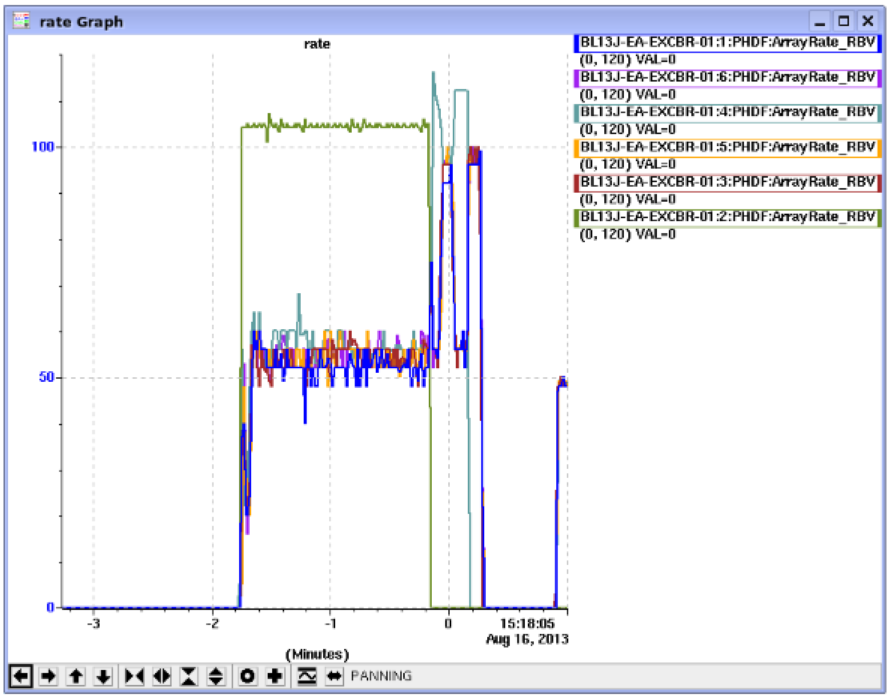


### 3.1.3. 104Hz, 10000 frames to GPFS

**Notes:**

Writing to GPFS appears to be less efficient than Lustre in the case of the Excalibur. One of the nodes managed the 104Hz, the others all seemed to saturate at a little more than 50Hz. The queues of the unfavoured nodes all overran and dropped frames.

| Module | Config |
|---|---|
| | configure.py --basic --fix --phdf5 --gpfs |
| fix | No Vertical Gaps = Enabled |
| phdf5 | Lazy Open = No |
| GPFS | Mounted |
| Ave, Data rate | 170 MB/s ~ best so far achieved |
| Software | Operation system of readout cluster nodes – RED Hat Enterprise Server 6 |
| File System | Server and Client GPFS 3.5.0.11 |



BL13J-EA-EXCBR-01:1:PHDF:ArrayRate_RBV
(0, 120) VAL=0
BL13J-EA-EXCBR-01:6:PHDF:ArrayRate_RBV
(0, 120) VAL=0
BL13J-EA-EXCBR-01:4:PHDF:ArrayRate_RBV
(0, 120) VAL=0
BL13J-EA-EXCBR-01:5:PHDF:ArrayRate_RBV
(0, 120) VAL=0
BL13J-EA-EXCBR-01:3:PHDF:ArrayRate_RBV
(0, 120) VAL=0
BL13J-EA-EXCBR-01:2:PHDF:ArrayRate_RBV
(0, 120) VAL=0

### 3.1.4. Final observations from the Excalibur example

From the outset it is clear that it is necessary to tune the parameters governing the operation of the file systems to optimize their performance for particular use cases. In addition this optimization demands careful choice of the hdf5 library write and read parameters to match the characteristics of the file system used.

Some Notes

Lustre;

- The lustre default block size is 1Mb. Improvements were made by increasing the stripe size to 4 MB using Lustre specific command lfs setstripe and writing 4Mb chunks in hdf5.

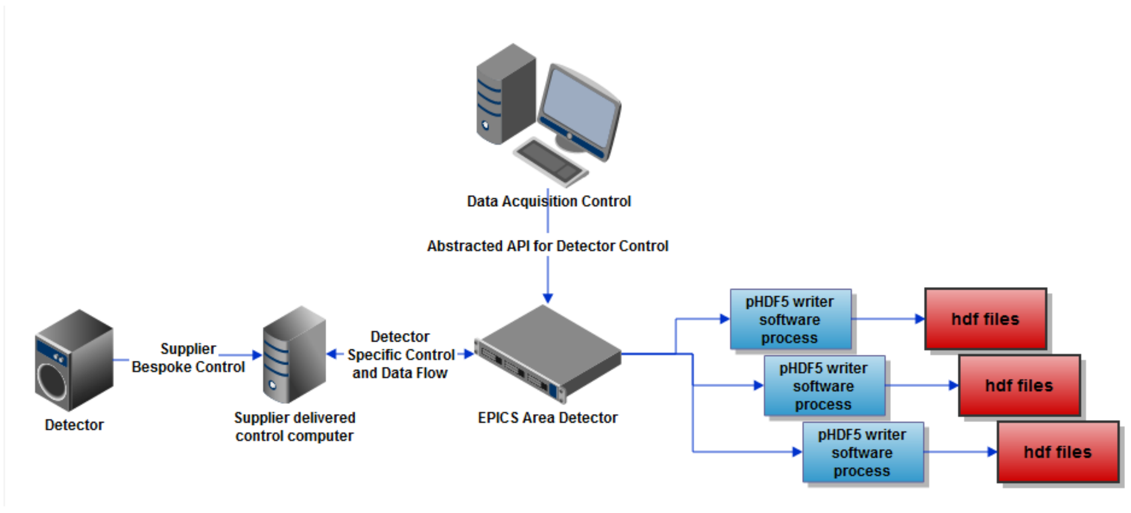
GPFS:

  - A Z Dimension data chunking is likely to be optimal as GPFS is normally setup to use 4 Mb blocks by default.

- When configuring the file writer to write a very large number of frames (>100000) the open time increases and each write operation also takes longer.
    - When writing extensible datasets the number of frames should be irrelevant, however, the HDF file format uses B-Trees for chunk indexing, and when these need to be extended to an extra level because of a large number of chunks, the overhead can be significant. HDF 1.10 (scheduled for release in 2014) introduces changes to the file format so that datasets with one or zero unlimited dimensions are stored without the need for B-Tree indexing.
- GPFS works very well with smaller datasets (<10000 frames) but with larger datasets its performance degrades very rapidly.

  - The underlying reason for this observation is not clear but may be associated with the detailed configuration of GPFS in this case.

# 4. Practical details of Hierarchical Data Format implementation


Data Acquisition Control
Abstracted API for Detector Control
Detector
Supplier Bespoke Control
Supplier delivered control computer
Detector Specific Control and Data Flow
EPICS Area Detector
pHDF5 writer software process
pHDF5 writer software process
pHDF5 writer software process
hdf files
hdf files
hdf files

The above diagram illustrates the latest architecture used to for high performance data acquisition. The EPICS area detector provides a standard API for a Data Acquisition Control program.

### 4.1.1. Notes:

- Parallel writing of HDF5 files is only possible if the writing is done using separate processes and either to separate files, or using MPI to coordinate writing to one file. It is not possible currently for separate threads in a process to write simultaneously as the core libraries available from the HDF5 group (http://www.hdfgroup.org/HDF5/) as in certain circumstances the thread safety of the core libraries may be compromised. There is an initiative to commission this update but the cost of this change is currently out of scope for the PaNdata project.
- Mpi i/o based pHDF5 usually won't go along with compression of the data streams as a consequence of the HDF5 file layout, so compression or any other image manipulation algorithm was bound to be single threaded. This problem has meanwhile been successfully tackled in a collaboration of Dectris, DESY, HDFgroup, NeXuS and PSI (see the Dectris success stories: https://www.dectris.com/successstories.html) and PaNdata ODI D8.3 for details.
- HDF5 files are being used for parallel read access with no specific parallel architecture, simply opening the file read only from several nodes of the cluster. As expected the performance is improved if we the file is distributed across the parallel file system ( e.g., in Lustre, set the number of OST's - Object Storage Target - to store the data on to more than 1). This results in an almost linear improvement to the read speeds.
- In demanding cases the overall data rates and processing speed can depend on optimizing intermediate resources particularly by using memory caching techniques.

#### 4.1.2. An illustration of Tomography an Imaging use of the above infrastructure

- Tomography and Imaging beamlines generally operate simultaneously and can acquire 1-2 TB per day depending on duty cycle and the availability of sample changers
- Typically PCO 4000 data sets tend to be 30 -> 120Gb where the latter would consist of 4k x 2.5k x 6k.
- The 6k could be expressed in 6000 separate files but large gains are made by using the chunking capability of HDF5 to speed up the matrix transposition by defining the chunk layouts appropriately.
- The file system must allow multiple read access and currently leads to < 30 minutes to reconstruct using 56 Tesla or 16 Fermi GPUs

# 5. Conclusions

Simple initial tests indicate that GPFS may significantly outperform Lustre at 10 GbE speeds, but the reality may depend on the exact problem workload.

There are no definitive recommendations for this report. The results gained from the commissioning of the Excalibur detector indicate that Lustre may be a more efficient underlying file system in this case. There are nevertheless arguments based on experience from other facilities that GPFS or derivatives would be preferable.

Current observations:

- Both Lustre and GPFS require significant tuning to optimize performance for a given workload.
- The overall performance of the application software running on a high performance parallel file system depends on careful optimization on the applications' calling of the available support libraries. Non optimized can negate advantages implemented in the underlying file system.
- There may be considerable benefit to be gained by setting up an open web site to which the optimal file system configurations for the detectors and associated computing architectures of different sites can be uploaded.

- Experience so far has shown that good results can be achieved without a thread safe hdf library but the implementation of the latter would be valuable for future projects.
- It was apparent from the Excalibur development that large HDF5 files provide a challenge to the underlying file system. Of particular concern, semi random file access and B-tree constraints can lead extended write delays and unexpected data stream behavior. See Appendix A
- In the case of very demanding data acquisitions using high data rate detectors an appropriate strategy may be to delegate a NeXus file as the analysis/evaluation application entry. This file would consist of internal data sections and metadata and use file links to the large data files being created. These files would be written in the most efficient format possible, a strategy also being employed by Dectris for the EIGER detector

## 5.1. Software available:

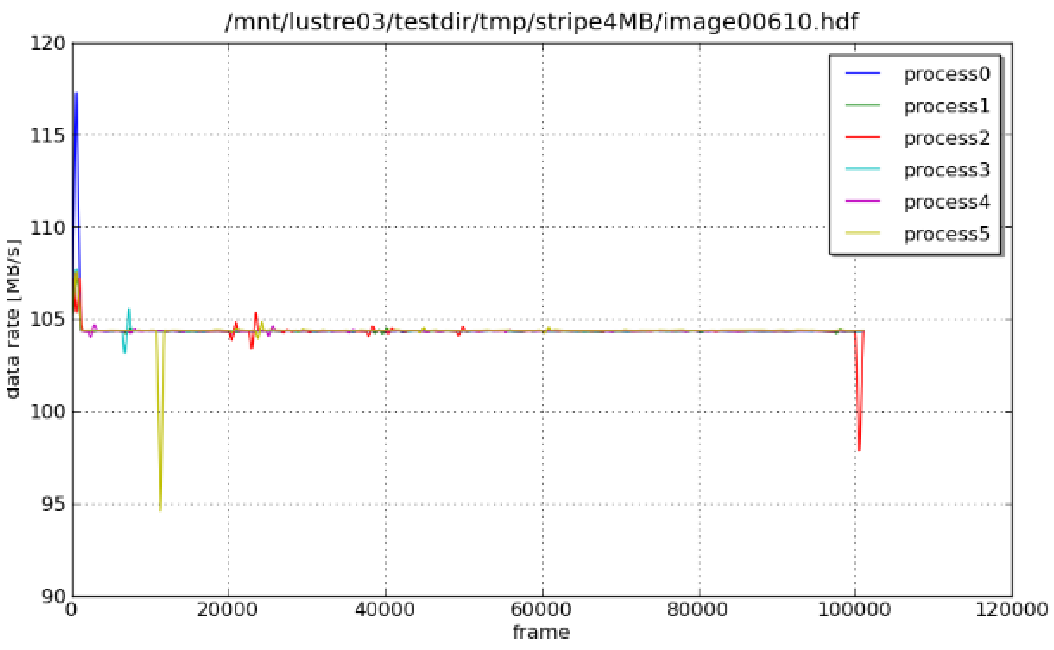
- http://www.h5py.org/ - h5python, a version of python optimized to access HDF5 files and allow the use of additional tools such as numpy.
- Parallel hdf5 http://www.hdfgroup.org/HDF5/PHDF5/
- http://sourceforge.net/p/cbflib/code-0/349/tree/ - cbflib -> NeXus
- http://www.opengda.org/ - contains a NeXuS data writer
- https://code.google.com/p/pni-libraries/ - a high performance library for directly reading and writing NeXuS files.
- http://cars9.uchicago.edu/software/epics/areaDetector.html - EPICS area detector - Software to provide a standard interface to area detectors from the EPICS controls system.
    - The detailed control of the detector is delegated to plugins within the EPICS area detector architecture; the plugins are normally written in C or C++.
        - The parallel hdf5writer is currently tailored to EPICS/Diamond requirements, however this is only superficial. The intention would be to abstract it out and publish it on our external website.

          The main issue is to abstract the TCP protocol from detector system to phdf5writer.
    - Given the plugin code it should be relatively straightforward to integrate into the LIMA architecture (Lima.blissgarden.org/applications/tango/doc/index.html)

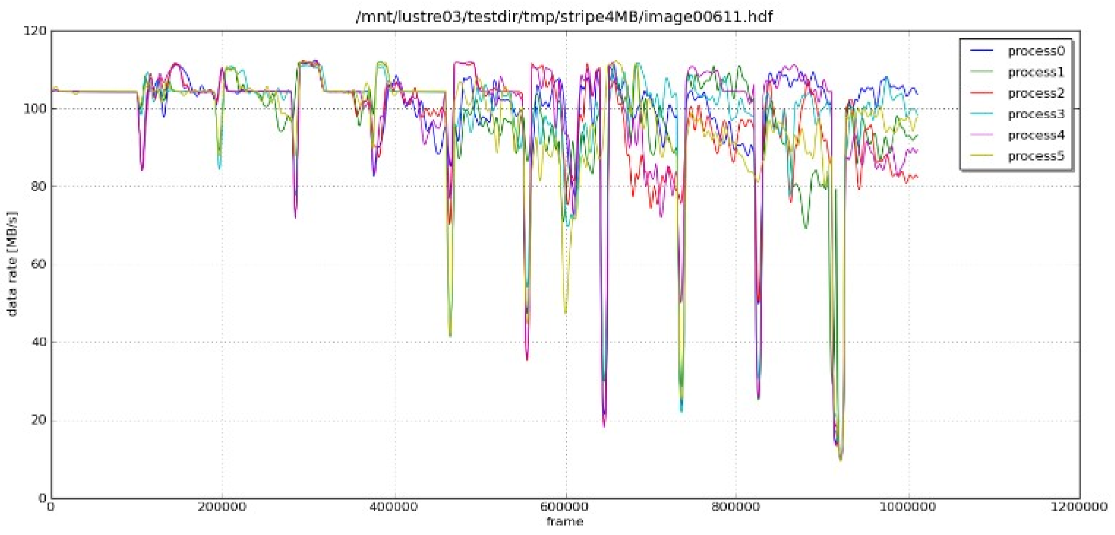
# Appendix A – Effects of file indexing

The below graphs show the effect indexing binary search trees during file writing

A graph of 100000 frames at 104Hz (6 nodes, each writing 1MB size frames)



Illustrating the effects of the chunk indexing binary search tree, growing a branch; see the following 1M frames of 1MB per process at 104Hz


/mnt/lustre03/testdir/tmp/stripe4MB/image00611.hdf
data rate [MB/s]
frame
process0
process1
process2
process3
process4
process5