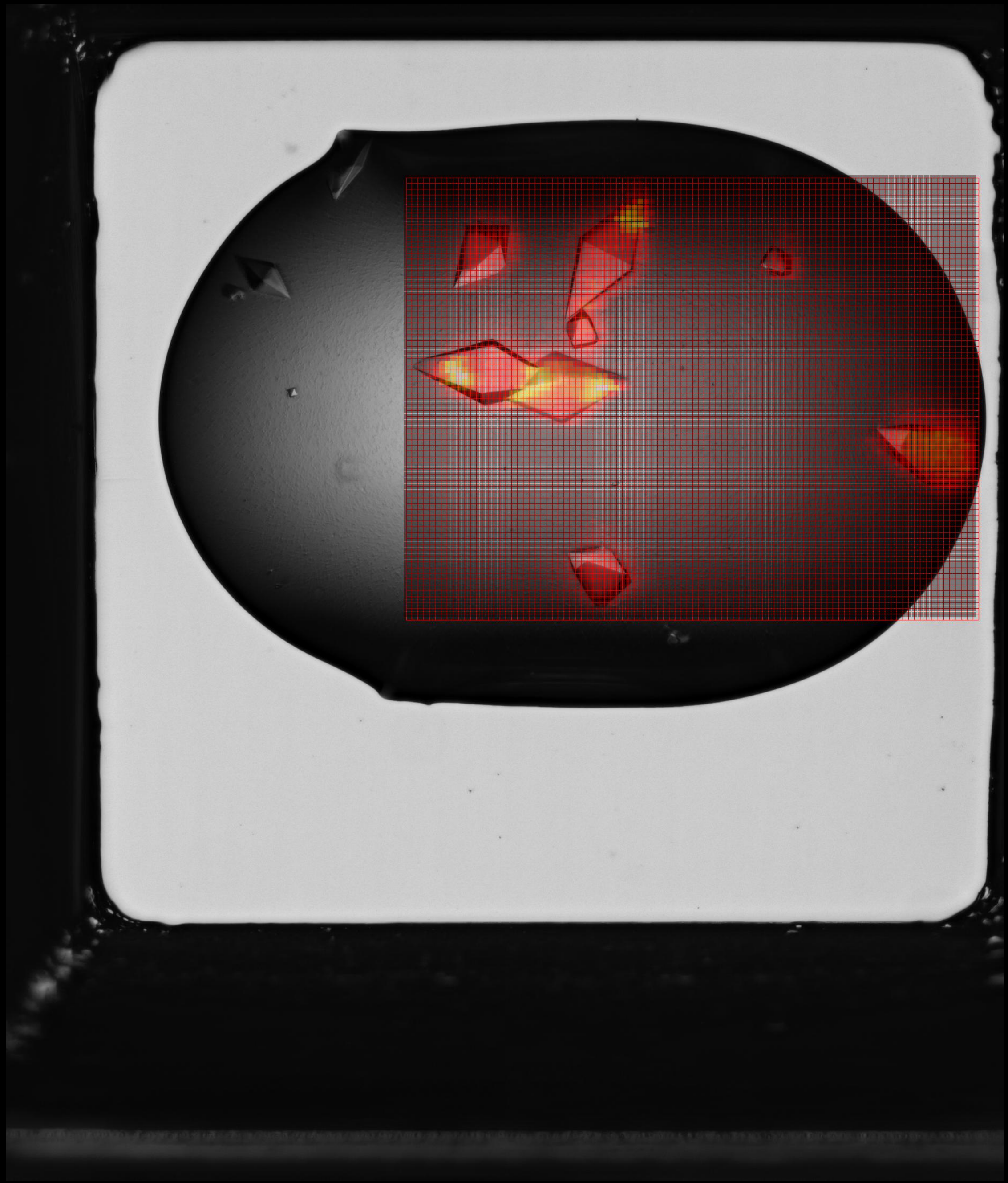
Dealing with Data Rates

HDRMX November 2019

Raster Scan

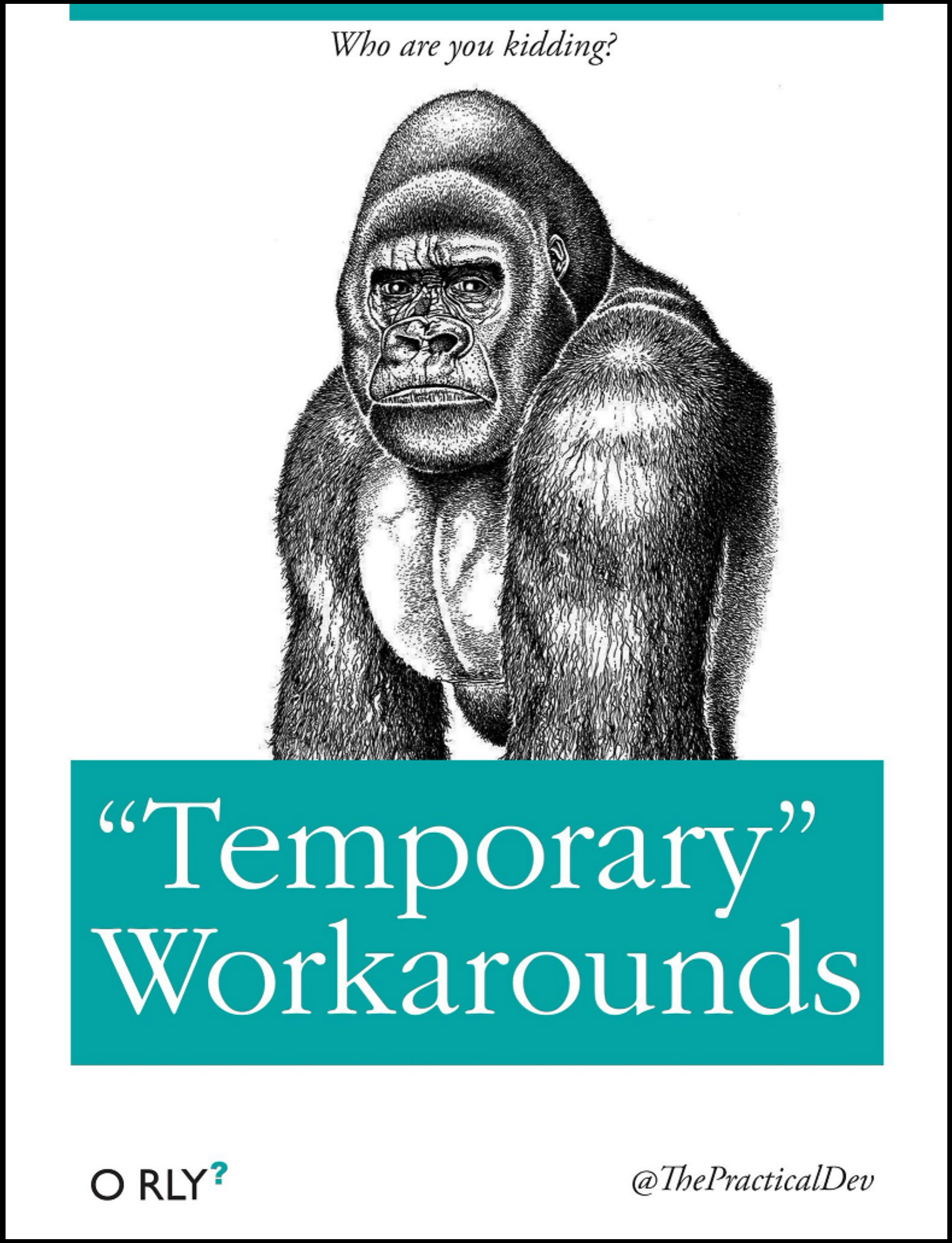
Raster Scans

- First use case VMXi - not interactive so analysis can happen whenever
- Grid scans of ~ 5-20 thousand frames typical
- Initial implementation - use one node to grind through the HDF5 representation after acquisition

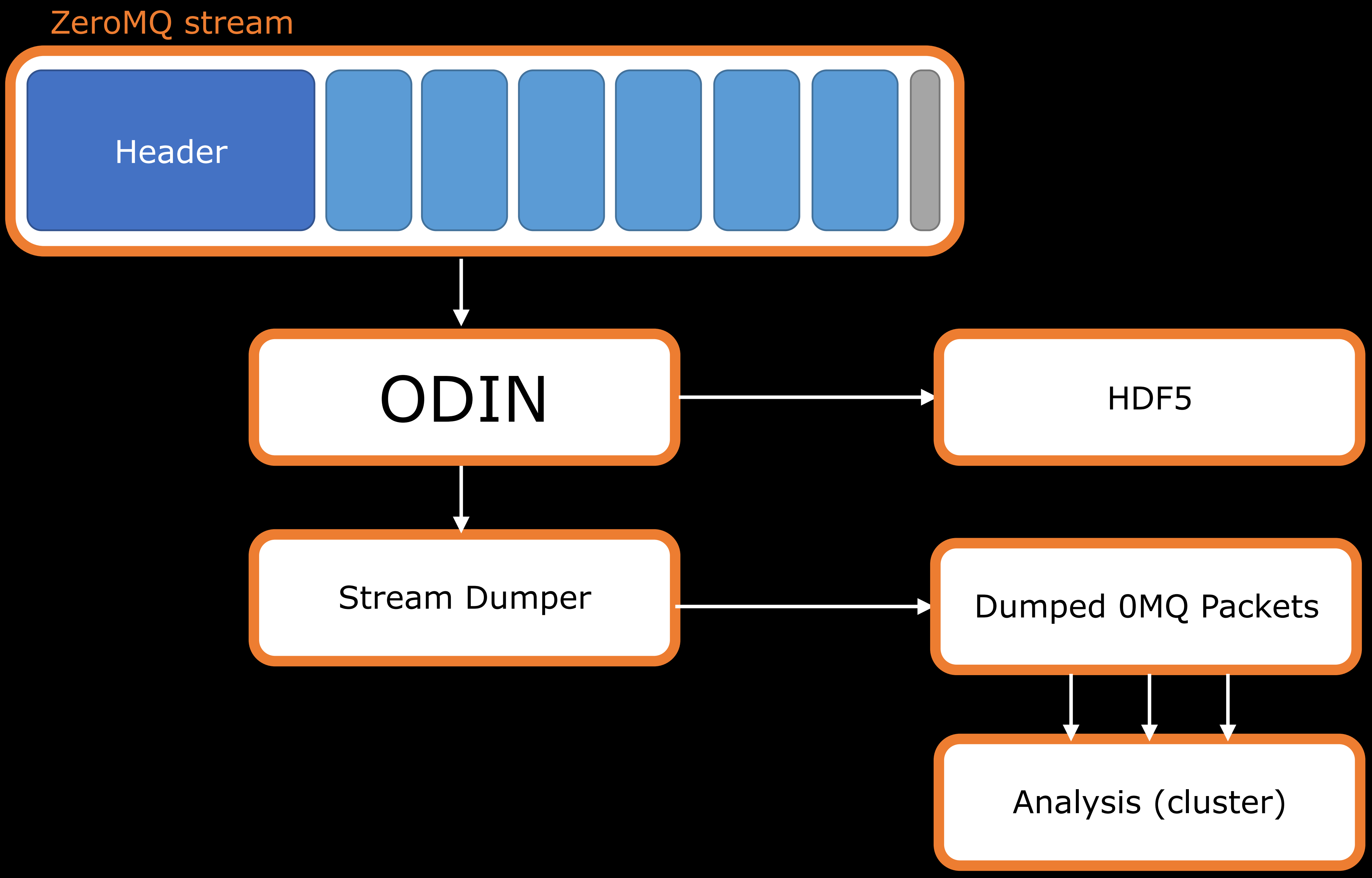


Raster Scans

- Interactive use analysis after acquisition is not acceptable - far too much latency
- Processing was limited to single node
- Alternative - the hack that shall not be named



The Hack


ZeroMQ stream
Header
ODIN
HDF5
Stream Dumper
Dumped 0MQ Packets
Analysis (cluster)

Hack Details

- Header + image packets written to ${VISIT}/tmp/${DCID}

- Extend DIALS to read these natively

- Use CBF equivalent analysis

- Benefits - analysis from the stream while collecting, parallel processing, end user experience far improved, file system builds in elasticity

- Costs - 50% load increase on DAQ system, 2 x write load on file system, >>>> inodes

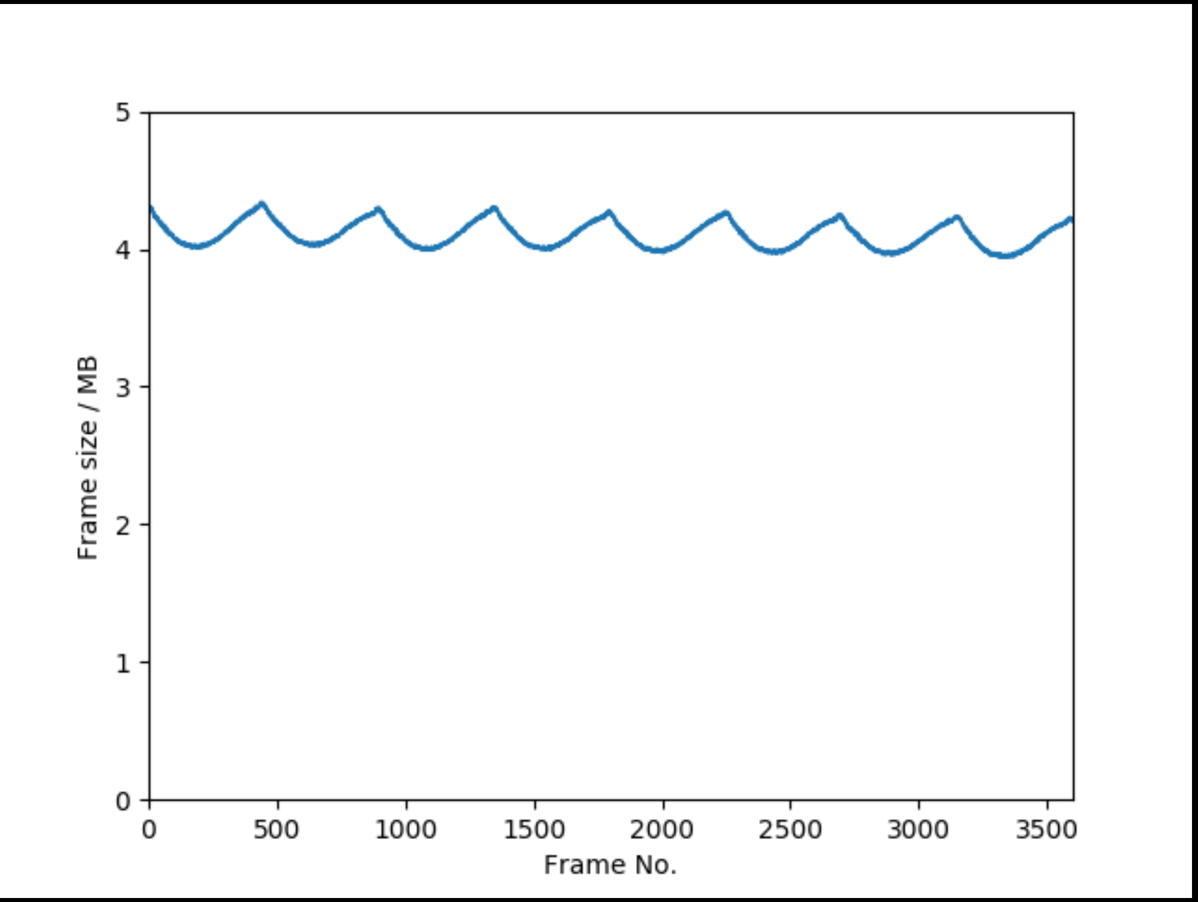
Rotation Processing

Data Rates

- Issues used to be inodes / s & MB / s
- Bigger issue now Gpixel / s - if measured carefully the data compress very well
- Even typical data get compression much better than CBF byte offset (limited to 1 byte / pixel)

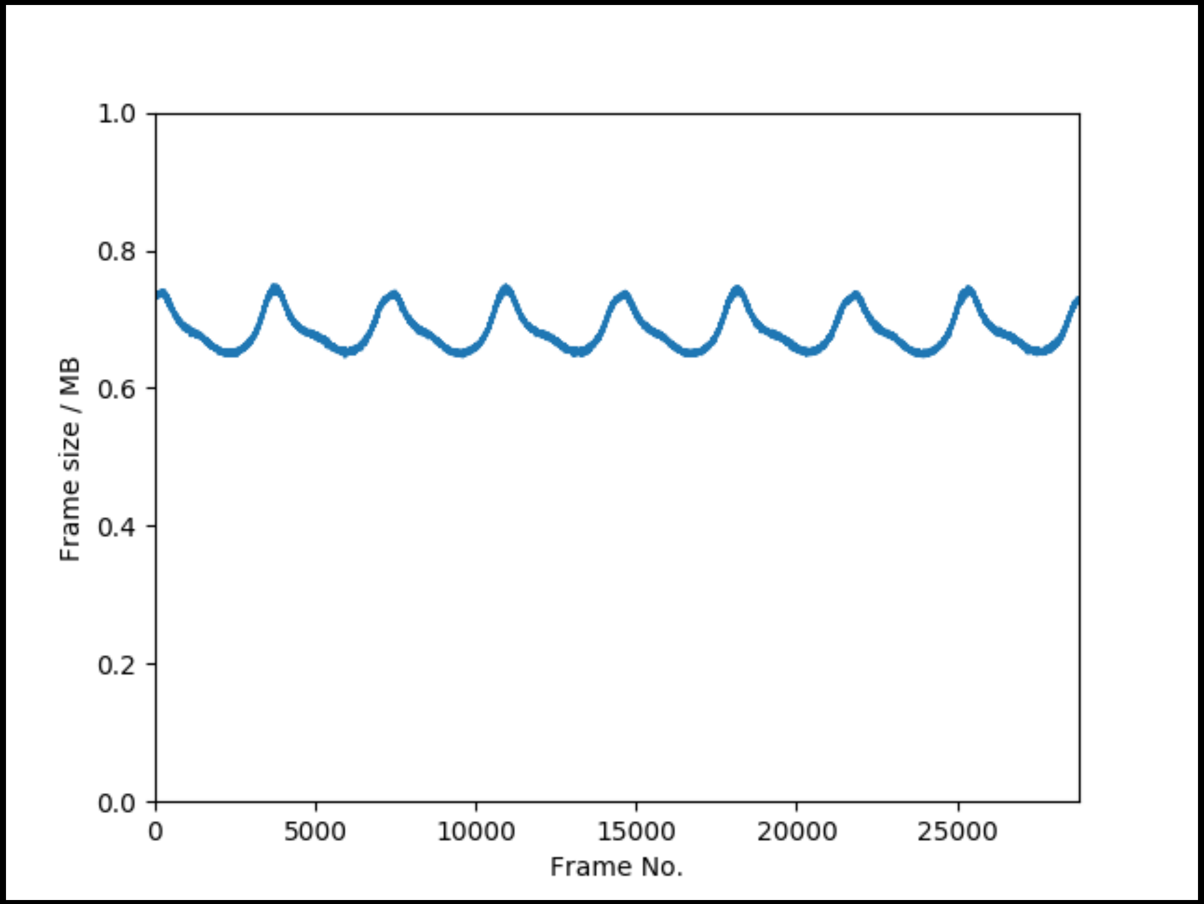
Data Volumes

- Typical data set - around 2 - 4 MB / frame - so 1 - 2 *bits* / pixel
- 3,600 @ 0.1° around 15 GB
- Pilatus 6M CBF around 20 GB for same
- 6 inodes not 3,600 🙂
- GPFS very happy



Data Volumes

- Sparse data set - < 1 MB / frame - so << 1 *bit* / pixel
- 28,800 @ 0.05° around 20 GB
- Pilatus 6M CBF around 170 GB for same
- GPFS very happy
- Processing very unhappy!



Processing Challenges

- For radiation sensitive samples with a photon counting detector high multiplicity / low dose rational strategy

- With detector capable of 50° / s @ 0.1° literally nothing preventing this strategy - 4 turn data set takes < 30s

- Any radiation damage spread across reciprocal space

- Data volume modest - comparable to 1 turn data set with 4 x transmission as compression close to entropy limit

Processing Challenges

- Spot finding / integration time proportional to no. pixels
- Scaling time proportional to no. reflections measured
- Eiger 16M ~ 2.7 x as many pixels
- Rational strategy 4 x as many frames, 4 x as many reflections
- Spot finding / integration 10 x as expensive, scaling 4 x as expensive at least

Responses to Date

- In DIALS - speed week - identify the bottlenecks and try to resolve them - MTZ output was a major one - writing batch headers ~ O(n^3) process?! also trim no. reflections used for symmetry analysis etc.

- Spot finding / integration - memory bandwidth limited? Can scale across machines e.g. fast_dp

- Scaling minimisation problem - serial-ish - want fast CPU's (GHz) therefore lower core counts

Kaizen

Continuous Improvement

No “Quick Wins”

- Already the DIALS / XDS etc. reasonably efficient
- Finding cases where the code is O(n^2) etc. key - try to reduce this
- Tuning hardware can help - some %
- Real benefits will come only from large number of small improvements - hence Kaizen approach
- Trying to “keep up” though - this will require massive investment