HIGH SPEED DETECTORS: PROBLEMS AND SOLUTIONS

N.P. Rees, M. Basham, F.J.K. Ferner, U.K. Pedersen,
T.S. Richter, J.A. Thompson
Diamond Light Source Ltd, Didcot, Oxfordshire, UK.

Abstract

Diamond has an increasing number of high speed detectors primarily used on Macromolecular Crystallography, Small Angle X-Ray Scattering and Tomography beamlines. Recently, the performance requirements have exceeded the performance available from a single threaded writing process on our Lustre parallel file system, so we have had to investigate other file systems and ways of parallelising the data flow to mitigate this. We report on the some comparative tests between Lustre and GPFS, and some work we have been leading to enhance the HDF5 library to add features that simplify the parallel writing problem.

INTRODUCTION

When Diamond Light Source, a third generation synchrotron light source, entered service in 2007 no beamlines had any detectors that even approached saturating a 1 Gigabit/sec Ethernet (GbE) link. Our first detector to exceed this data rate came on line in mid 2011, and in the 2 years since this time 8 beamlines have acquired detectors requiring 10 GbE links, and we are now working on detectors that exceed 10 GbE speeds by almost a factor of ten. This increase in peak detector data rates by nearly two orders of magnitude in a few years has challenged our software and hardware architectures. This paper presents a snapshot of the current status and developments we are undertaking which will help manage these high data rates.

HARDWARE SETUP

File System Configurations

In order to support the high performance detectors at Diamond, two different high performance file systems have been deployed. Lustre[1] has been used sucessfully for some time now and GPFS[2] has recently been put into production. Both use Data Direct Networks (DDN) disk backends, a DDN SFA10K for Lustre and a SFA12K-40 for GPFS. Both backends provide sufficient bandwidth to not cause any bottleneck during these tests and this has been verified using sgpdd_survey. Each file system has 4 servers connected to the disk arrays via InfiniBand (IB) and to the network with multiple 10GbE links. The Lustre Object Storage Servers (OSSs) are Dell PowerEdge R610s with 2x10GbE Link Aggregation Control Protocol (LACP) bonded links and the GPFS Network Shared Disk (NSD) servers are Dell PowerEdge R720s with 4x10GbE LACP bonded links.

Clients

Most of the benchmarks discussed have been run on six clients in parallel. To simplify the network configuration, two different sets of clients have been used for the 1GbE and 10GbE tests. The test clients are either Dell R610s, R620s or R720s and all are using Dell Broadcom network interface cards (NICs).

Network Interconnect

The network at Diamond is built around two separate core switches (one Force10 C300 and one Extreme Networks X8) with clients and servers usually connected through smaller edge switches (see Fig. 1).

Each edge switch is connected to both core network switches using one or more 10GbE links bonded using LACP. The edge switches act as a router for all clients connected to them. Equal Cost Multi-Path (ECMP) and Open Shortest Path First (OSPF) protocols are used to make efficient use of all available links.

The GPFS NSD servers are connected to a stack of two Extreme Networks X650 switches using 4x10GbE LACP links per server. The X650 stack in turn has 8x10GbE LACP bonded uplinks into each of the two core switches providing a total of 16x10GbE to the network.

The Lustre OSSs are an exception to the standard configuration. The two pairs of OSSs are each connected to one of the two core network switches with 2x10GbE LACP links in this case, the cores switches act as a router for these clients.

The 10GbE clients are connected via 1x10GbE links to one Extreme Networks X670 switch with 3x10GbE LACP uplinks to each of the core network switches.

The 1GbE Excalibur detector[3] nodes are connected to a Avaya 56XX switch which has 1x10GbE uplink to each core network switch.

PERFORMANCE

The parameter space for a given detector system writing to a parallel file system is very complex and calls for an understanding of the operation and configuration of file system, network and compute nodes. The detector readout design, number of pixels per chip and per module, as well as the experimental set-up and post processing application (disk reader), also effect the choice of I/O pattern.

Measuring the basic I/O performance to a parallel file system is a relatively simple task for system administrators. However, the I/O pattern defined by using a particular detector system often yields quite different results from a

ISBN 978-3-95450-139-7

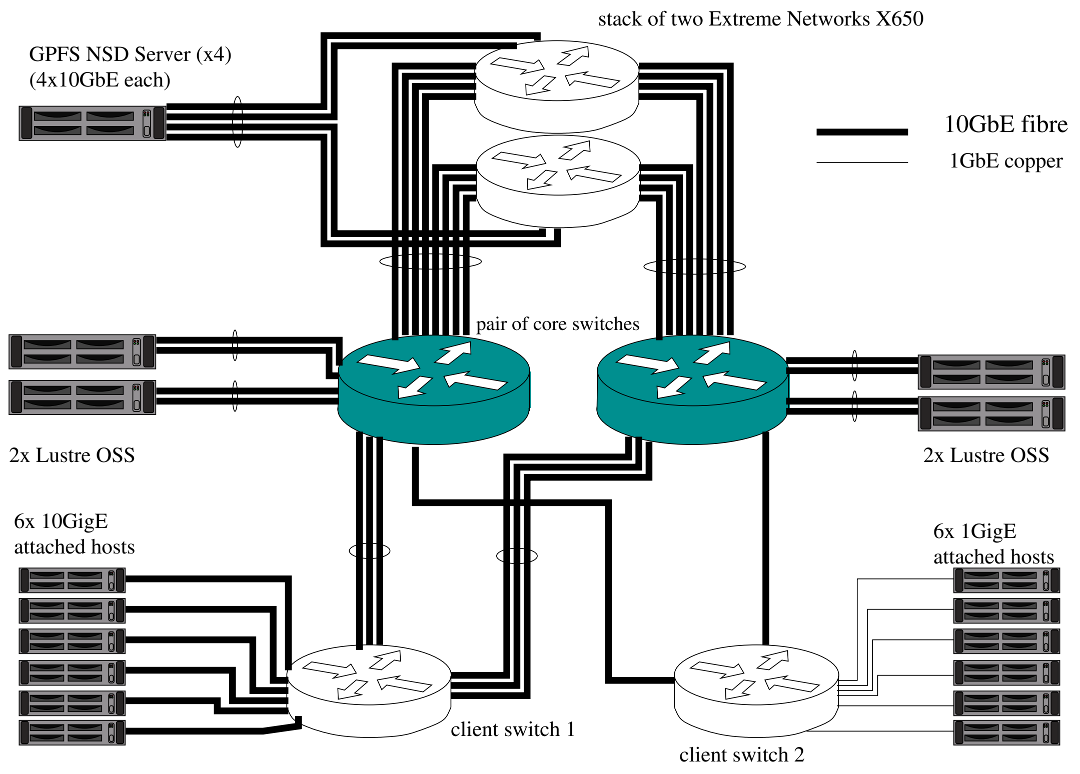

Figure 1: Network Diagram.

simple binary dump to disk. The challenge is to choose a sensible I/O pattern for a particular detector and to tune the available parameters of the system for optimal performance.

It is not feasible to simply scan through all available parameters in a brute-force approach. Initial starting points have been chosen based on knowledge of the file systems and detectors as well as best practices for using the HDF5 library[4] and the available file systems.

Benchmarking Tools

The measurements in this report have been set up based on the Excalibur detector use-case scenario. This set-up comprises a 6 node cluster where each node is connected to a detector element and equipped with individual 1 GbE links. For development and testing we also use a second 6 node cluster with 10 GbE links. The clusters are connected to central parallel storage systems based on Lustre and GPFS.

Benchmarking applications “ior”[5] and a Diamond development “phdf5write” are used to measure performance and tune parameters. The “ior” disk I/O benchmarking application with the HDF5 back-end is appropriate for system administrators to benchmark and test the settings of cluster nodes and the file system servers. The “phdf5write” application, in contrast, writes HDF5 datasets in exactly the same way as detectors write datasets; appending 2D images to an extending 3D dataset. It also provides timestamps for each individual write operation and utilizes the HDF5 chunking feature and the parallel variant of the HDF5 library. To measure and diagnose the write performance of the cluster, file system and configuration of a particular detector data acquisition system we use a combination of both these tools.

HDF5 Parameters

The HDF5 library provides access to set a number of low-level parameters, tuning the performance for different I/O patterns and file systems:

- The boundary alignment parameter. This is file system related and is set to 1 MB for Lustre and 4 MB for GPFS, to match the respective file system block sizes.

- The Write block (or chunk) size for a single individual write. This is 1 MB per node in the Excalibur detector case, however the HDF5 chunk sizes are set up to do 4 MB writes. This is obviously more efficient in the GPFS case but for the Excalibur detector it has proven more efficient on Lustre in combination with a 4 MB Stripe Size.

Selecting an appropriate chunk pattern is a balance between write and read performance. Optimal performance depends heavily of the pixel dimensions of the detector imaging chip. For efficient I/O, each block (or chunk) which is written to disk must fit neatly inside a multiple of the file systems transfer block size. For example, the Lustre “Stripe Size” is a multiple of 1MB. For real detector

ISBN 978-3-95450-139-7

systems this is often difficult as the horizontal pixel resolution is rarely a power of 2. On the Excalibur detector, the chip size is 256x256 and each readout node receives a 1MB block of data per frame which is ideal. However, there is a difficulty in that there are requirements for having blank (auto filled) pixels in the locations of the frames where there are physical gaps between sensors and modules. These gaps essentially overflow the clean 1MB chunk boundary, making write operations inefficient (see Table 1).

Table 1: Excalibur Detector Parallel Write Performance For Datasets With And Without Gaps

Gaps	Frame Size [MB]	Write rate [MB/s]
None	1.00	104
Chip gaps	1.02	49
Module gaps	1.18	46

The HDF5 library also utilises a binary search tree for indexing the individual chunks on disk. The size of the branches can be tuned. However, when writing huge files, and using a large search tree, write operation halts as all nodes process (full load of CPU) to grow another branch on the tree. The next major release of HDF5 (V1.10) introduces a file format change which allows dataset that extend only in one dimension to use an "Extensible Array" for indexing, which will overcome this deficiency if the dataset has only one unlimited dimension.

Measurements

The measurements using ior demonstrate the basic I/O performance of the two Diamond file systems. Table 2 lists the results of writing from 12 processes on 6 nodes (2 processes per node) to 12 individual files and to a single shared file, for Lustre and GPFS, using the "ior" application.

Table 2: Data Rate Measurements Using "ior"

File System	Multiple Files [MB/s] (stddev)	Single File [MB/s] (stddev)
GPFS	3148 (123)	3044 (115)
Lustre	3518 (154)	1328 (96)

The measurements with the "phdf5write" application are more realistic in terms of choice of I/O pattern in relation to real detector acquisition use-cases. Data consist of 2D blocks which are appended to a chunked 3D dataset. Table 3 lists the results, from the same cluster. There is a scatter in repeated measurements and sometimes GPFS is faster than Lustre, but this is an average of a number of measurements. Note that "phdf5write" currently only writes to a single shared file.

The "phdf5write" application provides a more detailed view of the performance of the individual write operations, and further investigation reveals that the performance is not balanced across all nodes. Each node writes the same amount of data and all nodes write to a single shared file. Figure 2 shows the I/O performance for each individual write on the two file systems. The total amount of data written in the test is about 580 GB across all nodes. This figure highlights the fact that some nodes write to disk significantly faster than other nodes in the cluster. This behaviour does not appear to be related to the node configuration because the relative performance of the nodes varies randomly from run to run - however, for Lustre, in particular, the performance of any particular node appears to be reasonably consistent in one given run, and so the aggregate instantaneous throughput drops off as the nodes complete writing.

Table 3: Data Rate Measurements Using "phdf5write". Data rate is (total data)/(total time), so is dominated by the time taken by the slowest node.

File System	Single File [MB/s]
GPFS	943
Lustre	1160

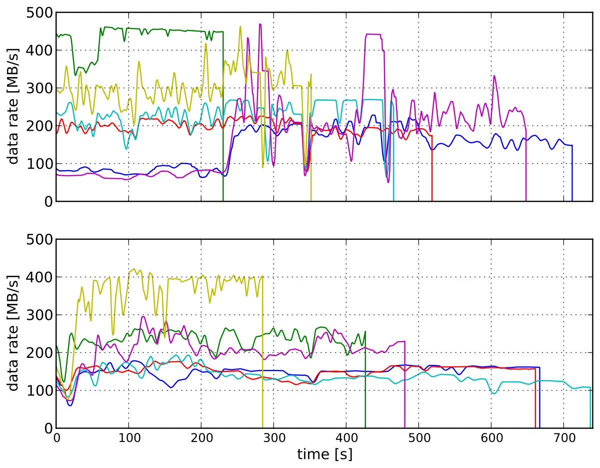

Figure 2: Six Node Parallel Write To GPFS (top) And Lustre (bottom). All nodes write the same amount of data, but performance varies between nodes.

HDF5 DEVELOPMENTS

As seen above, creating data files from high performance detectors is often a balance between storing data in one file or many files. If the data is in one file, the number of file system operations is minimised, but contention is created since many processes must access the file and a single piece of data corruption could make a large dataset unuseable. At Diamond we have standardised on the NeXus format for data storage, using HDF5 as the underlying file format. In order that we can develop systems that can handle the large data rates, we have been working with The HDF Group to define and implement the following extensions to the HDF5 programming interface.

Single Writer Multiple Reader

Single Writer Multiple Reader (abbreviated to *SWMR* and pronounced *swimmer*) allows data analysis programs to read an HDF5 file while it is still being written. As detectors get faster the traditional approach of writing one data file per frame leads to thousands or millions of files - often in a single directory. To mitigate this, data is written in the form of data cubes, with the third axis being a frame number or time axis. However, in the current version of HDF5 files cannot be read while still being written and so data analysis and display must be postponed until a whole series of frames have been acquired. This is unsatisfactory and is being addressed by the SWMR development. A set of requirements and use cases describing this functionality was produced and a design study (funded by Diamond Light Source) has been concluded to validate the design assumptions (see ftp://ftp.hdfgroup.uiuc.edu/pub/outgoing/SWMR/).

At this point Diamond Light Source and Dectris Ltd are funding The HDF Group to provide production code for this product. There is still a funding shortfall which means that the code will not be able to be released for general support within the HDF5 codebase, and we are looking for further funding partners to make this happen.

Virtual Datasets

As the performance tests have shown, whilst HDF5 has an ability to write multiple data streams in parallel to a single file, the performance is usually lower than writing to multiple independent files because of file contention. In addition, in this mode the data cannot be compressed and in some scenarios this can also reduce the effective throughput dramatically.

Virtual datasets are an extension to HDF5 which allows a single virtual dataset to be composed of data from datasets in multiple underlying files. This circumvents both the compression and file contention issues, whilst hiding the multiple underlying files from any program trying to access the data after it is written. A set of requirements and use cases describing this functionality has also been produced, and we intend to fund this work after the completion of the SWMR work.

CONCLUSIONS

In working with high speed detectors we have found that the system performance is significantly lower than the sum of individual component performance. Some general conclusions are:

- Standard HPC benchmarking tools such as ior often do not represent the performace seen in a real application.
- Lustre may be faster at managing the pHDF5 contention between nodes than GPFS is.
- GPFS is faster at streaming data from one node, if there is no contention.
- Writing separate files is faster than using pHDF5 to enable all nodes to write to one.

Having said this, we feel these should be balanced by caveats that we are still actively learning and we have more experience with Lustre than GPFS. However, in order to address these issues we are actively pursuing modifications to the HDF5 library to enable process to write separate data files that can be viewed as parts of a single dataset.

REFERENCES

[1] `http://www.lustre.org`.

[2] `http://www.ibm.com/systems/software/gpfs/`.

[3] J.A. Thompson, I. Horswell, J. Marshal, U.K. Pedersen, S. Burge, J.D. Lipp, and T.Nicholls. "Controlling the EXCALIBUR Detector". In *Proceedings of ICALEPCS2011*, 2011.

[4] H Howison, Q. Koziol, D. Knaak, J. Mainzer, and J. Shalf. "Tuning HDF5 for Lustre File Systems". Technical report, The HDF Group, 2010.

[5] W. Loewe, T. McLarty, C Morrone, and R Klundt. "IOR - Parallel filesystem I/O benchmark". `https://github.com/chaos/ior`.

ISBN 978-3-95450-139-7