Handling high data rate detectors at Diamond Light Source

U.K. Pedersen, N. Rees, M. Basham and F. J. K. Ferner
Diamond Light Source, Harwell Science and Innovation Campus, OX11 0DE, United Kingdom
E-mail: ulrik.pedersen@diamond.ac.uk nick.rees@diamond.ac.uk mark.basham@diamond.ac.uk frederik.ferner@diamond.ac.uk

Abstract. An increasing number of area detectors, in use at Diamond Light Source, produce high rates of data. In order to capture, store and process this data High Performance Computing (HPC) systems have been implemented. This paper will present the architecture and usage for handling high rate data: detector data capture, large volume storage and parallel processing.

The EPICS areaDetector framework has been adopted to abstract the detectors for common tasks including live processing, file format and storage. The chosen data format is HDF5 which provides multidimensional data storage and NeXuS compatibility.

The storage system and related computing infrastructure include: a centralised Lustre based parallel file system, a dedicated network and a HPC cluster. A well defined roadmap is in place for the evolution of this to meet demand as the requirements and technology advances.

For processing the science data the HPC cluster allow efficient parallel computing, on a mixture of x86 and GPU processing units. The nature of the Lustre storage system in combination with the parallel HDF5 library allow efficient disk I/O during computation jobs. Software developments, which include utilising optimised parallel file reading for a variety of post processing techniques, are being developed in collaboration as part of the Pan-Data EU Project (www.pan-data.eu). These are particularly applicable to tomographic reconstruction and processing of non crystalline diffraction data.

1. Introduction

Diamond Light Source data processing is based on a centralized set of computer resources engineered on HPC principles, and this has provided a solid reliable platform for high performance data reduction and analysis. Recently, driven by higher data rates (particularly in Macromolecular Crystallography and Tomography) all aspects of the software and hardware design have been reviewed to ensure that the infrastructure is suited to the new demands. This paper presents very much a point in time view of the current status.

2. Computing Architecture and Performance

The network used for data acquisition and analysis is Ethernet based with two independent core networks provide a resilient high performance backbone (see figure 1). Every beamline and all compute clusters have a dedicated network connected to both core networks with up to 2x10GigE per core network. Traffic is routed using standard routing protocols such as OSPF and ECMP.

Diamond has two production high performance storage systems, both using Lustre and Data Direct Network (DDN) disk backends. Data on the first is stored in a 10 tray S2A9900 with 1
TB SATA drives, and on the second in a 5 tray SFA10K with 2 TB SATA drives. The Lustre Metadata servers use separate storage with 15k SAS drives. All servers are connected via 10GigE to one of the core networks, with the failover server pairs connected to separate core networks.

The HPC cluster consists of 130 x86 based compute nodes with 8-12 CPU cores per node (1240 CPU cores in total) and 40 GPU nodes with 16 Nvidia S1070s and 24 Nvidia M2090s. Compute nodes with GPU units have 2 GPU units per node (i.e. 1 GPU unit per CPU socket). Each compute node is connected via 1 GigE to a cluster switch with the cluster switches having either 20 GigE or 40 GigE connections to the core, depending on number of nodes served.

IOR[1] has been used to simulate the storage load on the S2A9900 system. The focus during these tests was on write performance from a small number of clients to match the expected use case of individual detectors. Using IOR from 10 GigE clients with one process per node, there was a drop in throughput when going from 1 to 2 nodes (see figure 2) that has been seen elsewhere[2]. This was not seen with 1 GigE clients (see figure 3). Using 8 processes per 10 GigE node there was no dip for 2 nodes but adding a third node did not improve performance at all. Tests using a large number of 1 GigE clients show that the file system is capable of much higher throughput than is currently seen when using a small number of 10 GigE clients.

The same tests have been run at the Swiss Light Source using GPFS, also with an S2A9900 backend. These tests showed that GPFS was able to saturate a 10GigE link with a single process without any additional tuning. Using Infiniband for the interconnect and a 1 MB block size, one node achieved 2GB/s and four nodes 5.5 GB/sec total, close to the limit of the disk system.
3. Detector Software and Parallel Writing

High speed detectors require a scalable solution to handle data. New detectors have multiple sets of readout electronics, each reading out a section of the detector in parallel[3]. Diamond’s aim is to provide scalability by continuing this parallel design using independent data paths as far as possible.

Each readout node is based on commodity server hardware and handles the data from one or more sets of readout electronics. To aid analysis the data from all servers is written into one file on disk, eliminating the need for a separate "stitching" operation. This requires a file format that allows multiple nodes and processes to write directly into separate sections of the same file at the same time. An MPI/pHDF5 based service application has been developed to do this using a cluster of detector storage servers writing to a Lustre file system. NeXus compatible formatting of the hierarchical data layout is also a requirement for this application.

All data control and data flow is based on EPICS areaDetector [4] plugins. There is a generic plug-in for writing HDF5 files using non-parallel HDF5. For this project areaDetector plug-ins have been developed to efficiently transfer data directly from the multiple parallel areaDetector EPICS IOCs to the MPI/pHDF5 service using a pre-release version of the EPICS CAV4 protocol.

To analyze a dataset subsections of the 3D dataset must be read out. Often these are frames, but in tomography the sinograms are perpendicular to the frames. The HDF5 file format supports this type of slicing datasets, but it is necessary to ensure that the dataset is chunked appropriately. The choice of chunking scheme is often a balance between write performance and read performance and is highly dependent on the individual experiment and processing methods[5]. The Diamond HDF5 file writing software allows the user to specify chunking in all 3 dimensions to suit the needs for the particular experiment or post processing.
4. Parallel Read and Post Processing
Most processing requirements have been driven by tomography collection and processing. There are several difficulties with the processing of tomography data, first of which is the generation of sinograms from the collected projections and second is performing the computationally expensive filtered back projection, or iterative reconstruction routine[6],[7].

After the sinograms have been read the reconstruction process can be split up into separate processing jobs per sinogram. Although the HDF library always caches data, it is still important to completely read all the chunks of a sinogram in one operation (see table 1). As the IO requirements of the process is generally large (around 100GB of both read and write spread over all the systems) performance is further improved by reading data ahead in parallel with other processing. This is possible because the reconstruction algorithm predominantly runs on GPU hardware, thus freeing up the host CPU to deal with IO. The main processing task has four threads: one for reading the data out of the HDF file, two for feeding each of the 2 GPU’s and one for writing the files back to disk.

| Item being read                                      | Read Time (s) | Total Time (s) |
|-------------------------------------------------------|---------------|---------------|
| 1 x [4000,1,2000] sinogram (first in chunk)          | 18            | 15            |
| 15 x [4000,1,2000] sinograms (rest of chunk)         | 2             | 48            |
| 1 x [4000,16,2000] whole sinogram chunk              | 16            | 16            |

Table 1. Comparison of sinogram read speeds

5. Conclusions
The biggest problem in this development has been the performance problems found with Lustre and 10 GigE clients. Unfortunately, the commitment to Lustre was made before the LLNL results[2] were known. The recent GPFS performance tests at SLS have confirmed that GPFS does not seem to have these throughput restrictions (at the expense of poorer metadata performance in some areas), so GPFS is being considered for Diamond’s next major storage upgrade, but this paper is very much a snapshot in time and work to optimise the Lustre systems is still continuing.

Acknowledgments
The authors would like to thank Heiner Billich and the Swiss Light Source at Paul Scherrer Institute for his support in running the Diamond benchmarks on their GPFS storage system.

References
[1] Loewe W, McLarty T, Morrone C and Klundt R IOR - Parallel filesystem I/O benchmark URL https://github.com/chaos/ior
[2] R Hedges K Fitzgerald M G and Stearman D 2010 Comparison of leading parallel NAS file systems on commodity hardware Tech. rep. Lawrence Livermore National Laboratory URL https://e-reports-ext.llnl.gov/pdf/457620.pdf
[3] Thompson J A, Horswell I, Marshal J, Pedersen U K, Burge S, Lipp J D and Nicholls T 2011 Proceedings of ICALEPCS2011 URL http://accelconf.web.cern.ch/accelconf/icalepcs2011/papers/wepmn011.pdf
[4] Rivers M areaDetector: EPICS software for area detectors URL http://cars9.uchicago.edu/software/epics/areaDetector.html
[5] Howison H, Koziol Q, Knaak D, Mainzer J and Shalf J 2010 Tuning HDF5 for Lustre File Systems Tech. rep. The HDF Group URL http://www.hdfgroup.org/pubs/papers/
[6] Kak A and Slaney M 1988 Principles of Computerized Tomographic Imaging (IEEE Press)
[7] Gordon R, Bender R and Herman G 1970 Journal of Theoretical Biology 29 471