Detector DAQ at DLS

Gary Yendell – Beamline Controls Core Team

diamond

Detectors at DLS

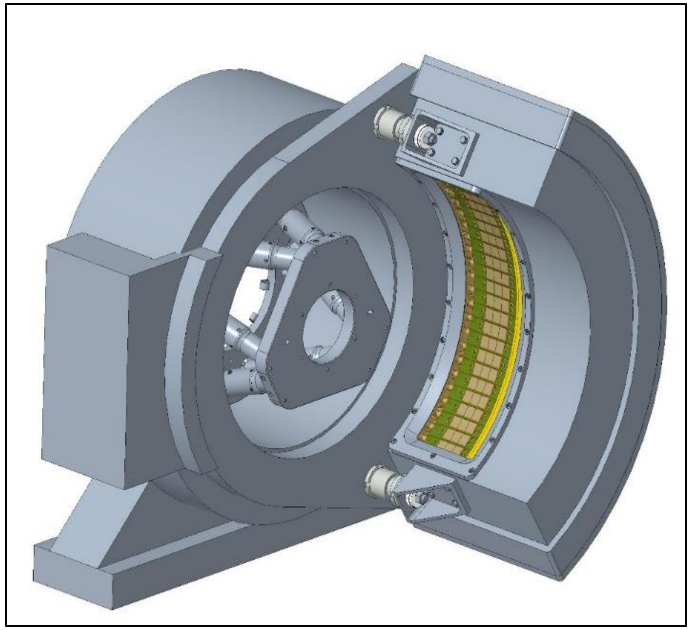
- AreaDetector widely used where support exists
  - Pilatus (<4)
  - Andor
  - PCO
  - Aravis
  - Phantom

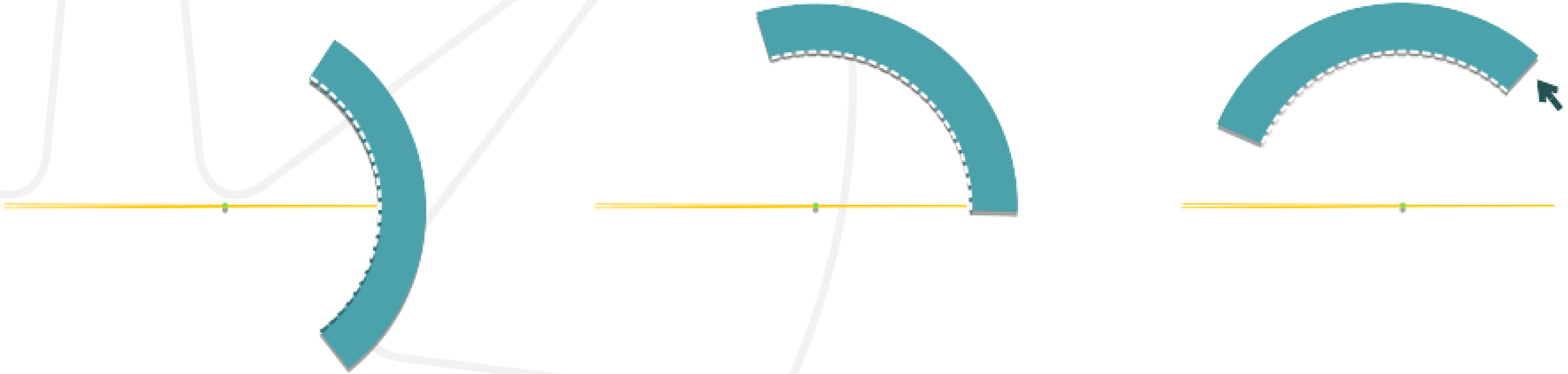
- Higher data rate detectors (>~10Gbit/s) use Odin
  - Arc XPDF
  - Eiger (Dectris)
  - Excalibur
  - Merlin*
  - Tristan
  - Xspress*

*Original AreaDetector still used where high data rates not needed

DLS Detector Group - Arc (XPDF)

- CdTe MediPix3 photon counting detector
- High resolution(s) | Qmax 35: (55 µm pixel size)
- Fast | Continuous collection @ ≥ 125 Hz
- Efficient | >85% @ 76.6 keV (1mm sensor)





DLS Detector Group - Tristan

- 160 Timepix3 chips from CERN Medipix3 collaboration
- Pixel size 55µm x 55µm
- 10,485,760 pixels – 23cm x 16cm
- 400Gbps Ethernet Readout
- 10M & 1M versions (so far)
- No Readout Dead Time – Continuous operation
- Data-driven Zero-suppressed Sparse readout
- Nexus / HDF5 dataset

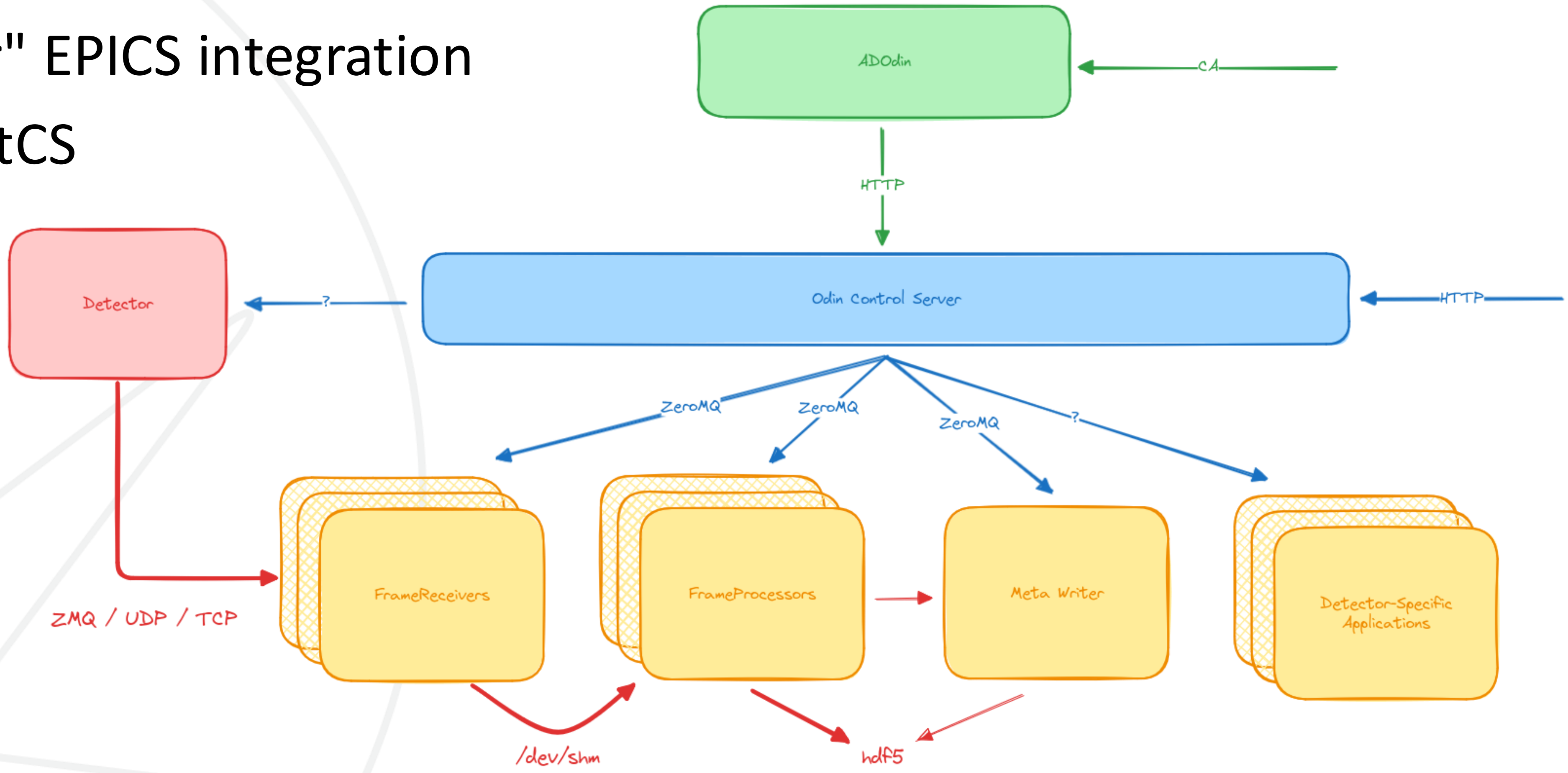
DLS Detector Group - Xspress IV

- Enhancement of Xspress3 detector readout system
- New spectroscopy DPP reduces detrimental effects of XTalk between pixels in monolithic detectors
- Significantly improves throughput and / or spectral quality of XAS scans (especially at high E) with existing workhorse HPGe detectors
- Switchable MCA or raw event mode to increase throughput

- STFC – DLS Collaboration

Current Odin Architecture

- C++ Data Readout
- Python Control
- "AreaDetector" EPICS integration
- ADOdin -> FastCS



Diamond-II Software Upgrade

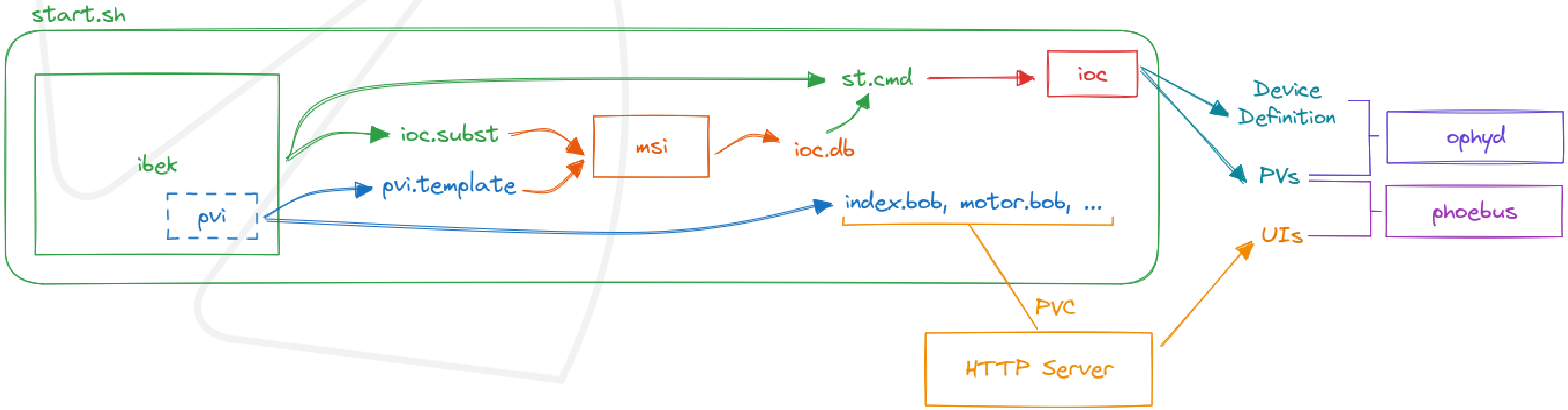
- **IOC Frameworks**
- Detector Readout

Diamon-II Software Upgrade - IOC Frameworks

- Make it easier to build and use IOCs
- Provide a high-level python interface for creating device drivers, compatible with modern tooling and deployment under Kubernetes
- Provide runtime introspection of IOCs for ophyd devices
- Generate standardised device-level UIs across C and Python IOCs

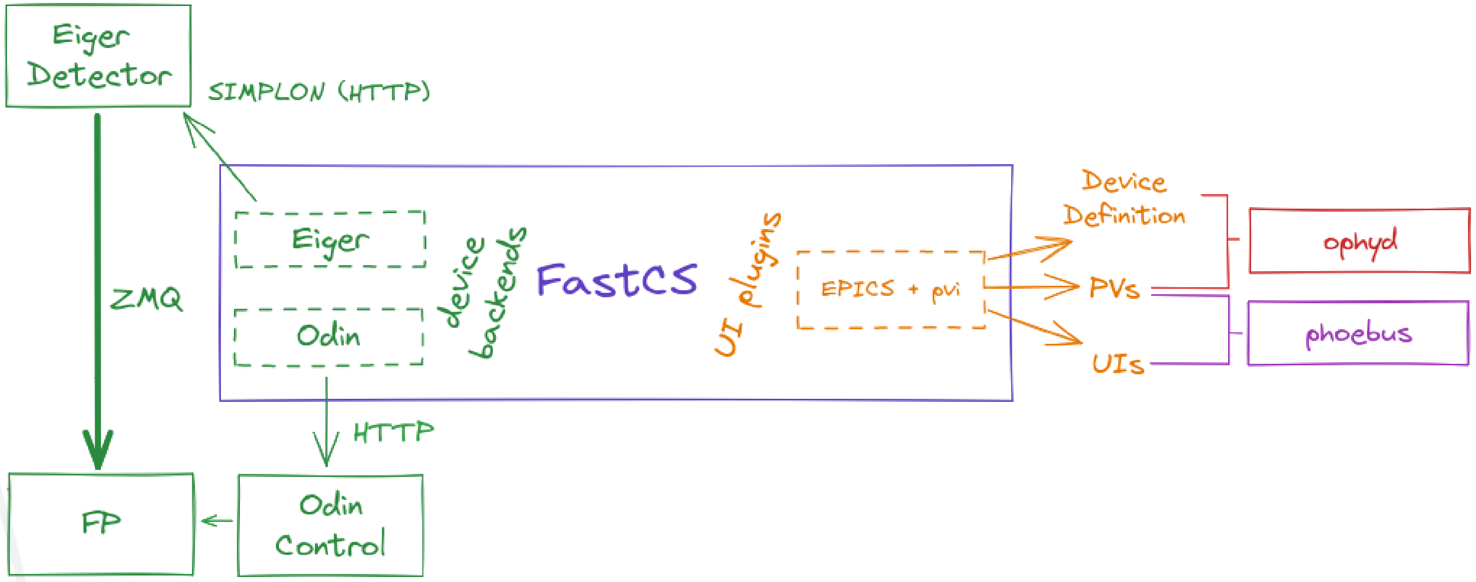
PVI – PV Interface

- A python library to provide an interface to the PVs in an IOC in the form of
  - A generated UI
  - An EPICS NTTable served over PVAccess
- Used in FastCS during IOC boot
- Used by ibek in epics-containers to generate UI and info tags in database before running IOC



FastCS – Fast Control System

- A python framework to create generic device support with a pluggable control system interface
- Implement a python library for a device with FastCS classes and load into a control system frontend, e.g. EPICS, Tango



- Uses PVI to generate Engineering UI of all parameters automatically
- Shared across teams in Beamline/Accelerator controls
- Simpler developer experience, gentler EPICS learning curve, modern standards and tooling

IOC Frameworks – Planned / Current Projects

- Eiger / Odin
  - Driver introspects APIs for available parameters
  - Removes need to maintain static definitions, e.g. EPICS database templates
  - New parameters exposed as PVs on IOC reboot

- PandA
  - Optional logic blocks in hardware introspected during IOC boot
  - Replace current custom pythonSoftIOC with a shared framework
  - Potential to extend collaboration with Tango-based facilities

- TwinCat
  - Introspection of EtherCAT I/O chain

- Alternative to StreamDevice with logic in python instead of EPICS database

FastCS Eiger & Odin Generated Phoebus UIs

Containerised Applications

- Moving towards containerising all IOCs and deploying to beamline K8S clusters via kubectl / ArgoCD

- epics-containers / ibek
  - Define containers built from upstream GitHub repos
  - Configure deployments in yaml
  - Wrappers around kubectl and argo cli for convenience

- Odin applications to be containerised in the same way for a consistent deployment process

Overview

- IOC Frameworks
- **Detector Readout**

Detector Readout – Requirements

- Support data readout for 100Gb detectors
  - Add support for writing data to the new storage architecture
  - Provide user configurable data reduction before data reaches persistent storage – circular buffering, frame/acquisition veto, ROI, downscaling, compression
  - Provide raw data in an ephemeral format when only the processed data is useful

- Enable analysis processes to access data at higher rates / with lower latency
  - Add more supported formats to detector readout framework - /dev/shm cache, S3 buckets, streams
  - Provide API to query where data for an acquisition is available

- Improve workflows for developing detector drivers and EPICS integration
  - Replace ADOdin with FastCS
  - Apply modern tooling and improve documentation to streamline developer experience

Near Future Detectors

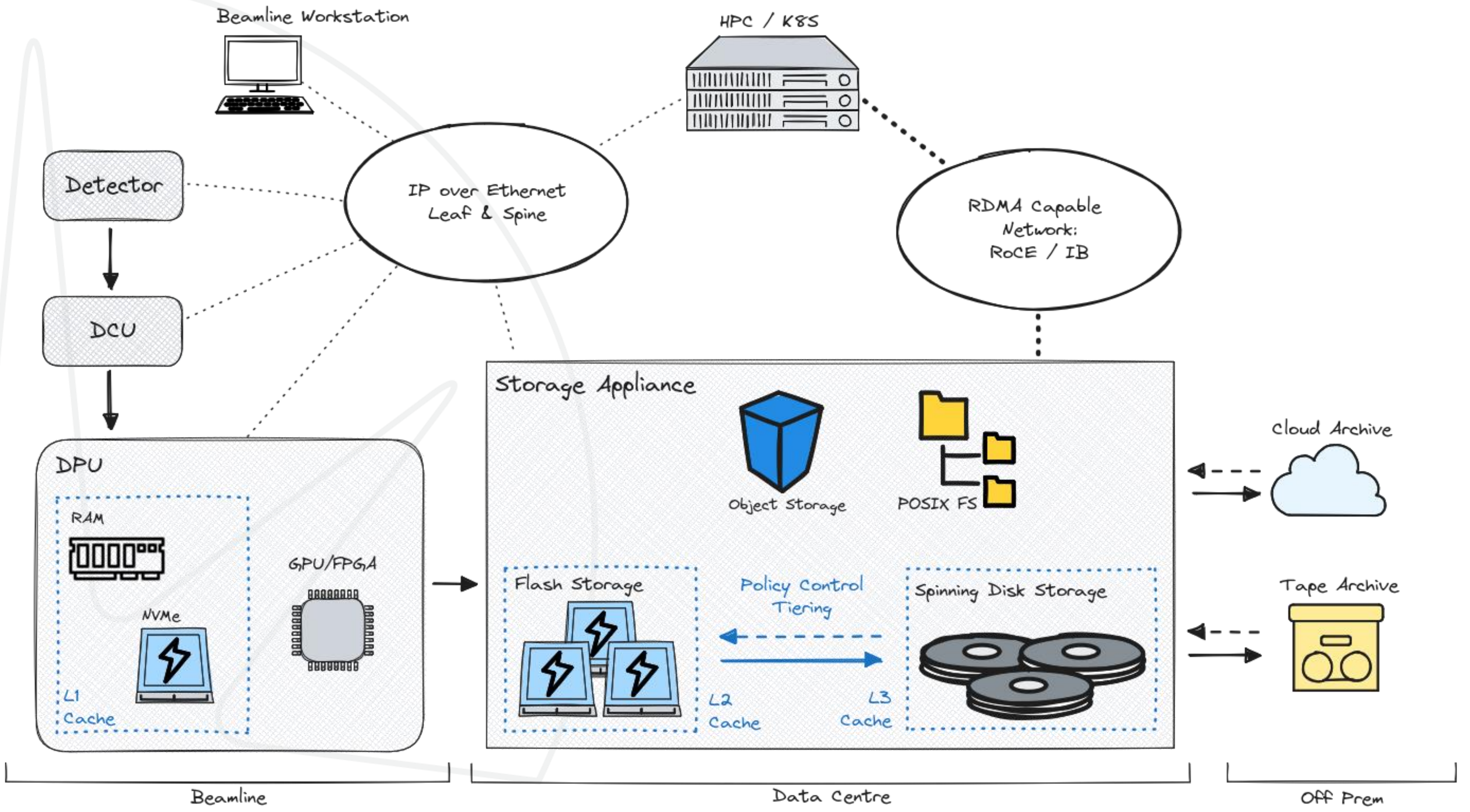
- Have 1M Jungfrau for initial testing. Positive results. 9M arriving in July
  - 2.2 kHz / ~300Gbit/s
  - Targeting jungfraujoch readout system for familiar Eiger-like API

- Xspress IV on new SWIFT beamline at 50kHz in event mode
  - Discussions around not writing raw events to HDF5 and streaming directly to cluster analysis processes to write results to file

- New K04 beamline to run Eiger 2XE at ~50% duty cycle
  - Fine tuning experiment orchestration / latency of control system and reliability of readout throughput
  - Storage to cope with ~ 4TB/hr data created

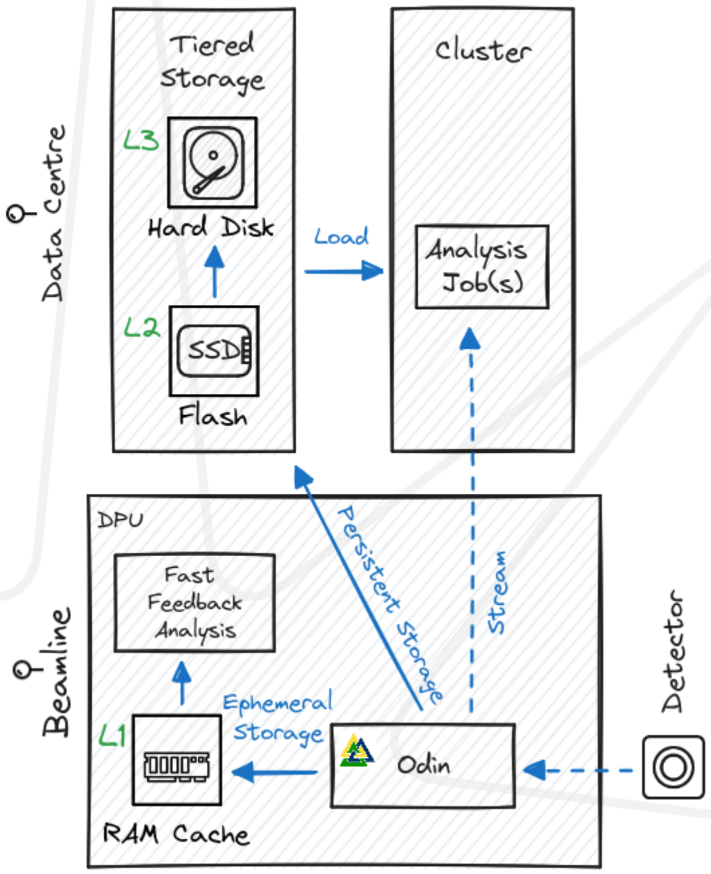
- Pilatus 4 possible, but nothing confirmed

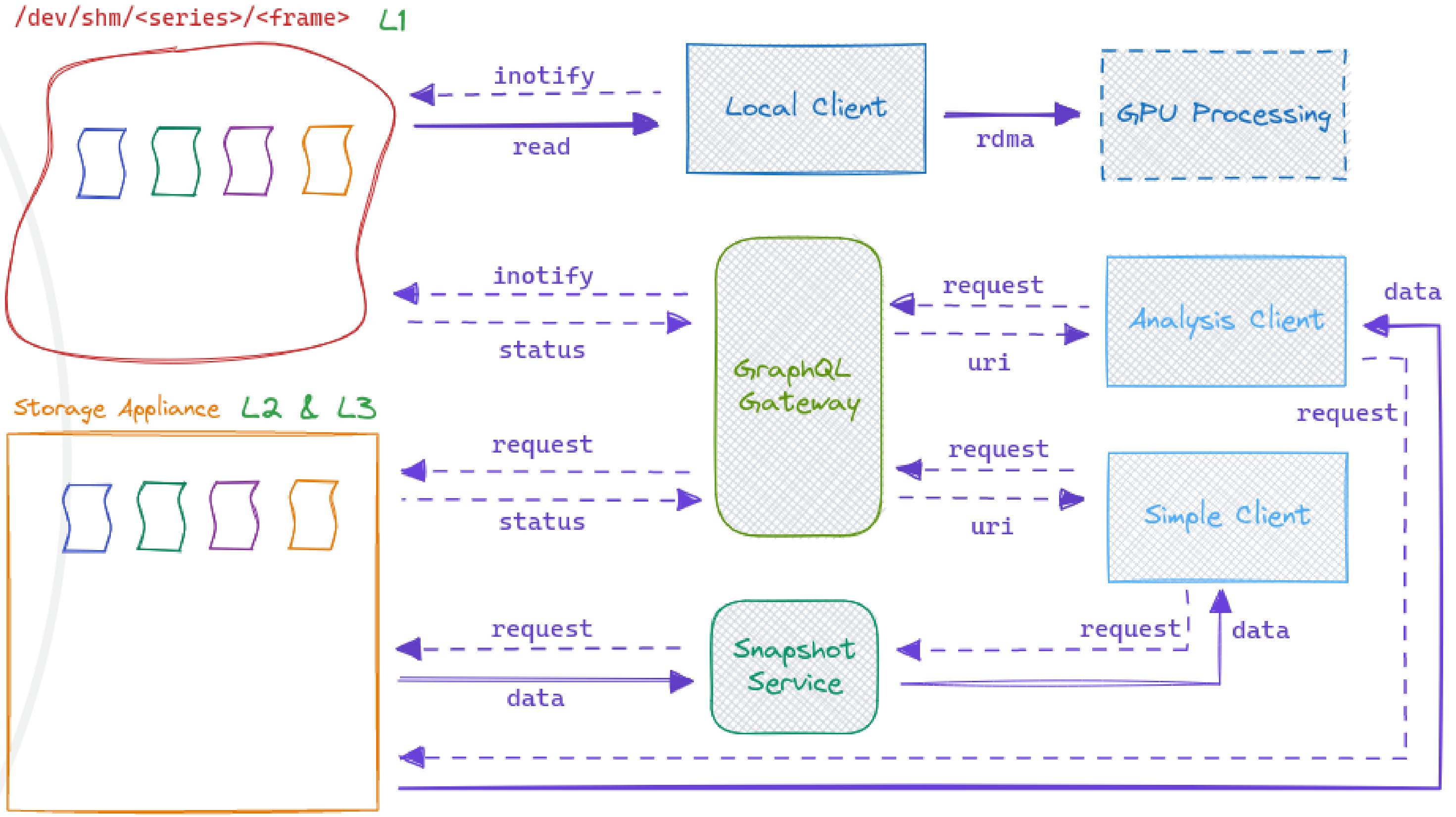
New Tiered Storage Architecture



Detector Readout – New Storage Architecture

- Detector data written to local /dev/shm cache (L1) and Object Store / Flash Storage (L2,3)
- Low latency processing and fast feedback via L1 cache in GPU/FPGA
- Compute-heavy processing in cluster via L2,3 storage appliance

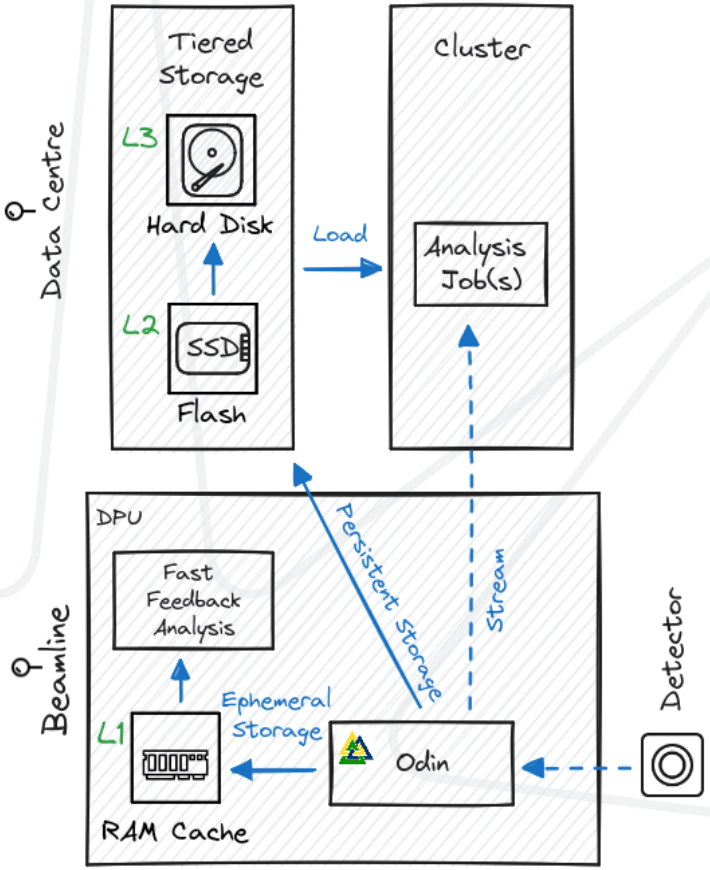


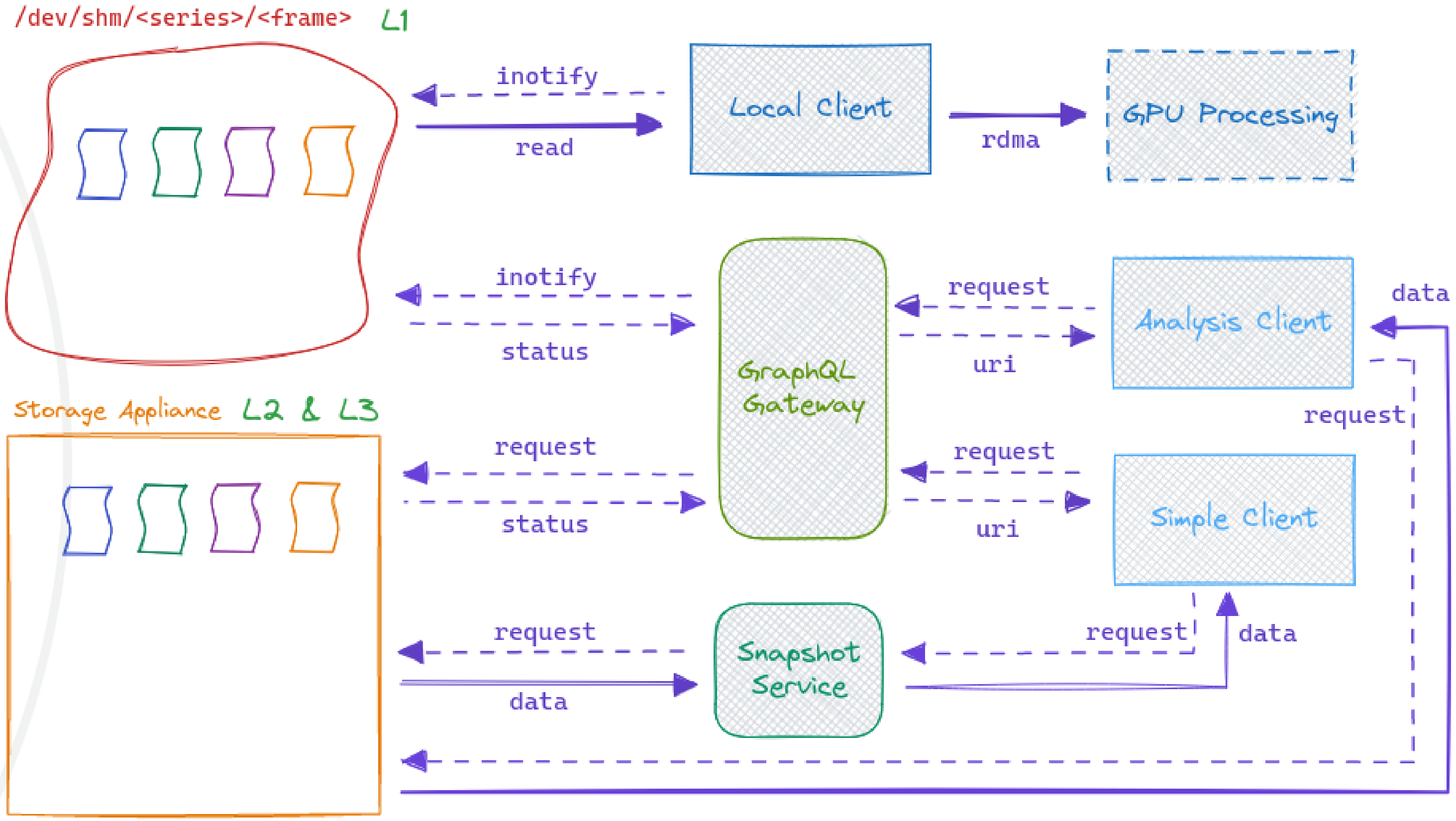


diamond

Detector Readout – New Storage Architecture

- Analysis clients could also connect directly to a full data stream served from the DPU
  - e.g. ZeroMQ, Kafka, Arrow
  - Lower latency to cluster
  - Useful for small, high-rate data - e.g. event-based detectors - or when only processed data useful

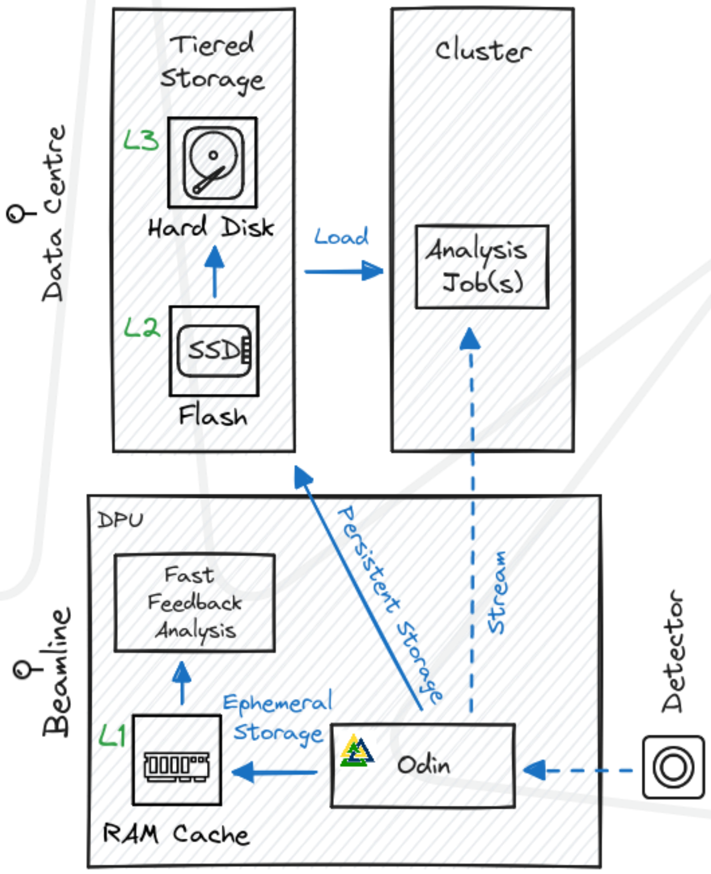


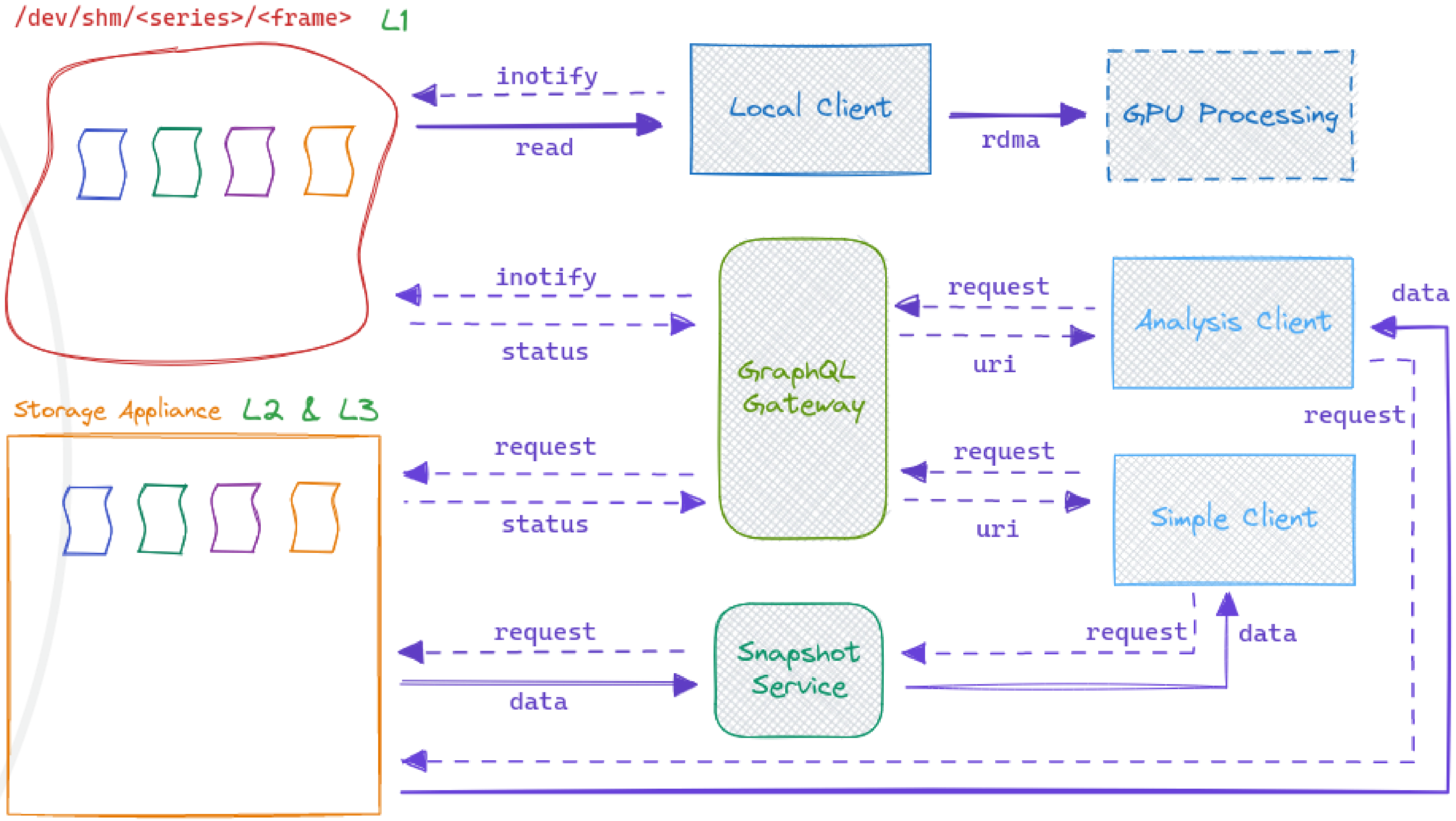


diamond

Detector Readout – New Storage Architecture

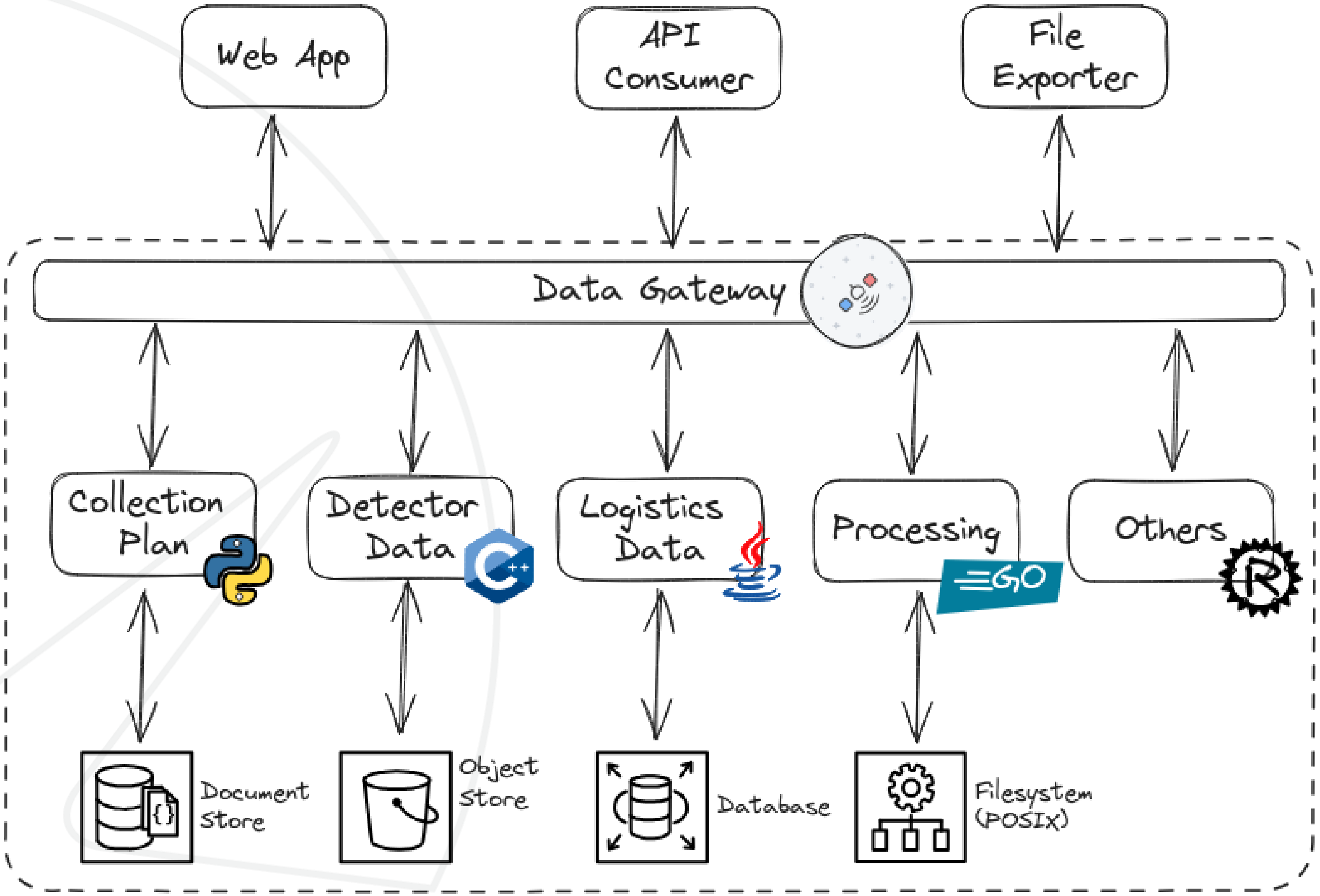
- Eiger /dev/shm local cache implemented on I03 to support GPU spot finding for fast feedback of alignment scans





diamond

GraphQL Gateway



Future

- Dynamix / XIDER / XIDyn
  - 100um pixel size
  - Possible 7.5M @ 200kHz
  - >1Tb/s

- Deprecate two-process architecture of FrameReceiver / FrameProcessor with/dev/shm memory buffer to use DPDK plugin in single application
  - Bypass OS kernel via poll-mode driver to push packets directly to application layer
  - Reduce multiple copies to get to user space - reduce latency and increase throughput
  - Saturate 100Gbit rate of network
  - Support for UDP / TCP – unsure on ZMQ

Acknowledgments

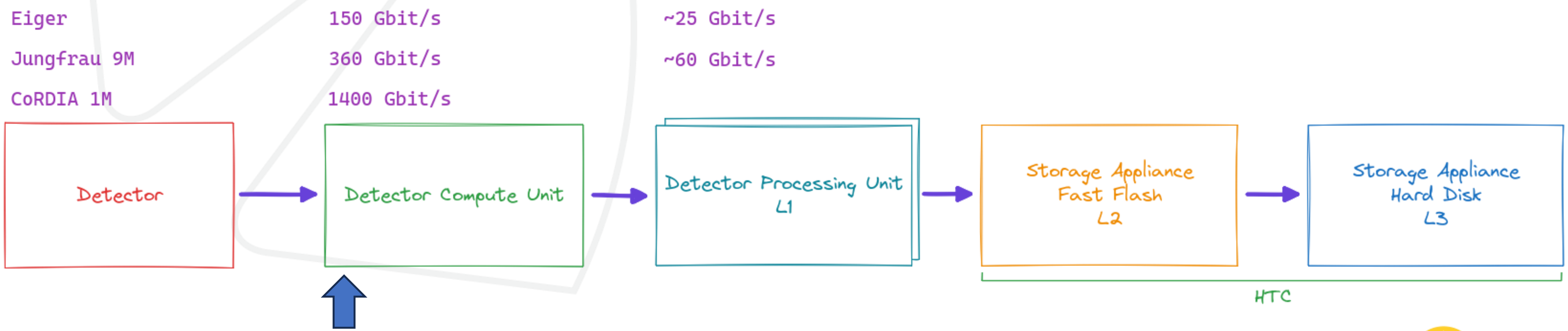
People who provided information / slides / diagrams for this talk

- DLS Detector Group: David Omar, John Matheson
- DLS Analysis Group: Graeme Winter, Garry O'Donnell
- STFC Application Engineering Group: Tim Nicholls

Detector Readout – Data Flow

Detector Compute Unit

- FPGA / CPU data stream processing
  - flatfield / mask / gain correction
  - timeslicing of frames
  - event buffering / rate reduction
  - compression
  - other lossless data reduction

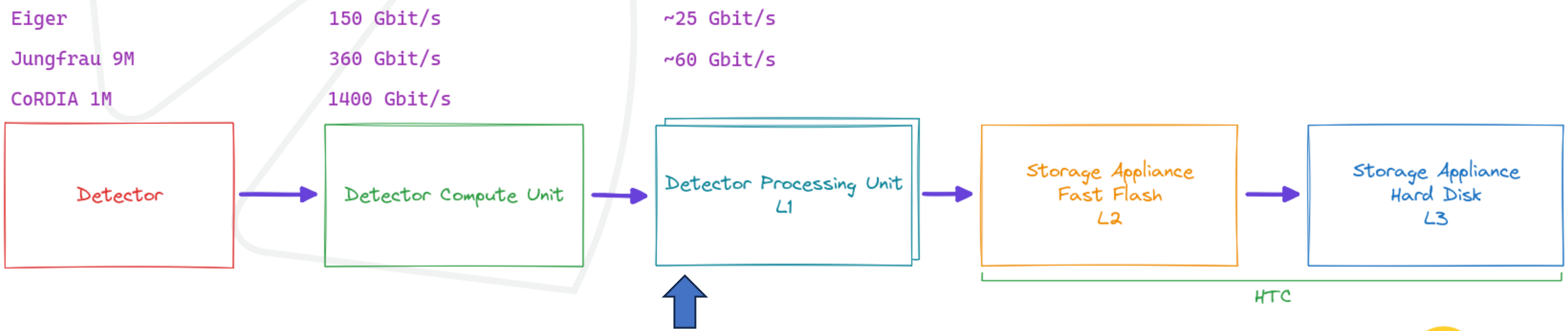


diamond

Detector Readout – Data Flow

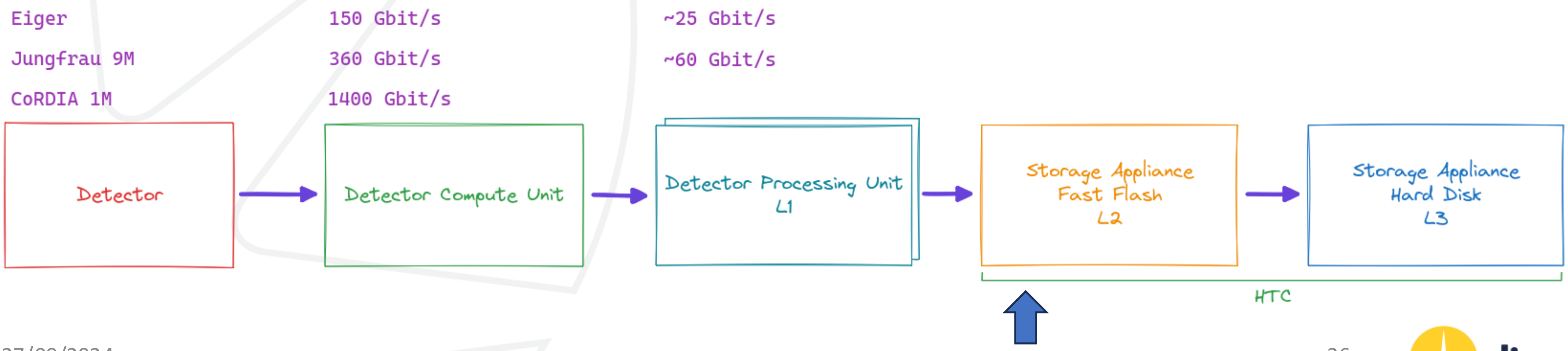
Detector Processing Unit

- CPU detector readout and control system integration
  - option for CPU / GPU / FPGA for
    - experiment control / fast feedback
    - or low latency data analysis
  - data quality reporting
  - circular buffering
  - frame / event level veto
  - write to disk and/or serve data over a stream



Storage Appliance – Fast Flash

- CPU / GPU scientific data analysis
  - likely final processed data for user
  - a few seconds of latency but during acquisition



diamond

Storage Appliance – Hard Disk

- Final state for archival
  - further / repeated data analysis
  - complete disconnect from acquisition