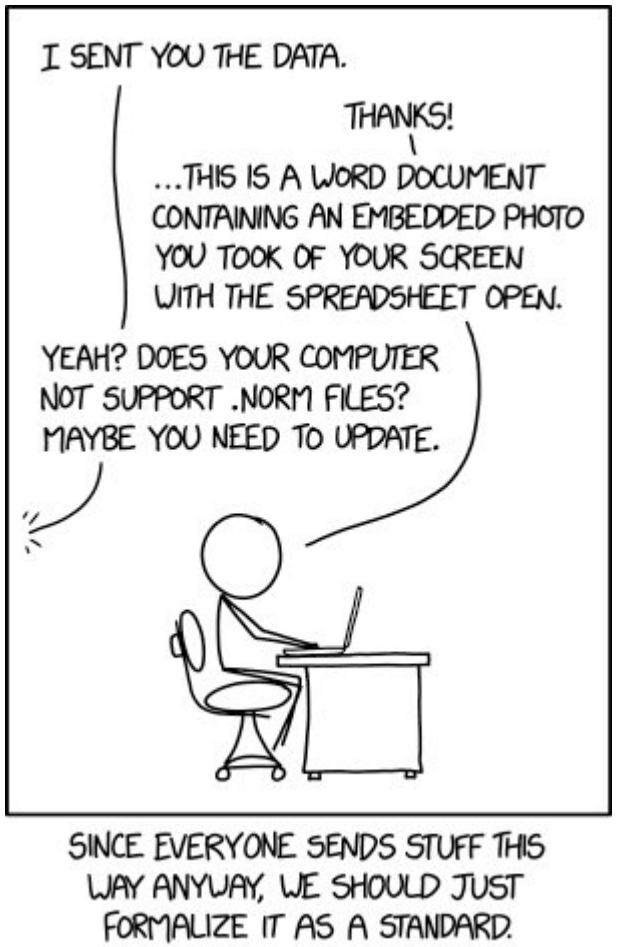
HDF5 with LImA

The ESRF use case

for **fast 2D Detector data**

with **LImA** today

and **distributed LImA2** soon

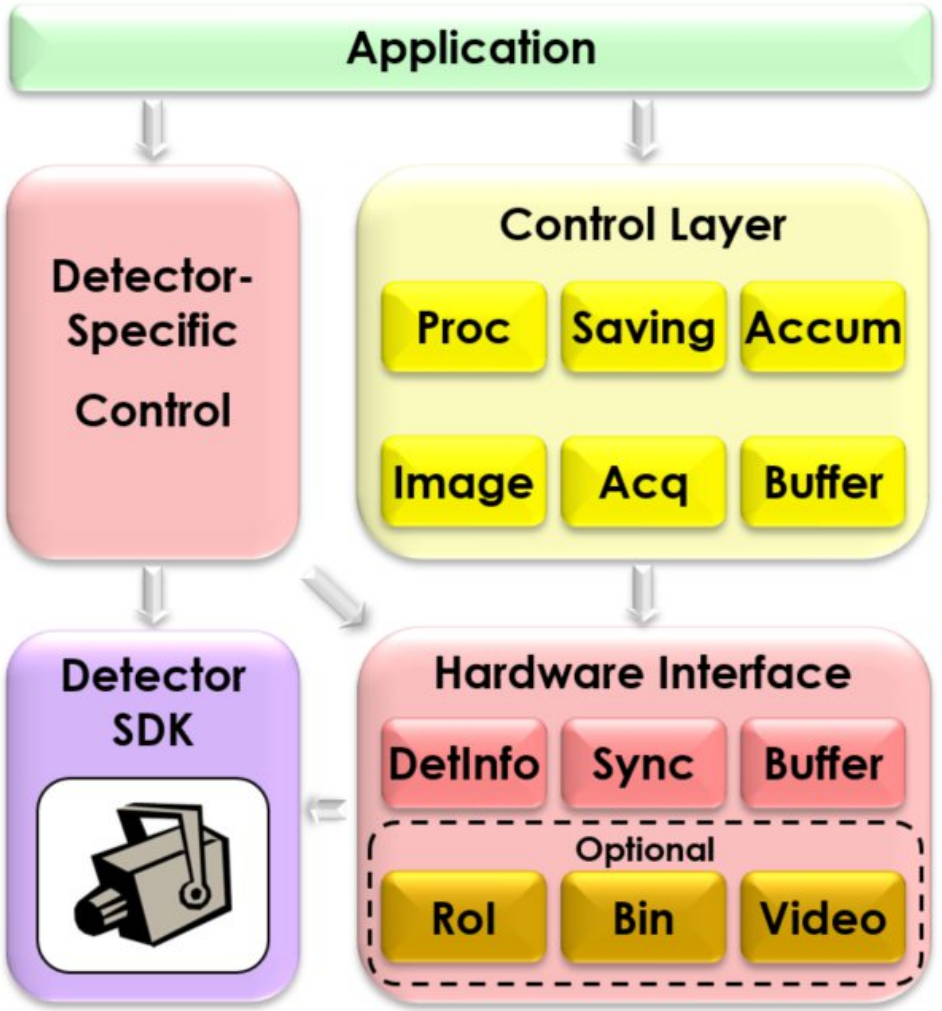


LImA overview

Library for Unified Detector **Control,** Image (2D) **DAQ** and **Processing** available in C++, Python, Tango

Open Source - GPL

https://github.com/esrf-bliss/Lima



LImA overview - DAQ

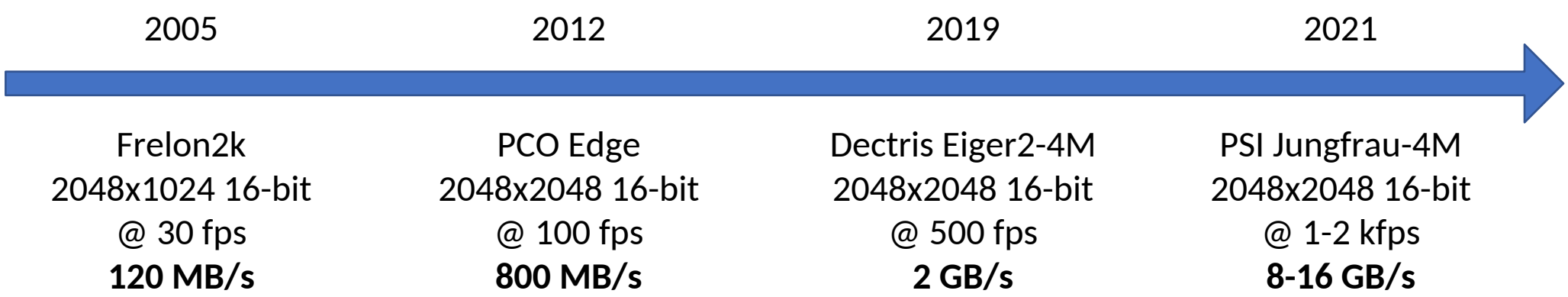


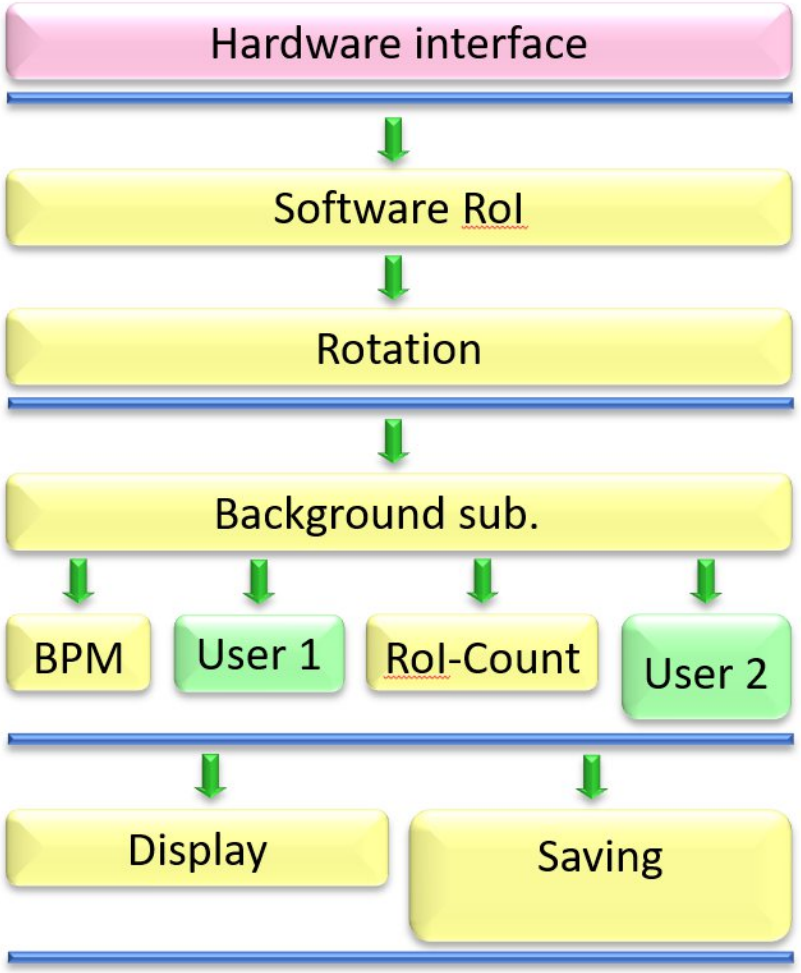
Fig. Fast 2D detector (raw) data throughput over time

+30 detector plugins for X-ray direct/indirect detection

- X-ray photon counting devices
- X-ray integrating devices
- Visible light integrating sensors

LImA overview - Processing

- Multi-threaded frame processing chain
- Common processing algorithms for synchrotron light sources:
  - Frame reconstruction: pixel geometry & intensity conversion
  - Pixel Binning, Region of Interest (RoI), Flip/Rotation
  - Background subtraction, Flat-field correction, Mask
  - Rectangular & Arc RoI statistics, Beam Position, X/Y Projection, Peak Finding, …
  - File saving with parallel compression: EDF, HDF5, CBF, TIFF, …



HDF5 in LImA - Requirements

- Producer perspective
  - Data integrity
  - Real-time context
  - Saving performance
- Consumer perspective
  - One or multiple executions
  - Reading performance
  - Reading as we write (SWMR)
- Analysis specific requirements
  - Tomography
    - high data-rate
    - “orthogonal” read pattern (slice in the slowest dimension)
  - X-ray Photon Correlation Spectroscopy XPCS
    - high frame-rate
    - low photon count appropriate for sparse storage
  - Macro-molecular Crystallography (MX)
    - peaks finding appropriate for sparse storage

HDF5 in LImA - Implementation

- **Native** HDF5 (to File)
- **Chunking** per frame (not configurable)
- **Direct chunk API** to bypass dataspace/filter pipeline and enable compression in parallel (BSLZ4, GZIP…) reusing LImA's thread pool
- **Nexus** Convention (to some degree)

Saturate central storage bandwidth

- IBM GPFS +2GB/s
- 25 Gbit/s network

Issues:

- Global lock (single effective write stream)
- Corruption on crash
- Direct chunk "backdoor" used

LImA2 overview - Motivations

- **Scalability** for Modular fast detectors
  - Mosaics of tiled independent modules
  - Detector size can scale to +10 Mpixels
  - Acquisition frame rate increases +10kHz
  - Data throughput +30GB/sec

- **Low latency** data processing
  - On-the-fly image analysis
  - Fast experiment feedback (<10 msec)
  - Data reduction for off-line processing

LImA2 overview - Distributed Topologies

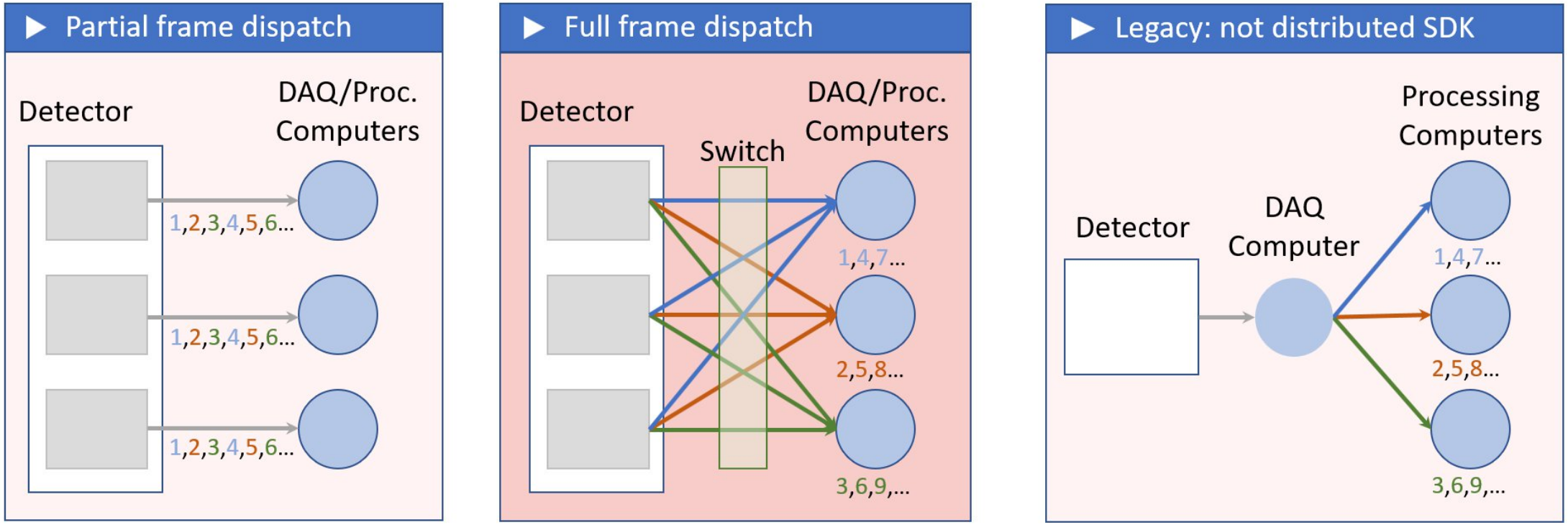


Fig. Topologies

HDF5 in LImA2 - Yet Another C++ Wrapper

Many implementations available

- Bundled with HDF5 (deprecated?)
- h5cpp from Steve Vargas
- h5cpp from Data Management and Software Centre - European Spallation Source (ESS DMSC)
- h5xx from Felix Höfling

`unique_handle` / `shared_handle`

- modeled on `std::shared_ptr`
- manage resources (RAII)
- used directly with the C API

```
using unique_file_hid_t =
    unique_handle<
        hid_t,
        decltype(&::H5Fclose),
        ::H5Fclose,
        raii::not_negative
    >;

using shared_file_hid_t =
    shared_handle<
        unique_file_hid_t
    >;
```

HDF5 in LImA2 - OneTBB Flow Graph integration

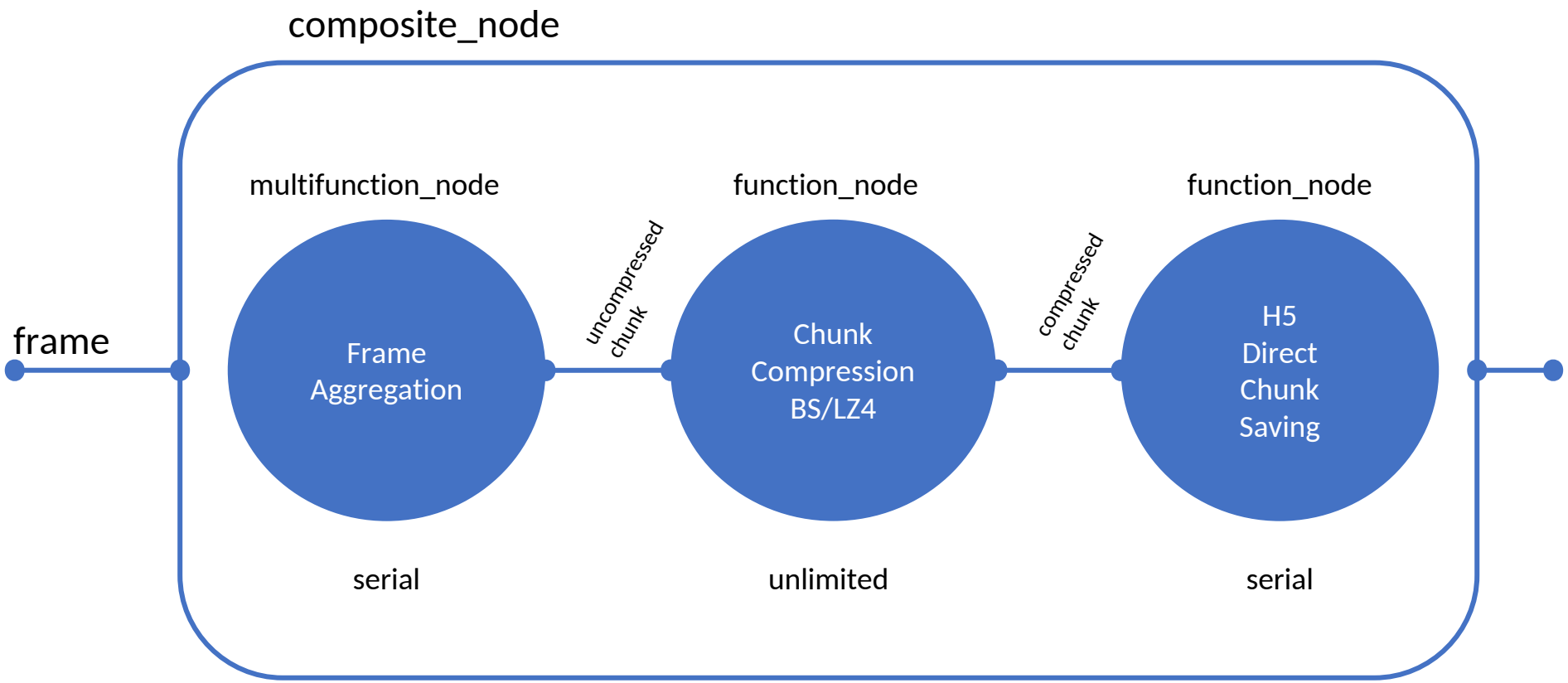
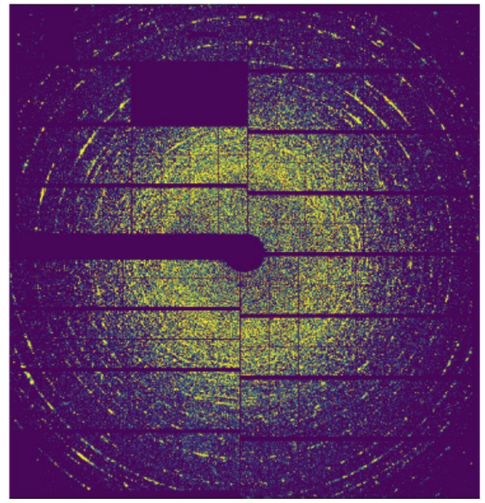
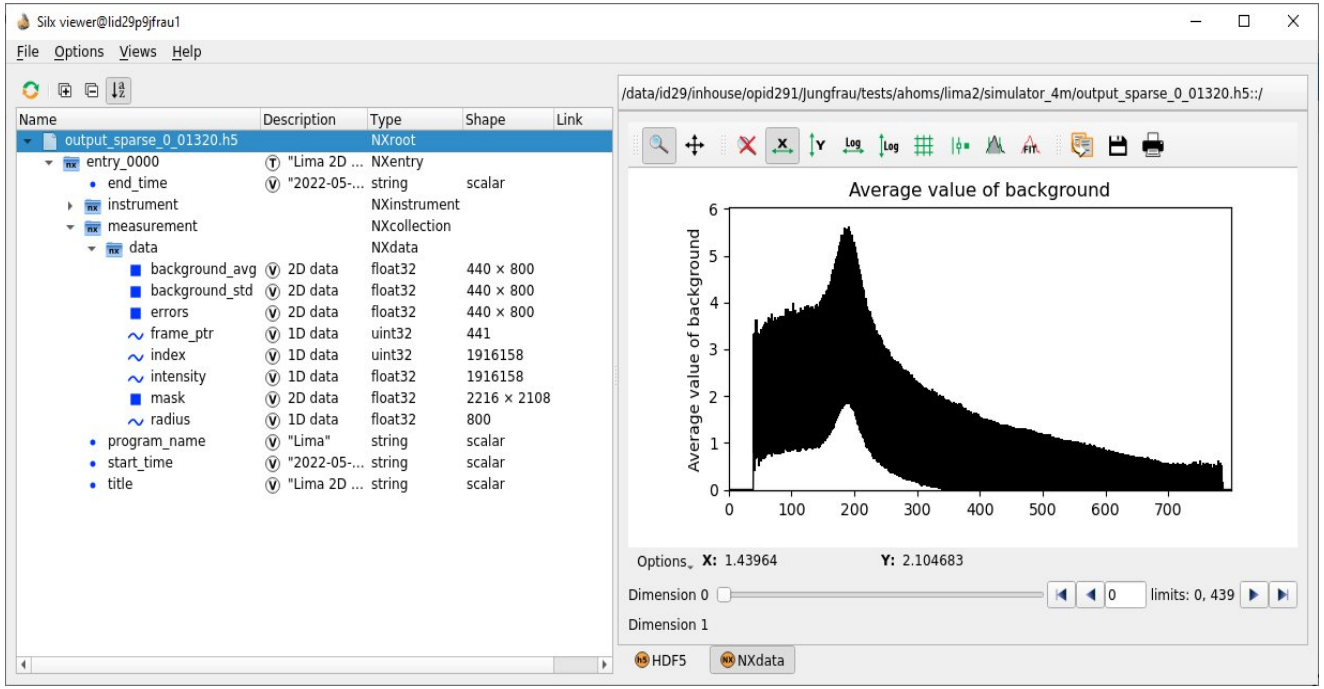


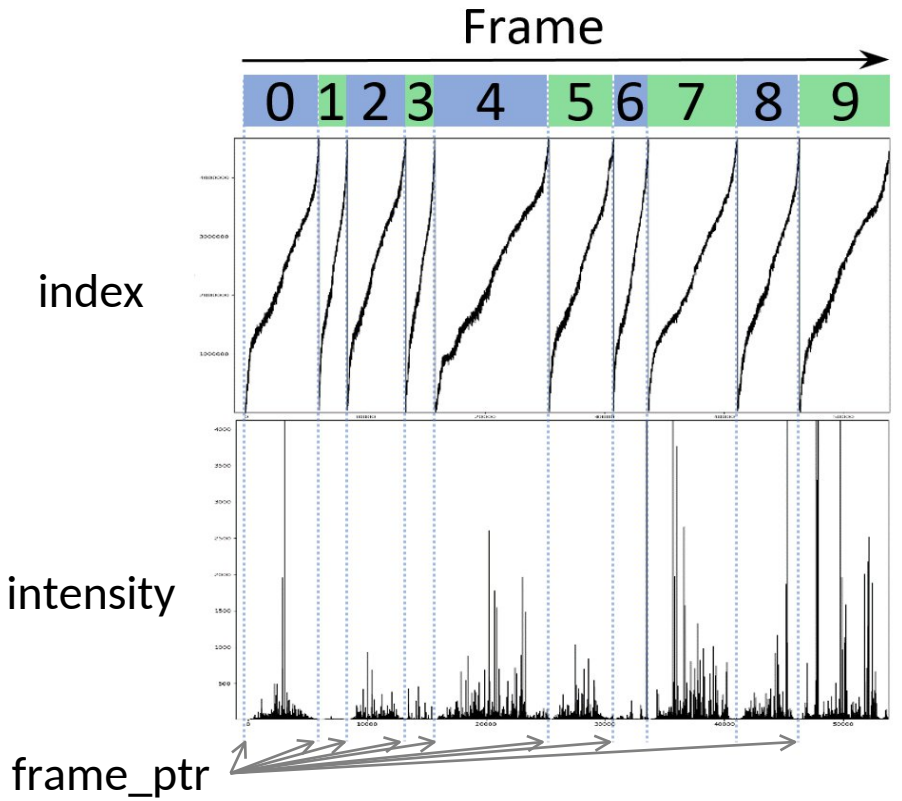
Fig. io_hdf5_node

HDF5 in LImA2 - SMX use case



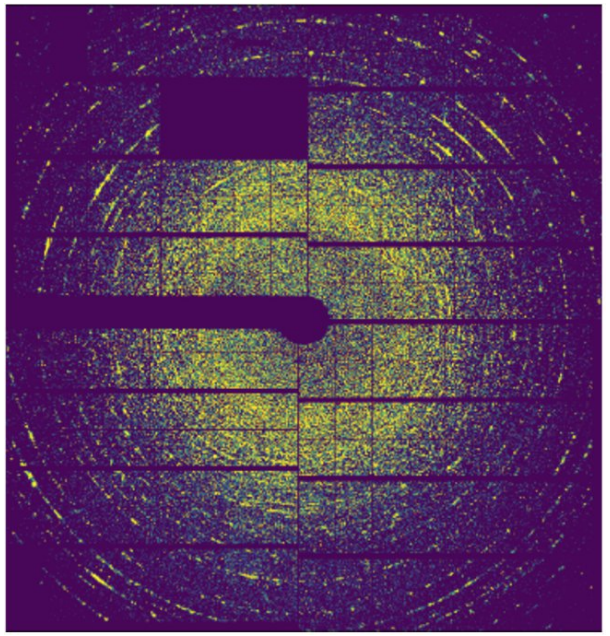
- Hardware acceleration:
  - GZIP with IBM Power9
  - GPU: Jungfrau pedestal correction, pyFAI & peak finder
- Sparse data CSR a la pyFAI + Background





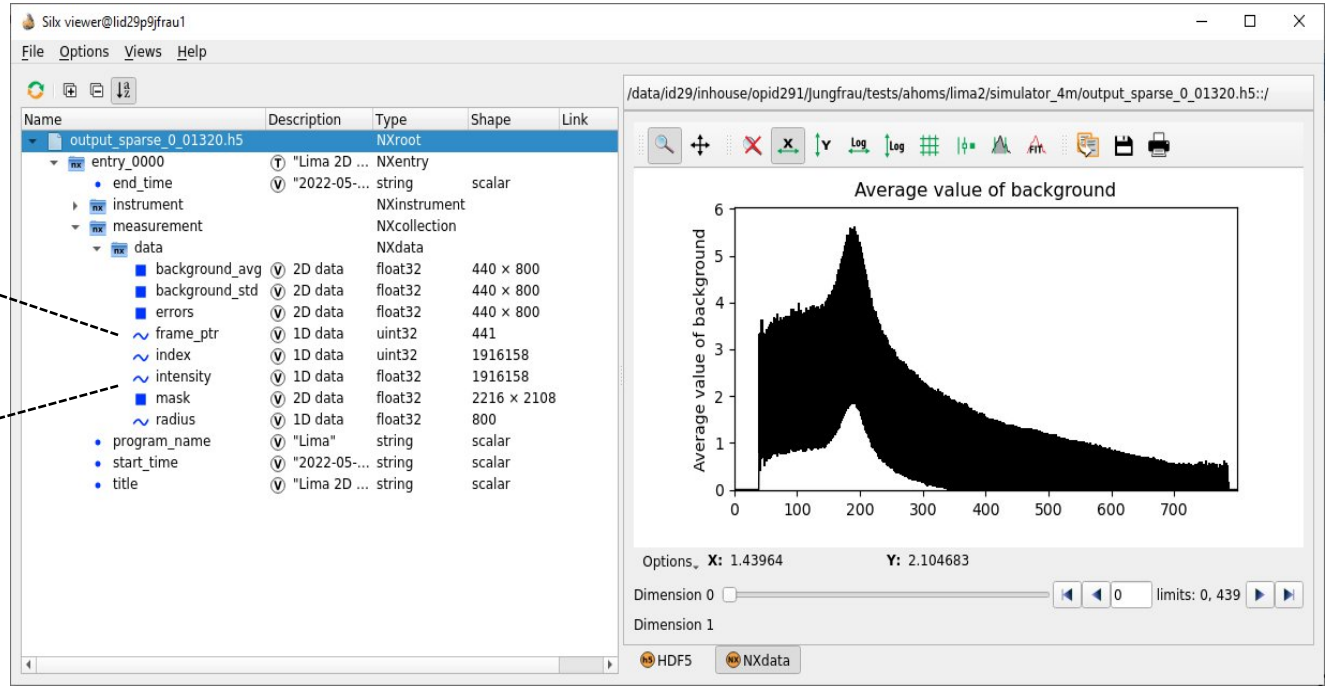
HDF5 in LImA2 - SMX use case

- Hardware acceleration:
  - GZIP with IBM Power9
  - GPU: Jungfrau pedestal correction, pyFAI & peak finder
- CSR representation a la pyFAI + Background



$ h5ls -r sparse.h5

/sparse_frames/frame_ptr Dataset {441}
/sparse_frames/index Dataset {1916158}
/sparse_frames/intensity Dataset {1916158}



Exploring HDF5 new features

- **Multi-thread** support
  - Application should provide “executors”
- **Asynchronous** API
  - Blocking I/O wastes worker thread
  - Move to new VFD / VOL?
- **Sparse Data** Support (SMX Peaks, XPCS)
  - XPCS: with fast detector, probability of 0 photon
  - SMX: Background identification and peak finding
  - compression filter?
  - Native support?
- Data **aggregation** (from multiple processes)
  - VDS
  - ParallellHDF5 / MPI-IO
  - Aggregation of sparse data
- **HERMES**
  - Fast scalable data access
  - Towards low-latency off-line analysis
- **GPUDirect** Storage

Discussions

- One size fits all or are we asking too much from HDF5?

- Inspiring technologies
  - HERMES Multi-tier storage (for fast data access) - ODA
  - HDFql for metadata (or H5Q and H5X APIs)
  - ADIOS for flexible I/O (to adapt to Processing)
  - TileDB (for Sparse data) - SMX
  - InfluxDB DB (for Time-Serie data) - XPCS