Integrating high speed detectors at Diamond

Nick Rees, Mark Basham, Frederik Ferner, Ulrik Pedersen, Tom Cobb, Tobias Richter, Jonathan Thompson...
(Diamond Light Source), Elena Pourmal (The HDF Group)
Introduction

• History
• Detector developments
  – Parallel detectors
  – Spectroscopic detectors
• HDF5 developments
  – HDF5 1.8.11 (Available now):
    • Dynamically loaded filter libraries
    • Direct write of compressed chunks
  – HDF5 1.10 (Being integrated):
    • New dataset indexing: Extensible array indexing.
    • SWMR
    • VDS
    • Journaling
History

• Early 2007:
  – Diamond first user.
  – No detector faster than ~10 MB/sec.
• Early 2009:
  – first Lustre system (DDN S2A9900)
  – first Pilatus 6M system @ 60 MB/s.
• Early 2011:
  – second Lustre system (DDN SFA10K)
  – first 25Hz Pilatus 6M system @ 150 MB/s.
• Early 2013:
  – first GPFS system (DDN SFA12K)
  – First 100 Hz Pilatus 6M system @ 600 MB/sec
  – ~10 beamlines with 10 GbE detectors (mainly Pilatus and PCO Edge).
• Late 2015:
  – delivery of Percival detector (6000 MB/sec).
DETECTOR DEVELOPMENTS
Potential EPICS Version 4 Model

Detector Controller

EPICS Area Detector Processing

Detector Memory: enough to fit a full scan

Cluster

Storage

GDA Control

Data Visualisation
Basic Parallel Detector Design

- Readout nodes all write in parallel
- Need a mechanism to splice data into one file.
Detector Wire Protocols

Data Receiver

Data Processing:
- 2 bit gain handling
- DCS subtraction
- Pixel re-arrangement
- Rate correction(?)
- Flat field
- Dark subtraction
- Efficiency correction

Data Compression

HDF5 File Writer

Detector Control

Control Driver

Configuration

Cmd

Status

Detector Array

Detector Control Software

Detector Data Stream (n copies)

Documented Controlled Interfaces

Beamline Control Software

Detector Engineer Software

Actual/potential network or CPU socket boundaries

Control Server

HDF5 file

Detector API

EPICS/Area Detector

Tango/Lima

Calibration Software

HDF5 file

Detector Engineer Software
Spectroscopic Detectors

- areaDetector is poorly named…
  - Base class is asynNDArrayDriver, but this name is not so catchy…
    - NDArray* classes provide basic functionality
    - Core plugins derive from NDPluginDriver and many will work with any NDArray.
    - Most popular plugins are the file writing plugins that get data to disk.
  - Basic areaDetector class is really NDDriver
    - Provides methods for reading out a typical areaDetector
    - The methods aren’t so good for other types of detectors, e.g.:
      - Spectroscopic (MCA like) detectors.
      - Analogue (A/D like) detectors.
Proposal for new ND Drivers

• Need a set of basic driver classes for other types of NDArrays
  – NDMCADriver (or NDSpectraDriver)
    • Generates 2-D array of energy vs detector channel
    • 3rd dimension can be time.
  – NDADCDriver (or ND DigitizerDriver)
    • Generates 1D array of values from a set of ADC’s
    • 2nd dimension can be time.

• Each driver can feed existing plugins, but also could benefit from specialist plugins.
HDF5 DEVELOPMENTS
HDF5 key points

- HDF5 is mature software that grew up in the HPC environment.
- It is a widely used standard and has the richest set of high performance functionality of any file format.
- It allows rich metadata and flexible data formats.
- It has some caveats we know about:
  - HDF5 is single threaded.
  - pHDF5 relies on MPI, which doesn’t happily co-exist with highly threaded architectures like EPICS.
  - pHDF5 is not as efficient as HDF5.
  - pHDF5 doesn’t allow compression.
  - Files cannot be read while they are written.
Recent Developments: Release 1.8.11

- **H5DO_write_chunk**
  - Funded by Dectris and PSI
  - Improves writing compressed data by:
    - Avoiding double copy of filter pipeline
    - Allowing optimised (e.g. multithreaded) compression implementations

- **Pluggable filters**
  - Funded by DESY
  - Allows users to provide filters as a shared library that is loaded at runtime.
  - Search path set by environment variable: HDF5_PLUGIN_PATH
Current developments: Release 1.10

• File format changes that need a major release:
  – Improved dataset indexing:
    • New B-Tree implementation
    • Extensible array indexing
  – Journaling
  – Virtual Object Layer
  – Single Writer Multiple Reader (SWMR)
    • Funded by Diamond, Dectris and ESRF
  – Virtual Data Set
    • Funded by Diamond, DESY and Percival Detector

• Beta release July 2015
CONCURRENCY:
SINGLE-WRITER/MULTIPLE-READER
New data elements...

... are added to a dataset in the file...

HDF5 File

... which can be read by a reader...

with no IPC necessary.
• Implemented for raw data “append only” scenario
  • No creation or deletion of the datasets, groups, and attributes is allowed at this time
• Product is under integration
  • Works on GPFS, Lustre, Linux Ext3, Ext4, FreeBSD USF2, OS X HDFS+
  • Documentation
    http://www.hdfgroup.org/HDF5/docNewFeatures/
  • Source
VIRTUAL OBJECT LAYER
Virtual Object Layer

• Goal
  - Provide an application with the HDF5 data model and API, but allow different underlying storage mechanisms

• New layer below HDF5 API
  - Intercepts all API calls that can touch the data on disk and routes them to a Virtual Object Driver

• Potential Object Drivers (or plugins):
  - Native HDF5 driver (writes to HDF5 file)
  - Raw driver (maps groups to file system directories and datasets to files in directories)
  - Remote driver (the file exists on a remote machine)
Virtual Object Layer

- HDF5 API
- VOL
  - Raw Mapping
  - Native (H5)
  - Metadata Server
  - Remote Access

Virtual File Layer
- VFL
  - mpiio
  - posix
  - sec
  - split

File System
- VOL plugins
- Remote machine(s)
Network VOL Plugin

Clients

- Application
- HDF5 API
- h5netvol plugin

Server

- h5netvol plugin
- HDF5 API
- HDF5 library
- HDF5 file

TCP, MPI
Other Capabilities

• Allows concurrent access, even by multiple writers
  • Could even be useful on a single machine
• Includes locking scheme that can be used to control access to objects
DATA INDEXING
Indexing and HDF5

- New APIs for indexing and querying of both structure and contents of HDF5 containers
- H5Q API defines query to apply to a container
  Create/combine queries (OR, AND)
  - Basic operators supported ($\leq$, $\geq$, $=$, $\neq$) on either dataset/attribute values, link/attribute names
- HDF5V API retrieves data
- HDF5X API adds third-party indexing plugins
VIRTUAL DATASET
Challenge

• How to view data stored across the HDF5 files as an HDF5 dataset on which normal operations can be performed?
  • High-level approach
    • Special library that applications like MATLAB and H5Py will need to use
    • Example: THREDDS Data Server based on OPeNDAP
      http://www.unidata.ucar.edu/software/thredds/current/tds/TDS.html
  • Native HDF5 implementation
    • Transparent to applications
Virtual Dataset: Excalibur Detector Use Case

- Series of images aligned in time (t)
- Virtual Dataset VDS (VDS.h5)
  - Image at time t_2
- Datasets:
  - A (a.h5) at t_2
  - B (b.h5)
  - C (c.h5) at t_2
  - D (d.h5)
  - E (e.h5) at t_2
  - F (f.h5)

September 25, 2014
Example: “Printf” Source Generation

Virtual Dataset VDS

VDS.h5

f-1.h5  f-2.h5  f-3.h5  f-N.h5

File names are generated by “printf” capability
HDF5 1.10.0 Roadmap

December 2014 – January 2015
VOL           SWMR           HDF5 1.10.0-alpha1

February-March 2015
Indexing      HDF5 1.10.0-alpha2

June 2015
VDS           HDF5 1.10.0-alpha3

July 2015
HDF5 1.10.0-beta

August 2015
HDF5 1.10.0

Features and release dates are tentative; may change

September 25, 2014
Thank you for your attention…