High Performance Computing at Diamond

Nick Rees

and Greg Matthews, Frederik Ferner, Tina Friedrich, Ulrik Pedersen, Matt Pearson, Jon Thompson, Tobias Richter, Mark Basham, Graeme Winter etc....

Summary

- Scientific Computing Overview
- High Speed Storage
- Real-time Processing
- HDF5 developments

diamond

COMPUTING OVERVIEW

diamond

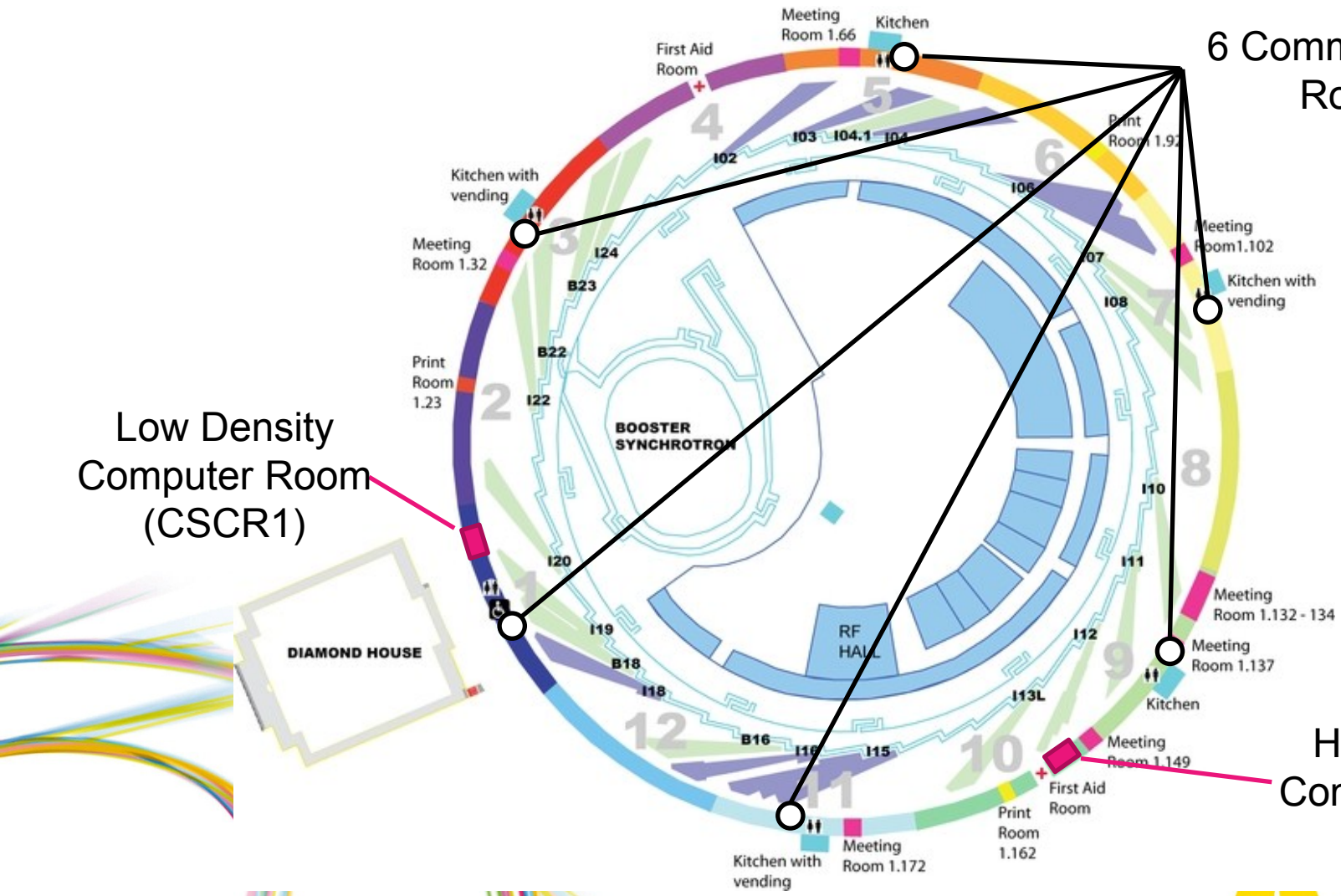
History

- Diamond originally had no provision for central science computing.
- Started to develop it in 2007-2008, with recruitment of some system administrators, building a computer room, and network, storage and compute cluster capital projects.
- Fully endorsed in 2011 with an international review that praised us for building up a world class system in a short time.

System Administration Team

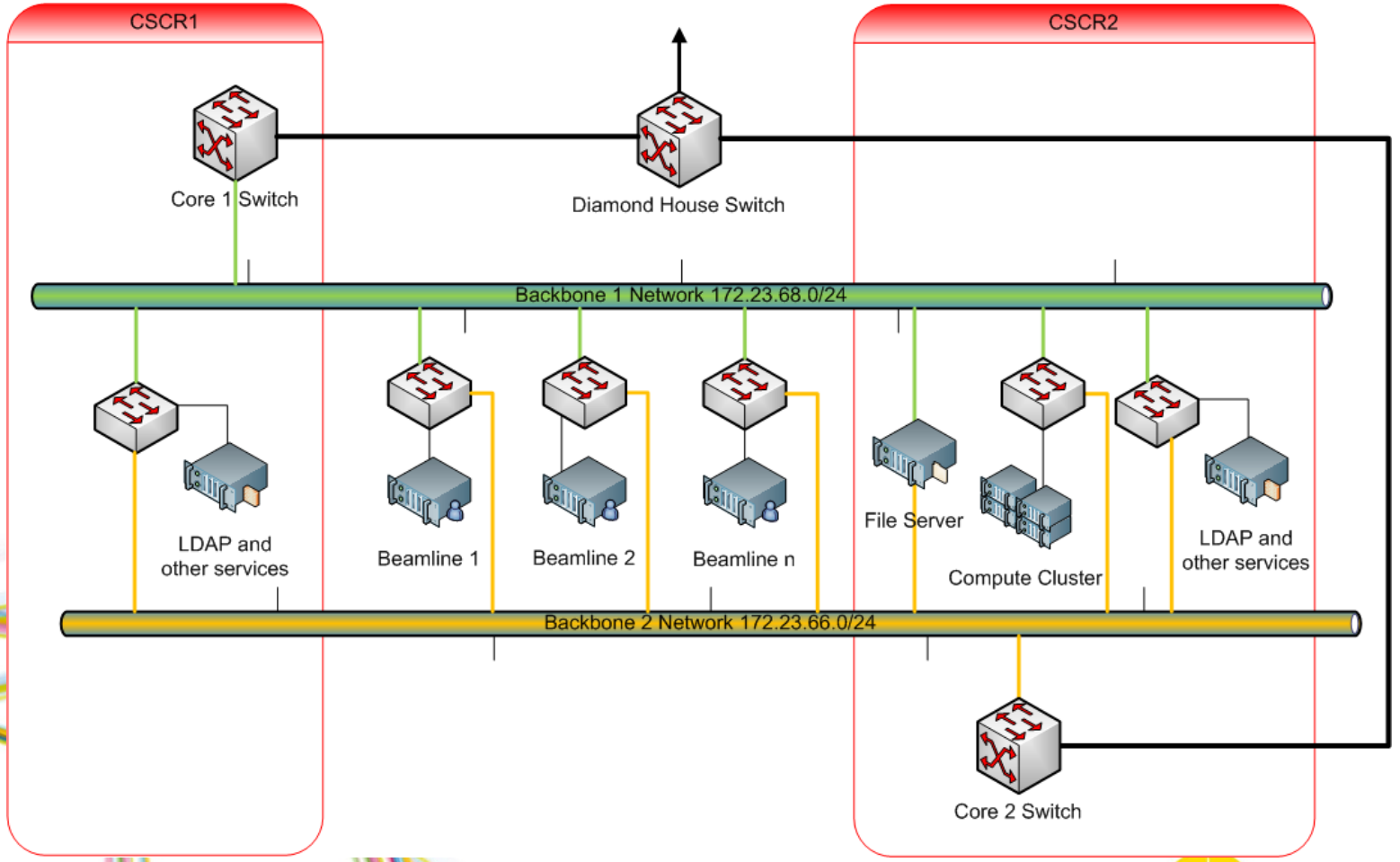
- 6 people, comprising:
  - Team leader
  - Two storage administrators (one senior)
  - Two systems and cluster administrators.
  - One assistant administrator.
- Network is managed through Business IT
  - One dedicated network administrator for the Science network.
- Works closely with other groups to provide for the beamlines' current and future needs.
  - Particularly the Scientific Software team.

Infrastructure



diamond

Science Network Layout



diamond

Computer Clusters

- Intel/AMD clusters:
  - 132 Intel based nodes, 1280 Intel cores in service.
- GPU Clusters:
  - 80 NVIDIA GPGPU's, 23328 GPU cores in service.
- Split across 6 clusters, with a range of capabilities.
- Mostly used by MX and tomography beamlines.
- All accessed via Sun Grid Engine interface.

diamond

Storage

- Three main types of disk storage:
  - High speed storage optimised for parallel access by compute clusters
    - Used by high data rate beamlines
  - Standard RHEL storage
    - Used for beamlines with lower data rates, home directories, software development etc.
  - NetApp NAS
    - For virtual systems and where we have replication and snapshot requirements.
- Client systems have a uniform view of all storage, with access controlled by permissions.
- Windows access provided by Samba.

diamond

Services

- Support the traditional set of computer services
  - DNS, DHCP, LDAP, Active Directory, Web Servers, DB servers, provisioning repositories etc.
- Have a number of virtual systems, now based on a VMware and NetApp NAS infrastructure.

diamond

HIGH SPEED STORAGE

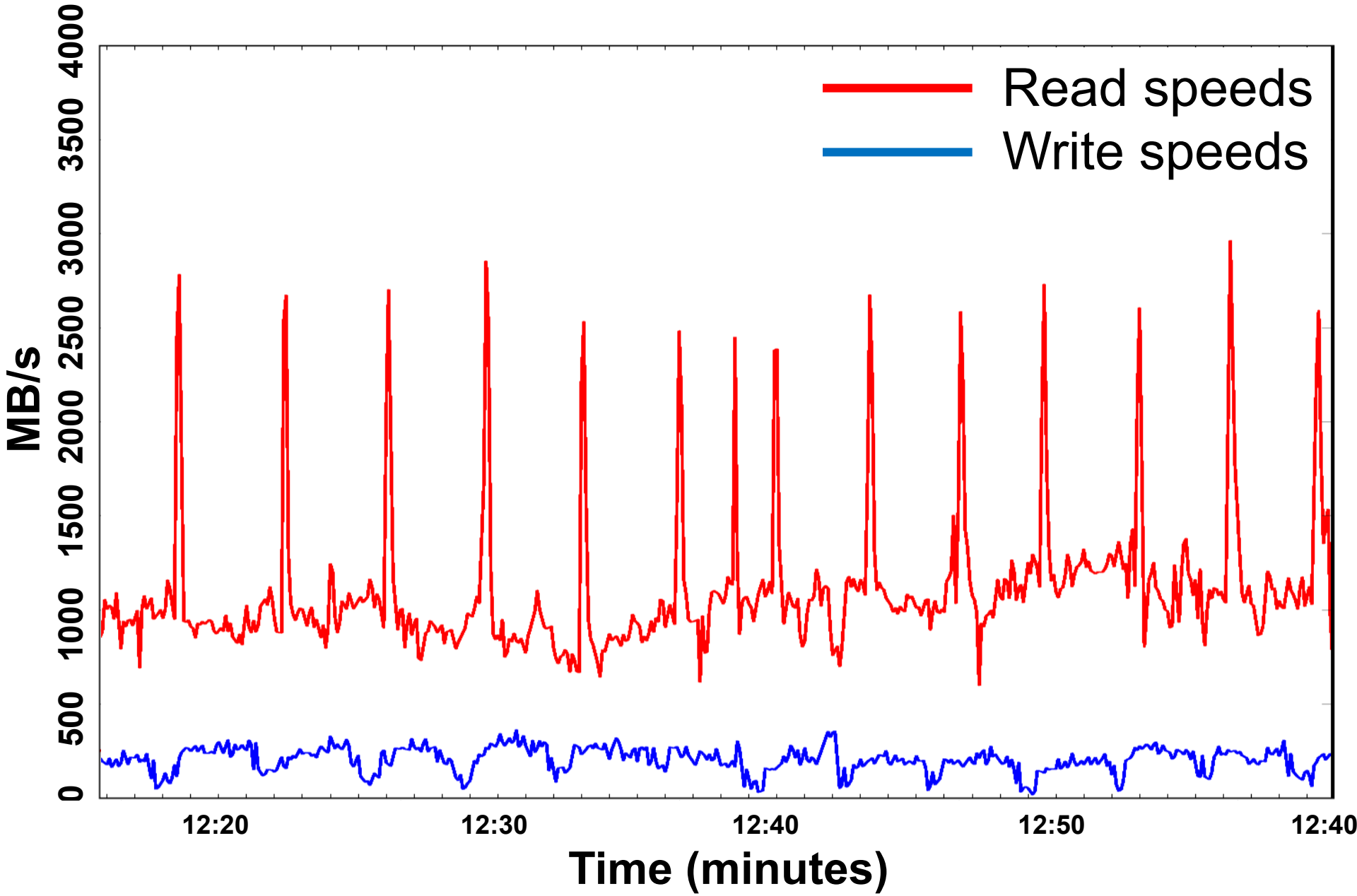
diamond

DLS Storage History

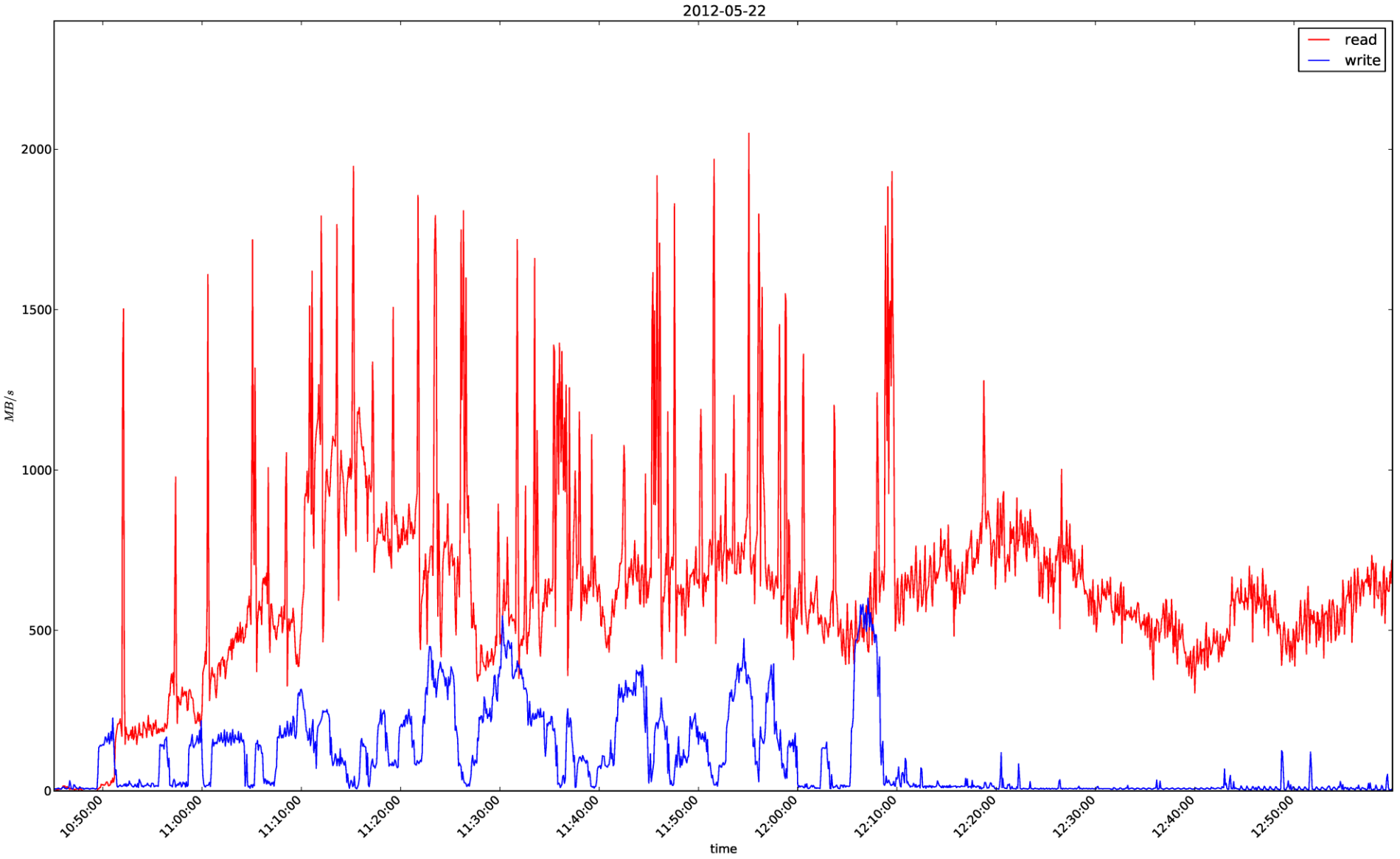
- 2006: First storage, installed separately all on beamlines
  - slow (30 MB/sec) and difficult to manage
- 2008: Bought central Lustre/DDN system
  - 3 GB/sec
  - worked OK for MX and cluster processing
  - had problems with metadata and small files
- 2011: Second Lustre/DDN system installed in April
  - 6 GB/sec
  - Faster metadata
  - Used mainly for MX:
    - 3 x 25 Hz Pilatus 6M (150 MB/sec each)
    - 1x30 Hz Pilatus 2M
    - 1 ADSC system
  - 2008 system is still used for for tomography
    - 4 Hz PCO4000 (90 MB/sec)

diamond

Data Rates while Data Taking

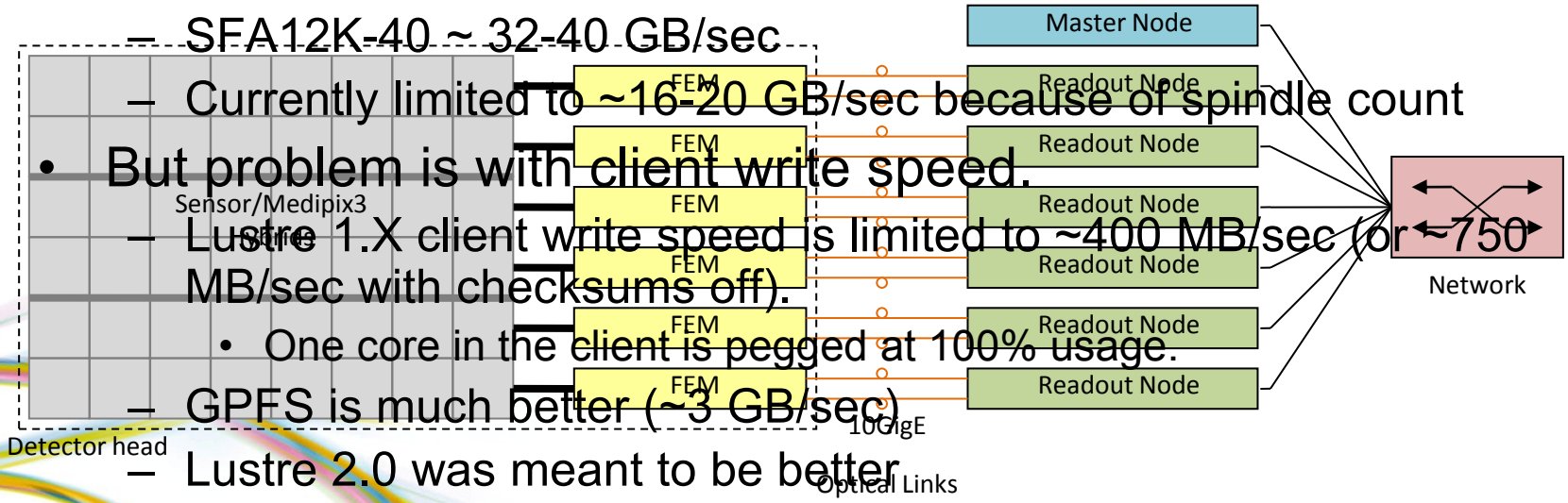


A less regular example

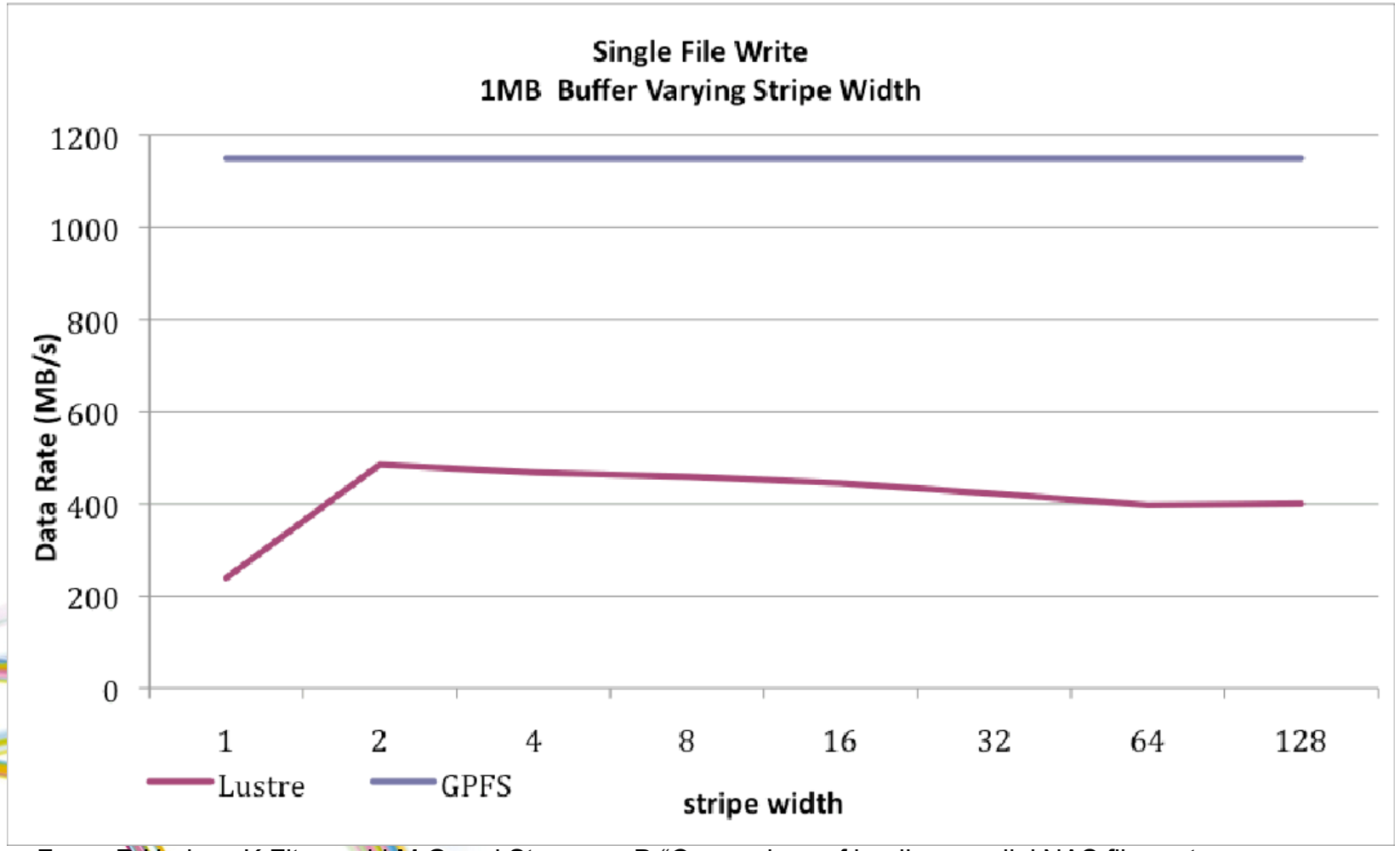


Next challenge

- Faster detectors
  - 100 Hz Pilatus 6M (600 MB/sec write).
  - Tomography detector with 2 PCO.edge systems writing simultaneously (2x900 MB/sec).
  - Excalibur – 6 parallel Medipix3 detector controllers.
- Recently bought next generation DDN system
  - SFA12K-40 ~ 32-40 GB/sec
  - Currently limited to ~16-20 GB/sec because of spindle count
- But problem is with client write speed.
  - Lustre 1.X client write speed is limited to ~400 MB/sec (or ~750 MB/sec with checksums off).
    - One core in the client is pegged at 100% usage.
  - GPFS is much better (~3 GB/sec)
  - Lustre 2.0 was meant to be better
    - However initial tests weren't convincing...



Client Write Speed



From: R Hedges K Fitzgerald M G and Stearman D “Comparison of leading parallel NAS file systems on commodity hardware” https://e-reports-ext.llnl.gov/pdf/457620.pdf

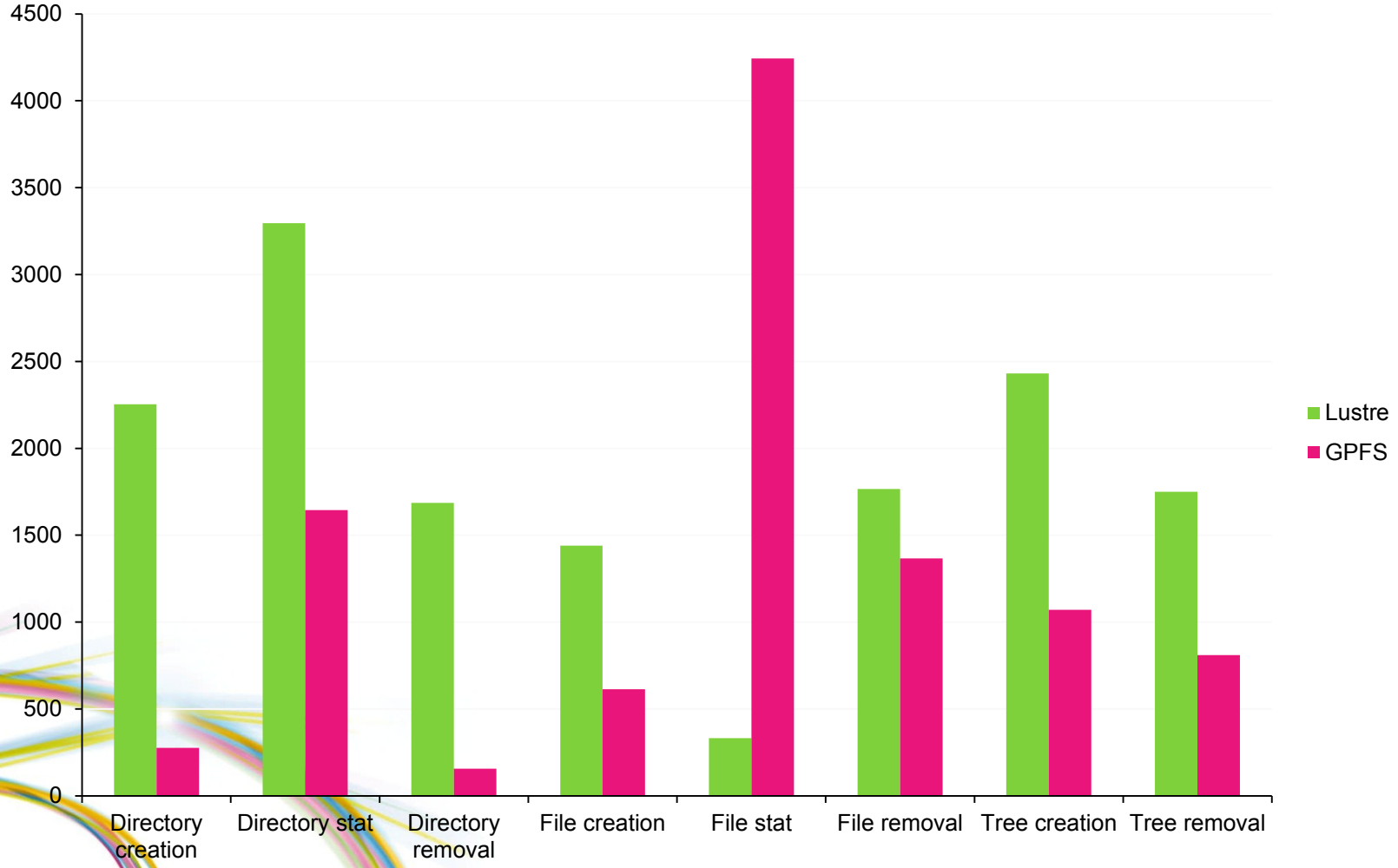
diamond

Data Throughput speeds

- Lustre 1.8
  - 115 MB, 1 GbE
  - ~400 MB/s, 10 GbE with checksums on
  - ~750 MB/s, 10 GbE with checksums off
  - Reports of ~ 1100 MB/s with IB
  - Client thread saturates 100% of 1 core.
- 2.3.0 Servers, 1.8.8 Clients
  - ~500 MB/s, 10 GbE with checksums on
  - ~700 MB/s, 10 GbE with checksums off
- 2.3.0 Servers, 2.3.61 Clients
  - ~650 MB/s, 10 GbE (no difference with or without checksums)
  - Still see some client thread saturation
- GPFS
  - 115 MB/s, 1 GbE
  - 1100 MB/s, 10 GbE
  - 2100 MB/s, QDR IB (single thread)
  - 3100 MB/s QDR IB (multiple threads)

diamond

Metadata speeds (ops/sec)



diamond

So, which Filesystem?

- Not a simple choice, but bottom line is that file system throughput should not limit detector.
  - Lustre fine for high aggregate cluster processing rates
  - GPFS clearly better for single point data rates
- Our current Lustre setup was explicitly designed for 1 GbE client systems.
  - Clearly insufficient now
  - New system must saturate 10 GbE.
- Testing the new DDN system with both Lustre and a GPFS file systems.
  - Final decision to be made next week.
- Will consider 40 GbE and QDR/FDR IB networks as a next step.

diamond

REAL-TIME PROCESSING

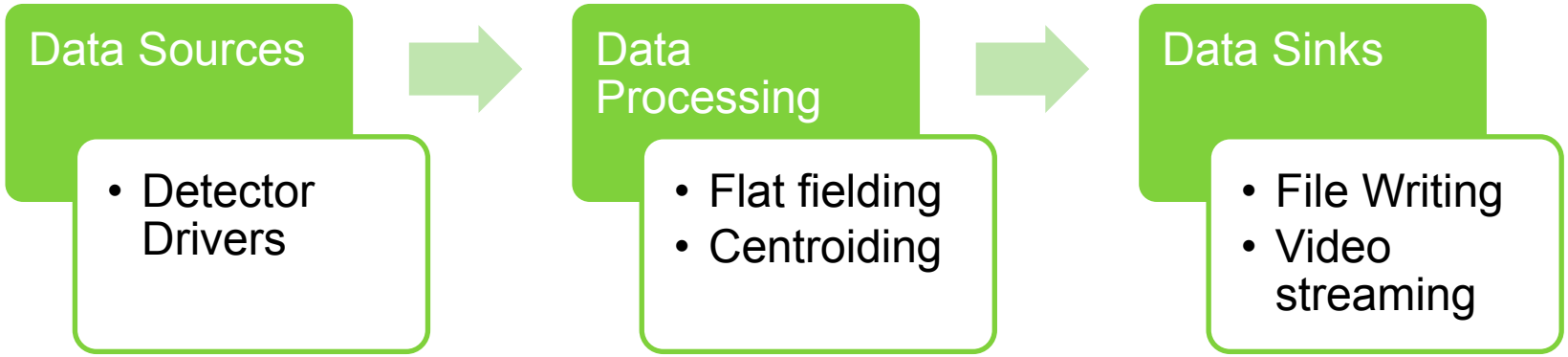
diamond

Diamond Software Goals

- We aim to do all we can to support the user to do the best science.
- Our responsibilities extend before, during and after the run.
- So, when it comes to data processing:
  - We must provide timely feedback on data quality
  - We must provide as much assistance as practicable to produce publication level data.
    - The more reduction we can do before they leave the quicker and easier it will be to publish.
    - The more publications the more likely Diamond will be successful.
- We have clear commitment from directors to invest heavily in software and IT.

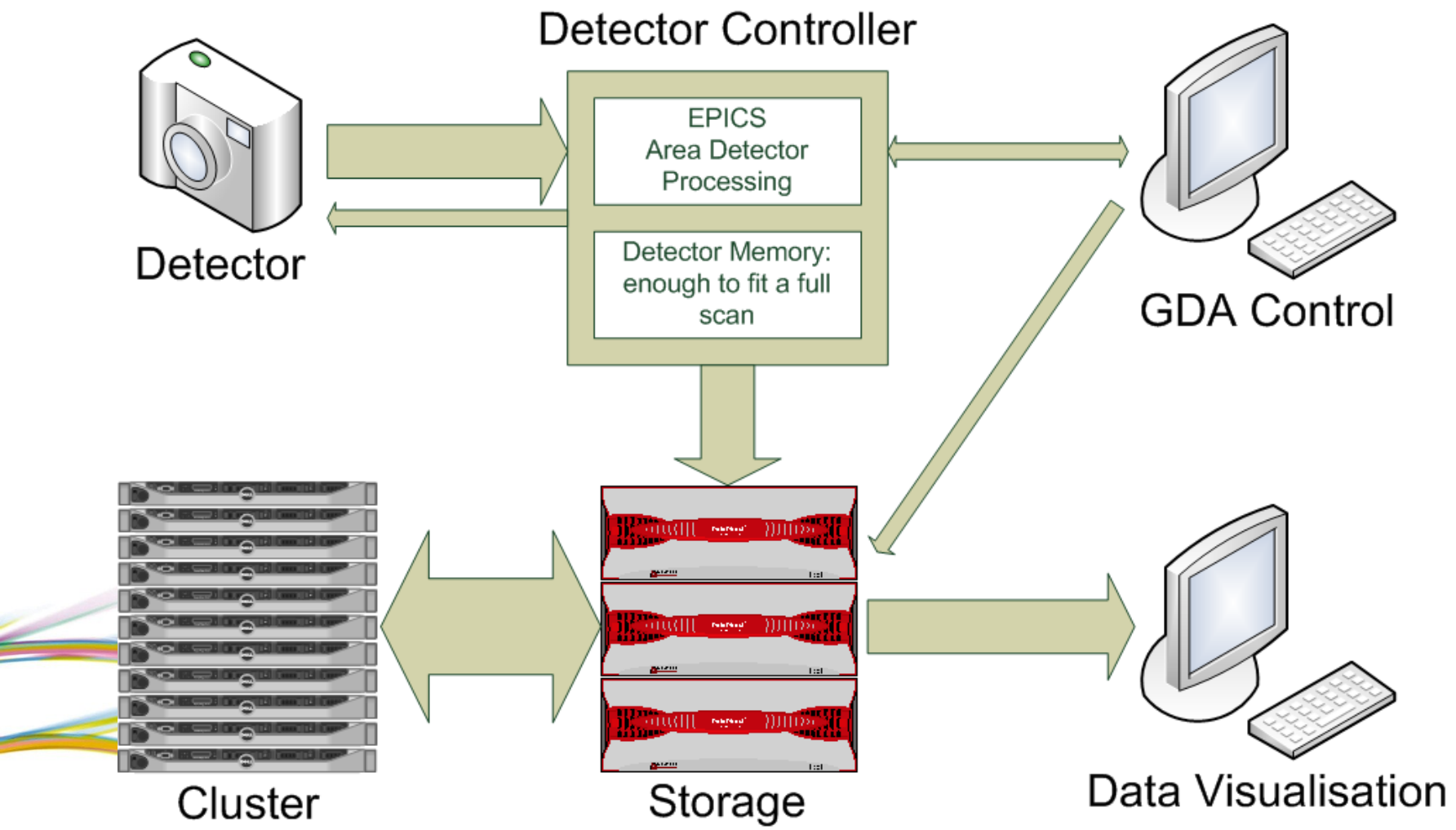
diamond

areaDetector Software Model



- Virtually all Diamond detector software is based on the areaDetector model.
- We have never had a case where area detector limits detector performance.

DLS High Data Rate Approach

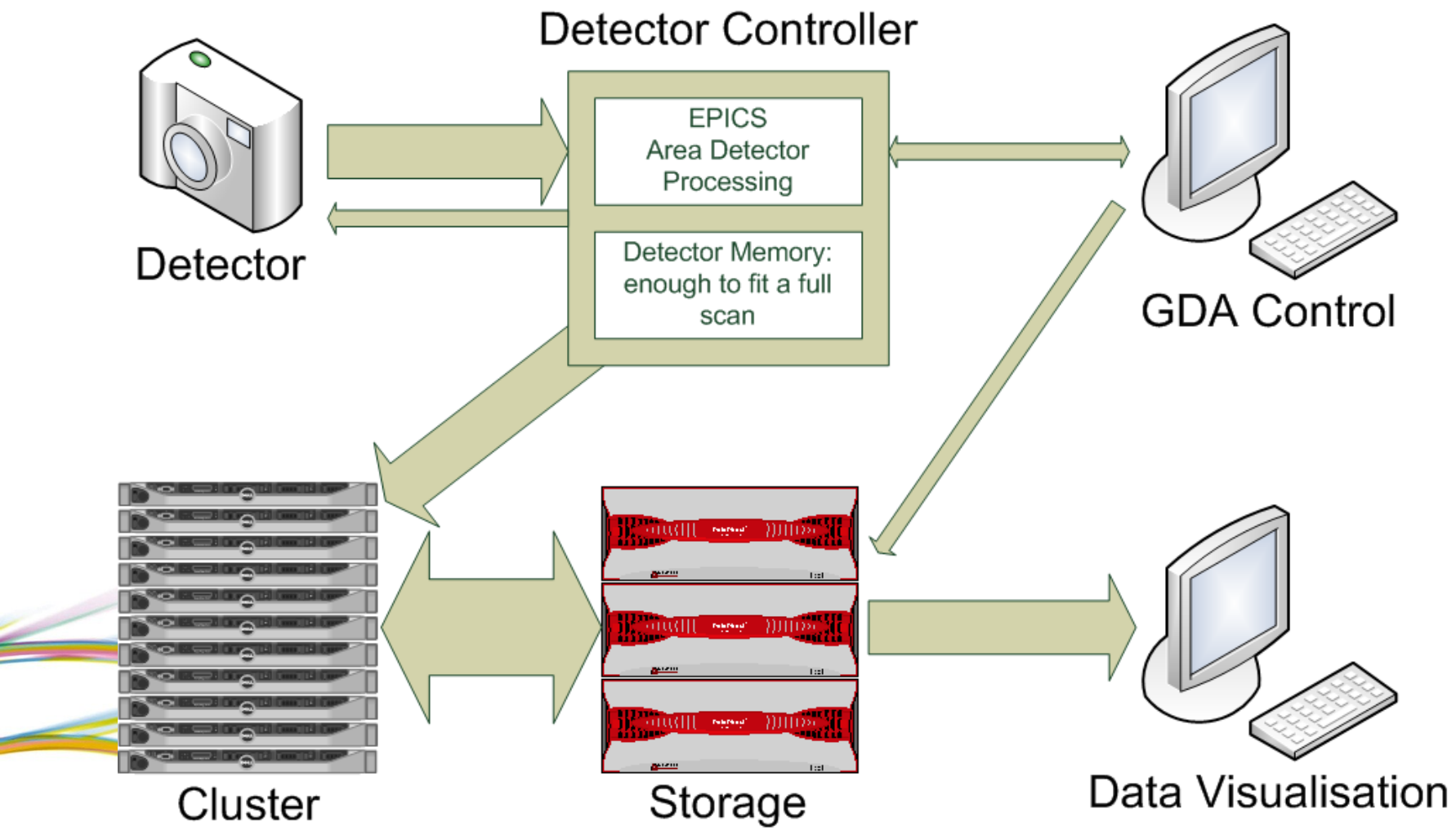


DLS High Data Rate Approach

- Largest scan identified so far is ~ 100 GB, so:
  - Buffer data in RAM of detector controller.
  - Migrate data to high-speed parallel disk.
  - Process data on clusters.
  - Compress data if possible to reduce problems
    - SAXS data compresses by ~50.
- What we are working on is:
  - Improving file writing so we can saturate a 10 Gbit link.
  - Providing similar support for Windows and Linux detector systems.
  - Provide processing on detector controller with areaDetector..
  - Providing feedback by displaying some of the images directly from memory in the detector controller using a fraction of the network link.
  - Improving the user interface (particularly in Tomography) to make the technical problems of large data rates transparent to the user.
- What we the next steps may be:
  - Passing the data directly from the detector controller to the compute systems (possible protocols include MPI, EPICS V4 or ActiveMQ).
  - Different forms of compression to improve transfer rates at the expense of CPU cycles.

diamond

DLS High Data Rate Approach

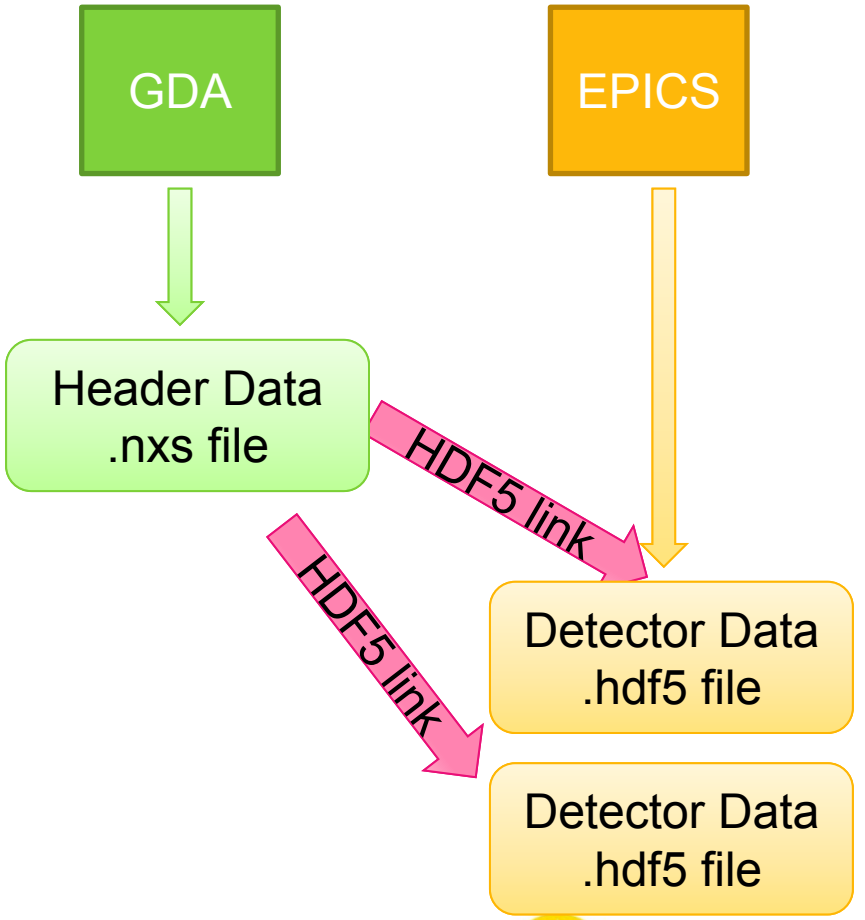


HDF5 DEVELOPMENTS

diamond

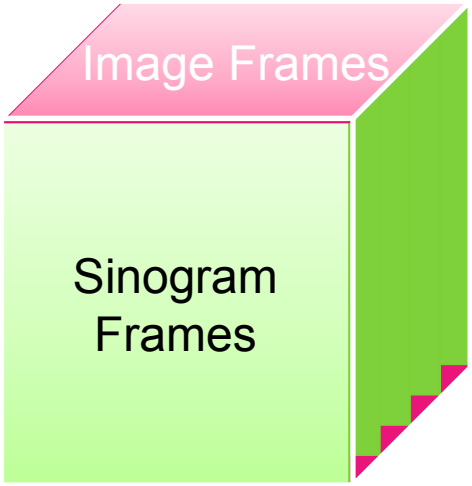
HDF5 Data Files

- EPICS and GDA both need to write the data file.
- We use HDF5 links to create one logical file from multiple real files.
  - Avoids file contention issues.
  - Allows detector files to be highly optimised for performance.
- The header data is written directly by GDA.
- The detector data is written using EPICS HDF5 area detector plugin.



Example: Tomography

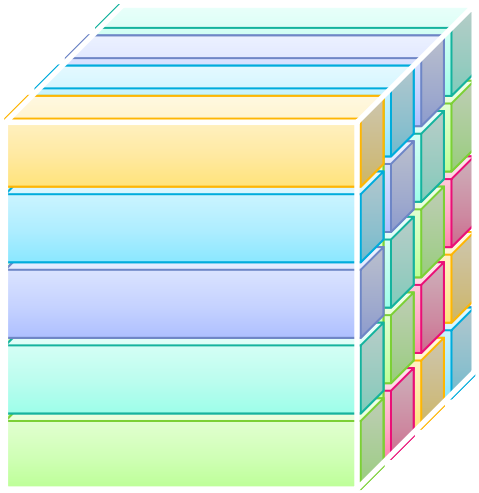
- Tomography scans are demanding:
  - Data rate ~ 500 MB/s.
  - Data size > 100 GB.
  - First operation is read data perpendicular to write direction.
- Classic matrix transpose problem
- Real challenge for typical cache design.
- Completely unsuited to running inside the GDA server.



diamond

Tomography data file format

- Data must be optimised for reading sinograms.
- All frames are written to a single file.
- File is arranged in chunks of a fixed number of rows and a fixed number of frames.
- The chunk size matches the Lustre stripe size so is written to a different Lustre server.
- Data is in cache until all frames in a chunk are written.



diamond

Recent HDF5 developments

- Last year the following developments were funded:
  - Writing of pre-compressed chunks.
    - Acquisition software can pre-compress data efficiently (i.e. in parallel), and write those directly to the HDF5 file without going through the HDF5 filtering mechanism.
    - Implemented by The HDF Group, funded by PSI and Dectris.
  - Filter plugins
    - User can supply a shared library at run-time that implements an HDF5 filter.
    - Allows the user to read and write compressed datasets with custom compression algorithms without recompiling the application by dynamically loading the corresponding filter.
    - Implemented by The HDF Group, funded by DESY.
  - Available in HDF5 1.8.11 (May 2013)

diamond

Future HDF5 developments

- Single Writer Multiple Reader (SWMR)
  - Data can be added and modified, but file hierarchical structure cannot be changed.
  - Writer task pre-creates file structure and then enables SWMR.
  - Reader tasks open file with read-only and SWMR flags and can then read data as it becomes available.
  - A Reader always sees a consistent HDF5 file - no errors occur while reading HDF5 metadata and raw data.
  - If the Writer crashes, it leaves the file in a non-corrupted state.
  - There is no performance penalty for file access and modifications under SWMR.
- Available in HDF5 v1.10.0 (2014)
- Funded by Diamond, Dectris and ????

diamond

Future HDF5 developments

- Parallel compressed writer
  - Allow multiple writers to write a compressed HDF5 in parallel
  - Have thought of 2 approaches:
    - Have parallel MPI jobs, and manage the compressed file layout by inter-job communication.
    - Have independent jobs writing separate files in parallel, but have an additional file referring to a aggregate HDF5 dataset via an array of soft links.
  - The latter requires a HDF5 dataset to be able to be defined from an array of HDF5 soft-links. The dimensions and chunk layout of the linked datasets will have to be compatible with the dimensions and layout of the parent.
  - Has many other use cases, not just parallel compressed writing...
  - Which approach would you prefer?

diamond

Summary

- We have come a long way in a short time.
- Diamond is committed to investing in software and IT.
- We support the efforts to improve the HDF5 file format as a foundation for us all to use Nexus as a common interchange format.

diamond