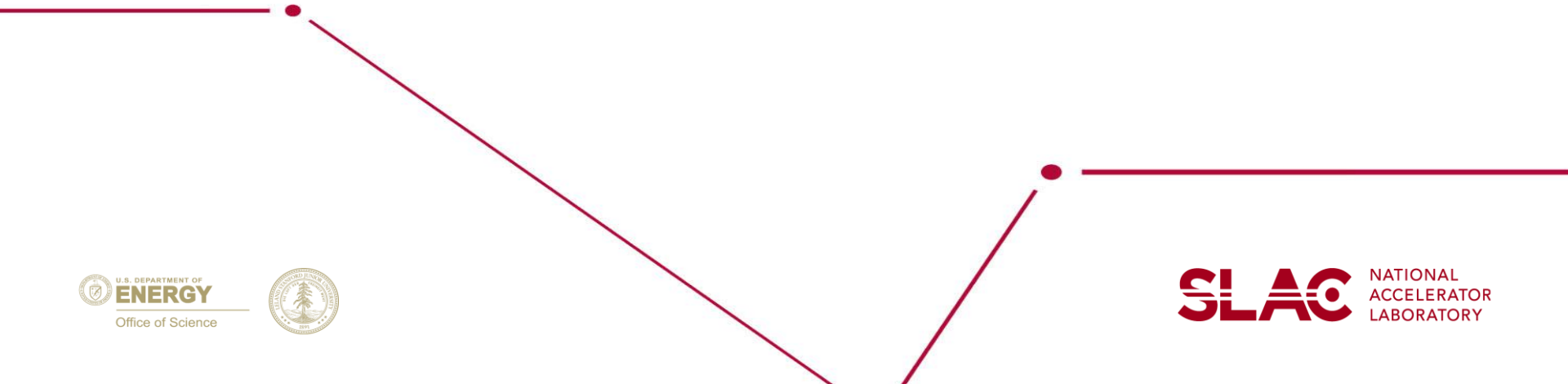
Building a Data System for LCLS-II

Jana Thayer, LCLS Data Systems

Workshop on New Concepts in Ultrafast Data Acquisition

April 10th-11th 2018, PSI



U.S. DEPARTMENT OF ENERGY
Office of Science

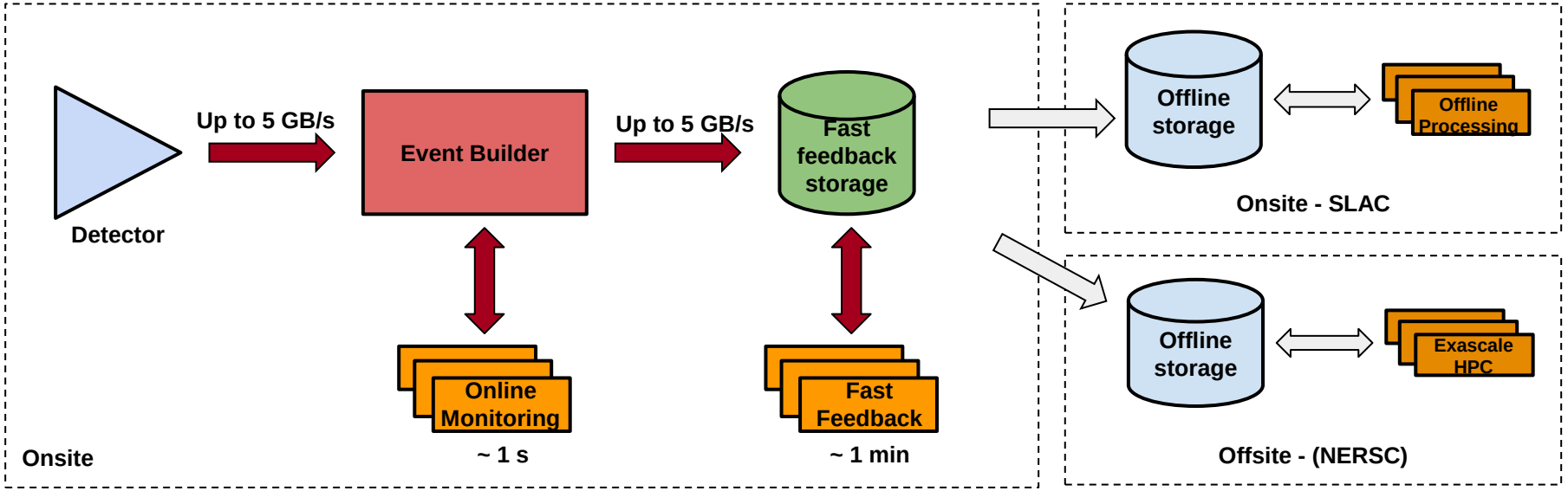
SLAC NATIONAL ACCELERATOR LABORATORY

Summary of key requirements and characteristics

1. **Fast feedback** is essential (seconds / minute timescale) to increase success rate of experiments
2. **24/7 availability**
3. **Throughput** between storage and processing is critical
4. **Storage** represents significant fraction of the overall system, both in cost and complexity
5. Speed and flexibility of the **development cycle** is critical

		Phase I	**Phase II**	**Phase III**
Parameter	LCLS-I **Present**	LCLS-II comm. **2020**	LCLS-II ops **2024**	LCLS-II HE **2028**
Ave throughput	1-2.5 GB/s	2.5-25 GB/s	5-200 GB/s	1296 GB/s
Peak throughput	5 GB/s	200 GB/s	200 GB/s	1.3 TB/s
Data cache storage	50 TB/hall	1 PB	3 PB	10 PB
Peak Processing (offline)	50 TFlops	1 PFlops	5 PFlops	>130 PFlops
Disk storage	6 PB	16 PB	36 PB	>100 PB

Today's LCLS Data System



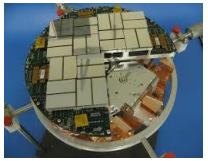
Today: 120 Hz acquisition, throughput up to 5 GB/s; fast feedback at 1 second and 1 minute timescales is crucial; majority of experiments are analyzed on SLAC offline processing farm

Today: no inline data reduction needed but fast feedback and data visualization found to be essential to efficiently execute experiments

LCLS-II Data System

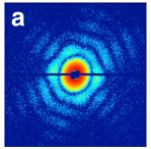
SLAC

Megapixel detector

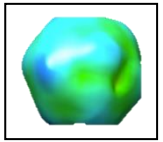


X-ray diffraction image

Intensity map from multiple pulses



Interpretation of system structure / dynamics



New for LCLS-II

Detector

Up to 1 TB/s

>10x reduction

Data Reduction Pipeline

100 GB/s

Fast feedback storage

Online Monitoring

~ 1 s

Fast Feedback

~minute s

Onsite

Offline storage

Offline Processing

Onsite - SLAC (standard experiments)

Improved for LCLS-II

Offline storage

Exascale HPC

Offsite - HIghest Demand Experiments (NERSC, LCF)

Data reduction mitigates storage, networking, and processing requirements

Three Different Levels of Data Processing

SLAC

Data Reduction Pipeline (DRP):

- Purpose: lossless or lossy reduction of data volume (~10x) before data reaches disk
- Multi-threaded C++ running on ~40 nodes (written by the core LCLS team)
- Small number of algorithms (~20) supports most experiments

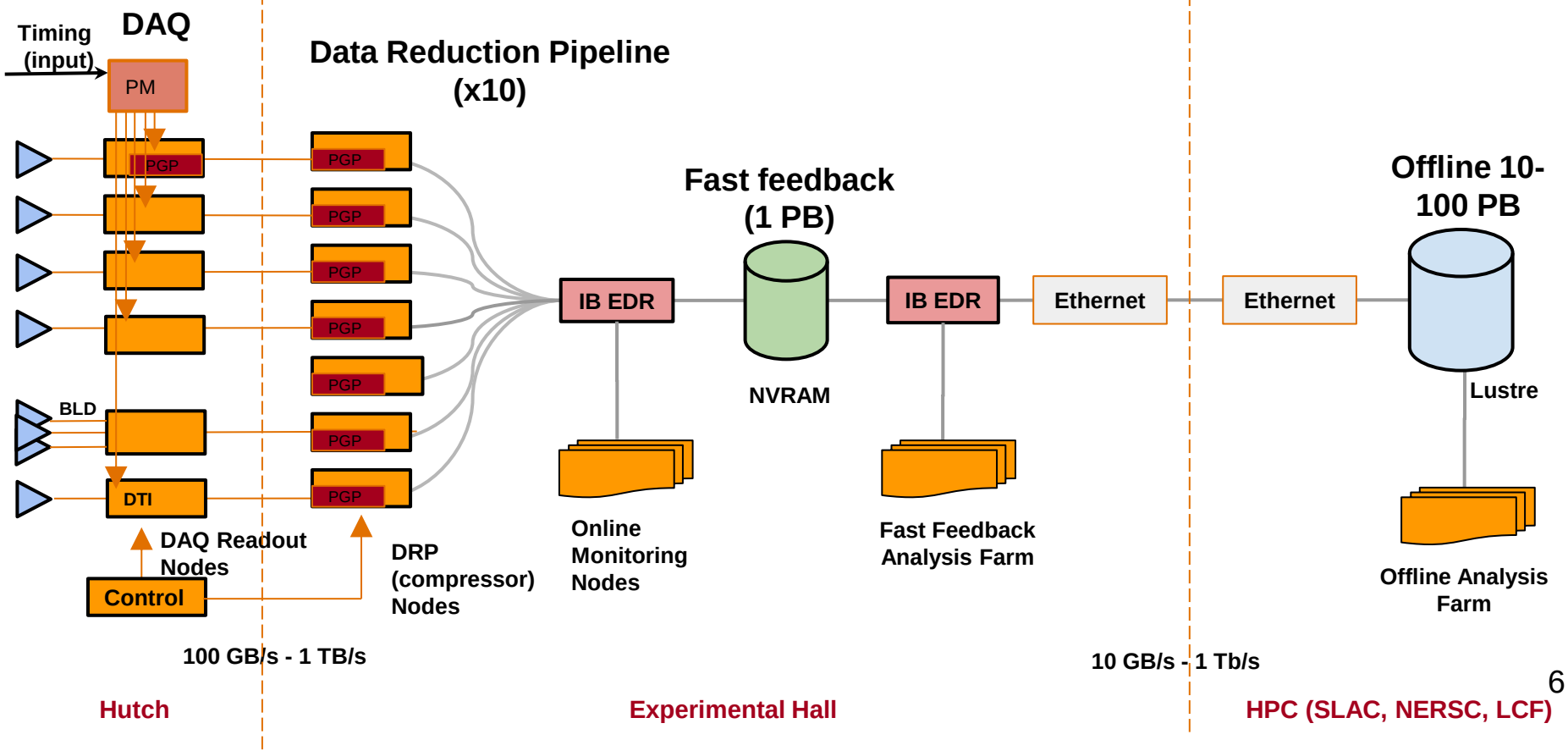
Fast FeedBack (FFB):

- Purpose: near real-time feedback to allow experiments to make operational decisions
- Runs on disk-based data (reserved for running experiment) with latency of ~1 minute with parallelized-python code (top level written by users) for quick development

Offline:

- Purpose: obtain final physics results
- Mostly parallelized-python code (top level written by users)

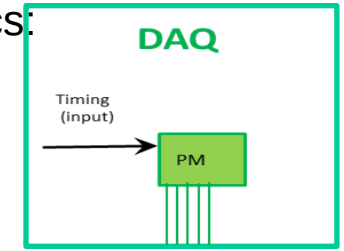
LCLS-II Data Flow

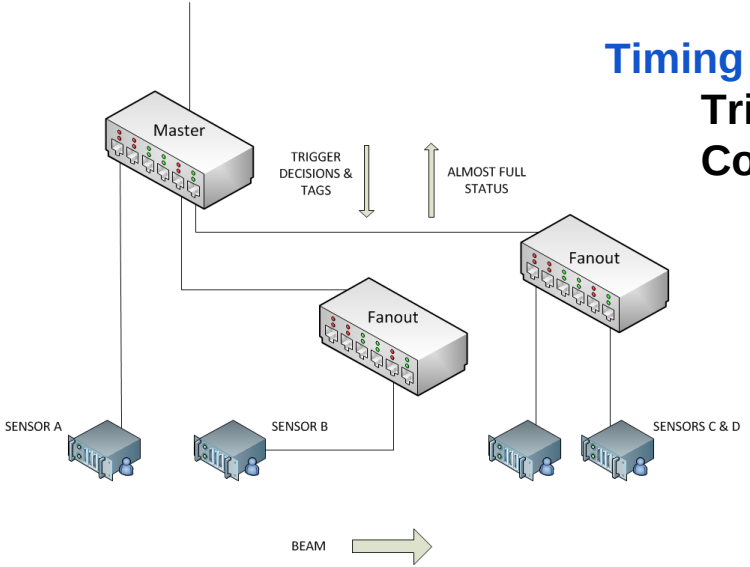


DAQ and Data Reduction Pipeline (DRP)

DAQ: LCLS-II Timing and Fast Control Distribution

LCLS-II Timing system delivers frames of fast control data over fiber optics:
Frame rate: 929kHz
Beam synchronous clock: 186MHz
Frame contents: PulseID, beam present, timing markers, control words sequenced by experiment request.





Timing Master appends commands to frame data:
Trigger decisions: exposure and readout control
Commands: configuration control and event handling

Distribution will fanout command data and fan-in feedback information.

Sensors can now participate in controlled deadtime.

Enables Deadtime and Fast Veto

DAQ: LCLS-II Timing and Sensor Readout

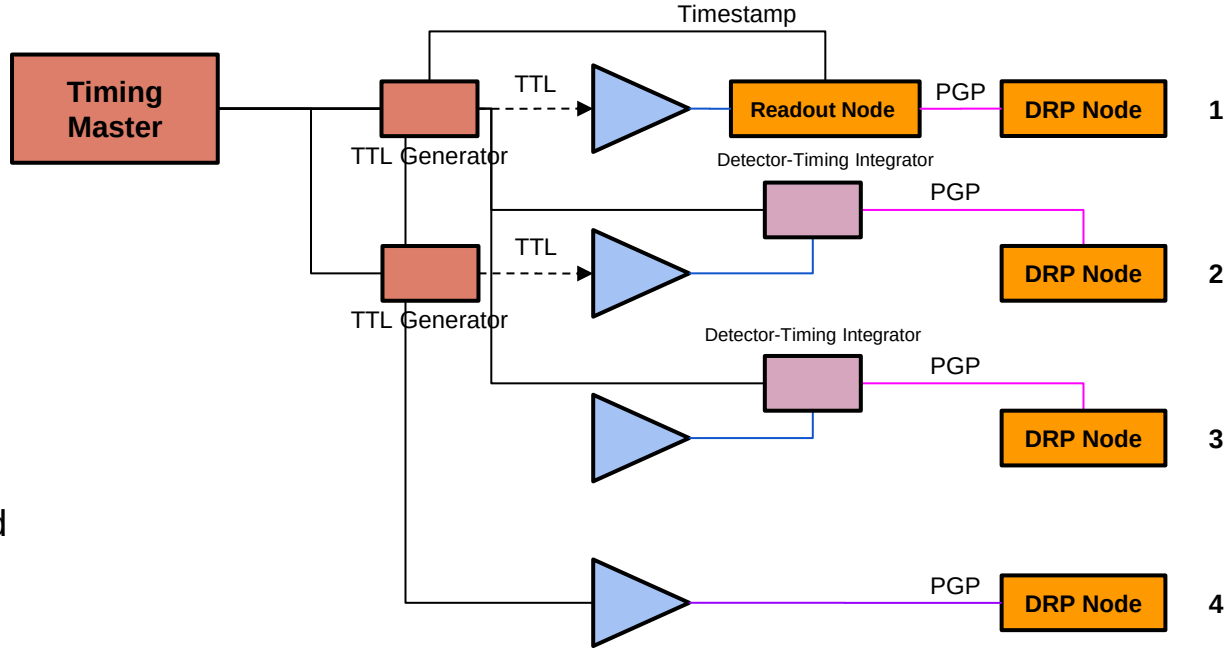
Four models of sensor integration:

1) *Commercial-I*
   Separation of trigger, data, and timestamp path (CPU required)

2) *Commercial-II*
   Separation of trigger, data, and timestamp path (CPU not req'd)

3) *Custom-I*
   Paths joined by an interface card

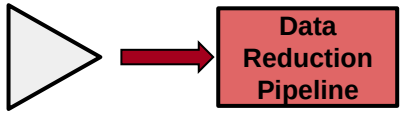
4) *Custom-II*
   Sensor receives timing directly



Attaching a timestamp to the data at the source (detector) offers a powerful advantage - the ability to online event build and monitor data

Design capable of integrating wide variety of sensor readout

Data Reduction Pipeline (DRP) Design Drivers



SLAC

Data Reduction Pipeline: reduce data volume before writing to disk

- **Reduce data in a way that does not affect the science result**
  - Reduce rate through **feature extraction**, **compression**, or **event veto** using flexible and configurable toolkit of data reduction techniques
  - Match the data reduction algorithm to the experimental technique
    - 100+ experiments per year, 3 - 4 experiments per 5 day period
    - 20+ experimental techniques identified, computing needs quantified
- **Keep up with data acquisition rate**
- Most experiments today already reduce their data volume offline: move this feature extraction and data reduction online

Throughput, not computing, drives the size of the data reduction pipeline

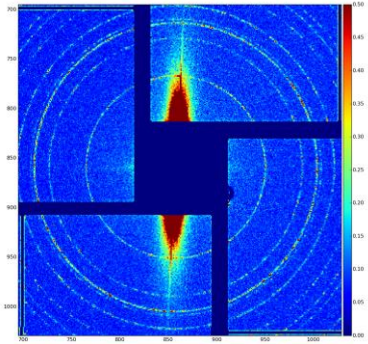
All current experiments reduce data volumes → Now do it in realtime

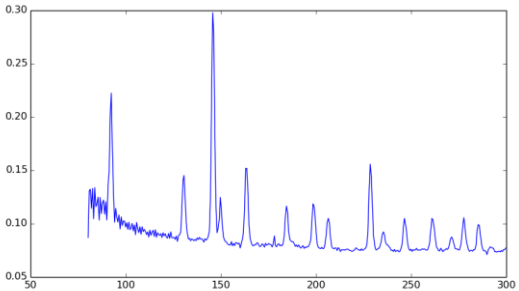
DRP: Algorithm toolkit

Went through all LCLS-II experiment types through 2026.

Identified **~10 data reduction categories**:

- Triggering (software veto on entire event)
- Accumulating
  - Includes angular integration averaging
- Binning
- Lossless compression
- Lossy compression (SZ algorithm from ANL)
- ROI
- Zero-suppression (software and firmware)
  - Includes peak finding
- Timetool calculation (firmware)



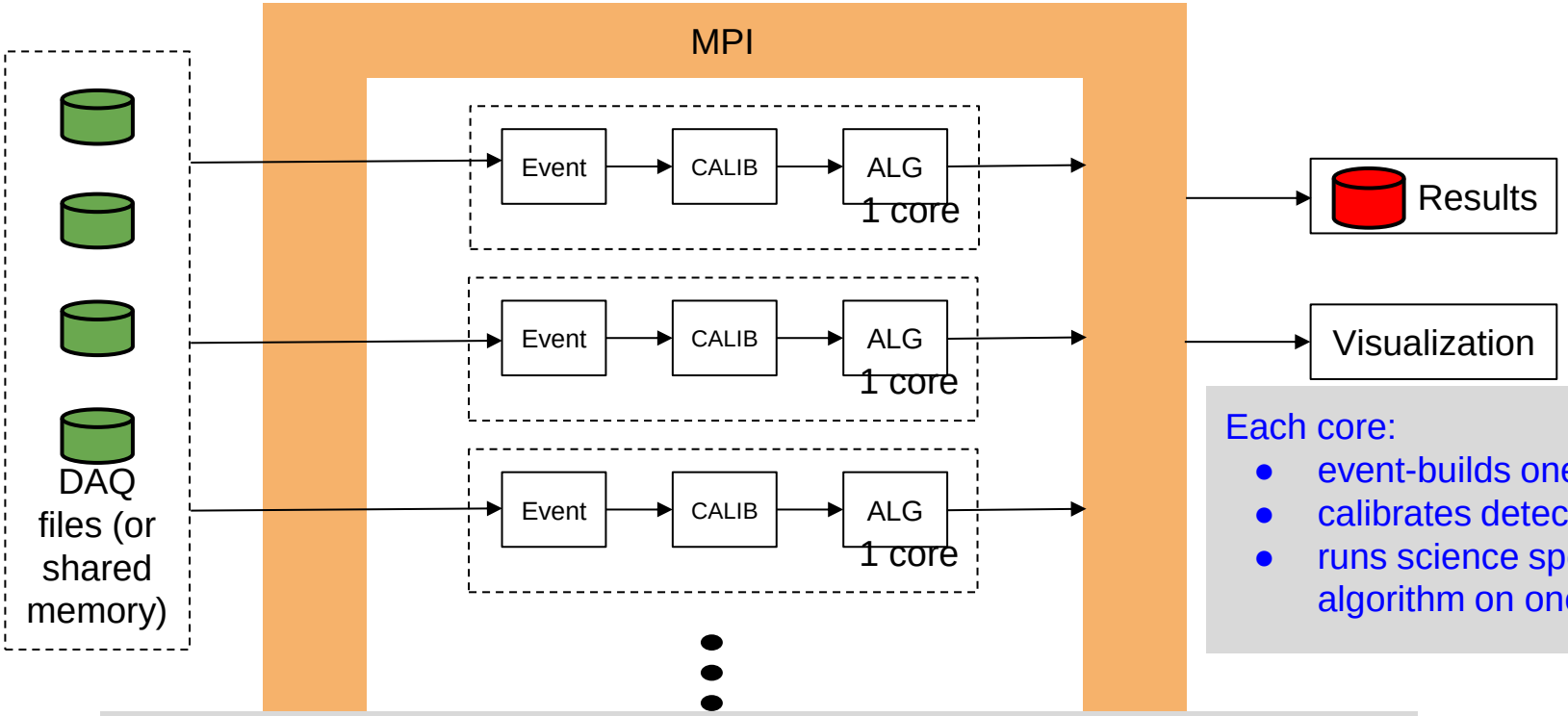


Implementation is on CPU (no GPU for L2S-I) and some FPGA for special cases like timetool and high speed digitizer.

DRP: Event Batching

- 1MHz event rates (interrupt rates) difficult on linux.
- **Batch events in groups** of definable size.
- Major examples:
  - New PGP Card interrupt generated only when it goes from empty to non-empty
  - Re-entrant Queues in DRP only take a semaphore when they go empty/full
  - Trigger/monitoring event-builder
  - Persistent storage: reading/writing data

DRP: Parallelization



Each core:

- event-builds one shot
- calibrates detector data
- runs science specific algorithm on one event

"Perfectly parallel" pattern: applications can be scaled, limited only by data distribution from filesystem or shared-memory (typically 20-100 cores)

DRP: Data Formats

SLAC

- In-memory representation (streaming): Self-describing XTC
  - Event Builder and online analysis read data from shared memory
  - No serialization/deserialization required for transport
  - Self-describing data using fundamental types
- File Data Format: HDF5
  - Full SWMR (Single Writer Multiple Reader), available summer 2018:
    - Natural management of variable-length data
    - Batching efficient writes of small data
    - Potential file corruption on crash
  - User-complexity-saving wish-list:
    - When file is opened, it is not readable with SWMR for “a while”
    - User must manage meta-data refreshes to not compromise performance. Increases complexity
    - Offline event-builder ideally done by HDF5 instead of user. Virtual Dataset cannot tolerate dropped data (requires regular pattern).
    - Additional software is needed to do HDF5 real-time copying.

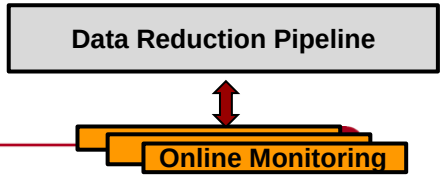
DRP: Lessons Learned

SLAC

- In-memory data format: XTC looks best
- Language: pure C++ is the right answer for performance
- Event-batching: necessary everywhere to avoid per-event overhead
- Trigger/monitoring event-builder: infiniband RDMA via libfabric works
- Algorithms:
  - Supporting tools: beam-center finding very nice but not perfect
  - SZ compression helps, but could use extra compression. CPU consumption is large: need a factor of 5 improvement in CPU usage.
  - Fast-thresholding veto looks good for crystallography/spi
- FPGA looks good for timetool and digitizer
- GPU looks not a big win vs CPU for common tasks
- Offline event-builder simplifies the DRP significantly

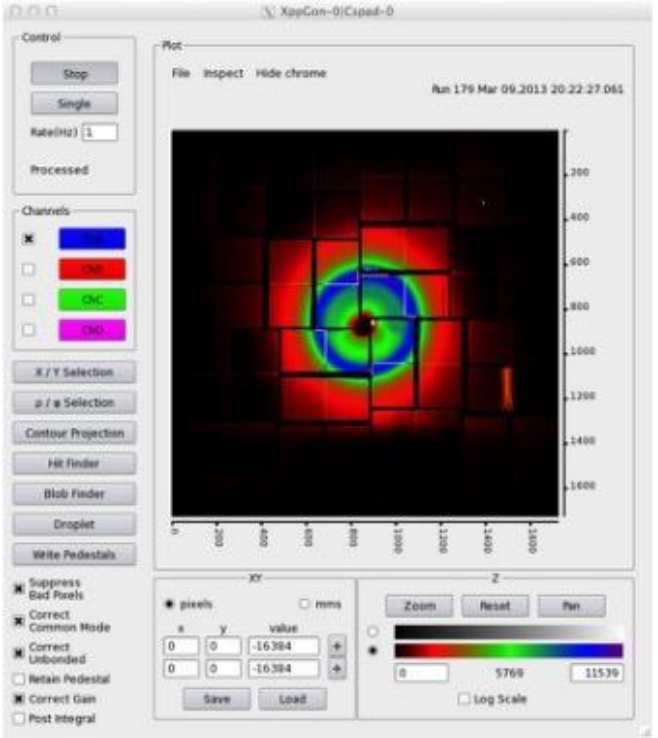
Online Monitoring

Online Monitoring - real time data visualization



Online Monitoring: display and analyze data on-the-fly

- Provides simple, real-time analysis of sampled data
  - Configured through a **graphical user interface**
  - No coding experience or preparation required
  - Analysis results used to
    - direct beamline operations, e.g. to optimize alignment of beam on sample
    - tune DRP parameters
- **Statistical subsample** of data read from memory
  - Builds selectable subsets of the data flowing through DRP
- Data and results are **transient**



Data visualization and simple analysis in < 1s crucial to beamline operations

Online Monitoring Design

SLAC

Small compute farm for live analysis

- latency < 1 second, data read from shared memory
- whole events, but only a fraction of event rate (event built)
- software preselection of monitored events
- distributed processing, collected results

Programmable or graphical analysis

- share code base with offline analysis
- reuse standard tools; leverage python eco-system
- simple user python code

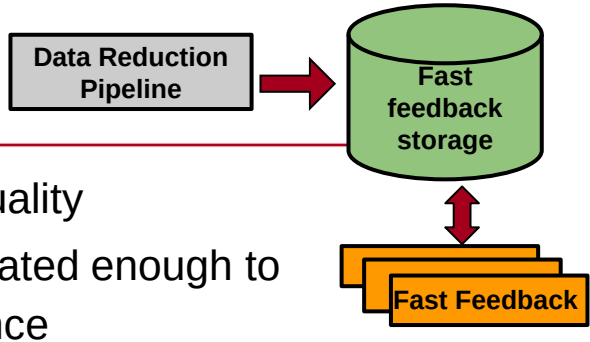
Feedback for automation

- hutch controls (bluesky)
- accelerator (EPICS PVaccess)

Based on LCLS experience / lessons

Fast Feedback

Fast Feedback - data quality monitoring



Fast Feedback (FFB): near real time feedback on science data quality

- Runs a simplified version of the full analysis: fast yet sophisticated enough to provide event-level information used to drive experiment science
  - **Dedicated processing** reserved for the running experiment
  - **Scripted** interface gives user **flexibility** to express any analysis
  - **Full statistics,** access to all event sources
  - Data and results are **persistent**
- Users obtain a measure of **scientific data quality** as data are acquired
  - Results are needed on the timescale of minutes, driven by the time needed to tweak and optimize experimental setup between data acquisition runs
  - **Validate** performance of DRP
- Scientific steering requires near real time latency (~1 min)

FFB critical for measuring science data quality in near real time
Required to maximize experiment efficiency

Fast Feedback (FFB) System Overview

- Fast Feedback system consists of an SSD-based data cache and a processing farm carved from the Offline processing farm.
- Data cache storage
  - Data Reduction Pipeline (DRP) writes event data to the FFB
  - Exclusively for the active experiments; lifetime of data ~12 hours
- Many files per run will be written, one file per DRP node
- Access to the FFB is a posix file system (Lustre) mounted on the clients
- Data movers will read the data from the FFB and transfer them to:
  - Offline storage
  - Remote location for real time analysis (e.g., NERSC)
- Data will be read by users while being written by DRP
  - Write: 1x DRP data rate
  - Read: 3-5x the DRP data rate (users and data transfer)

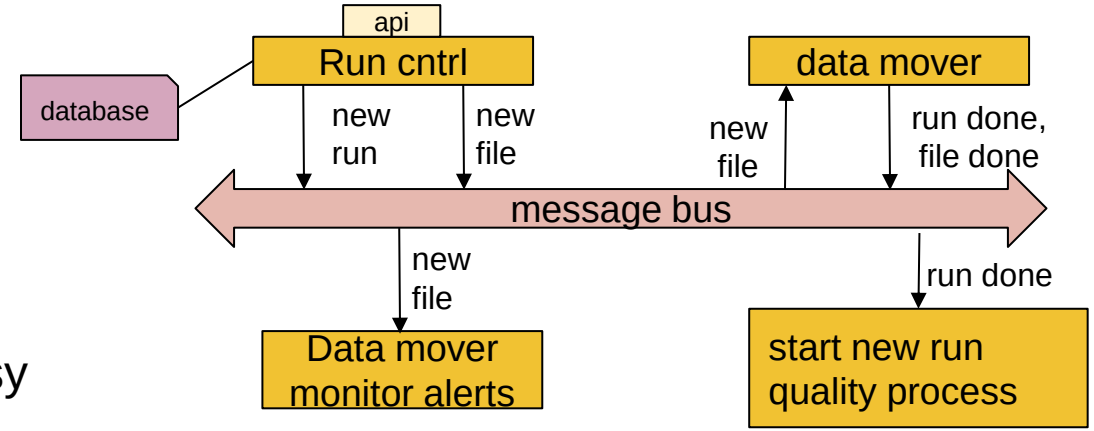
Data Management

Data Management System Responsibilities

- Provide storage resources with different requirements:
  - storage for fast feedback during data taking
  - analysis-storage for science data analysis
  - User storage supporting the analysis activities
  - tape storage for long term archival
- Automatic transfers of the science data files to the different storage within SLAC or remote sites
- Tracking and management of the science data files: cataloging files, distribution (local and remote), lifetime on disk
- Experiment and experiment metadata management
- Automatic data processing (workflows, automatic batch processing)
- Monitoring and log file aggregation
- The user facing site of the data management tools should look and behave the same for LCLS-II and LCLS-I experiments

Data Management: Integration of Services

- The current system use a database to integrate all services
  - Changes to DB could affect many services
  - Lots of polling the DB for new entries or updates
  - Extending requires new tables or extending tables
- Switch to an service/messaging architecture
  - There are many independent service. A service exposes an API that other applications can use (e.g.: get new run number).
  - Services can publish and subscribe to messages
  - Loose coupling of services
  - Adding new services is easy



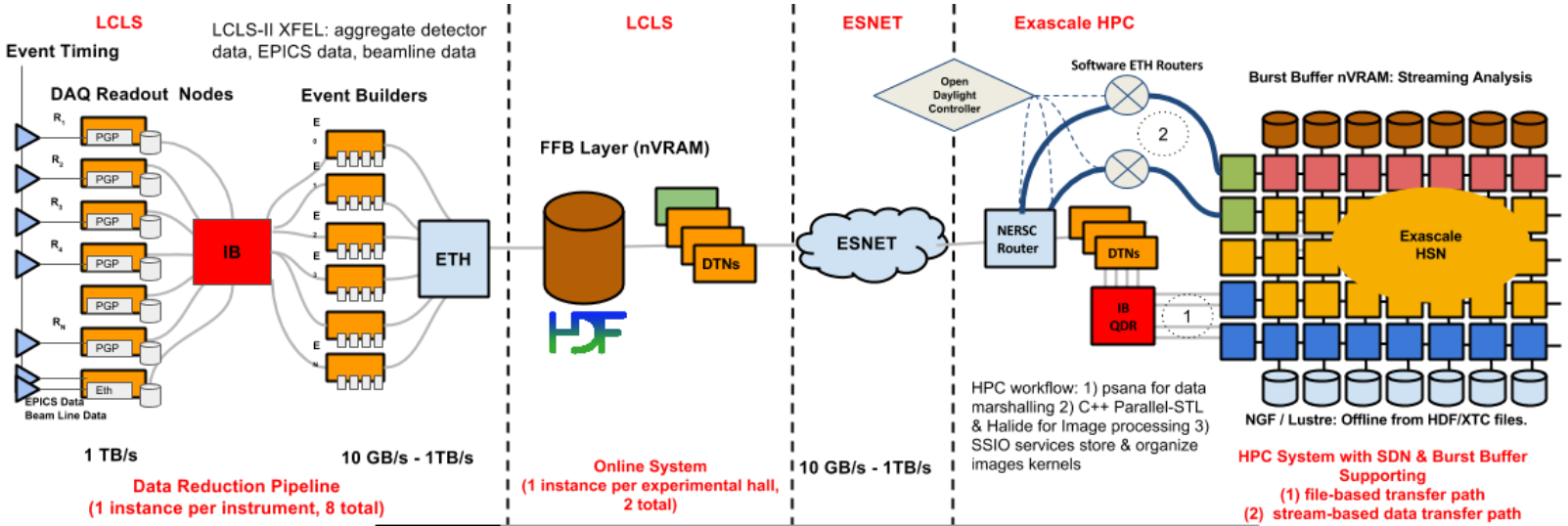
Offline

Strategy: Dedicated, local offline system complemented by DOE High End Computing (HEC) Facilities

SLAC

LCLS-II will require:

- **Dedicated, local** capabilities
  - **Data Reduction Pipeline**: Data compression, feature extraction, real time analysis
  - **Science Data Facility**: Storage and analysis for standard experiments, fast feedback analysis for all experiments
- Access to **HEC Facilities** for highest demand experiments (exascale) allows users to stream science data on-the-fly from LCLS beamlines to HEC vis ESnet; data management handled transparently



Summary

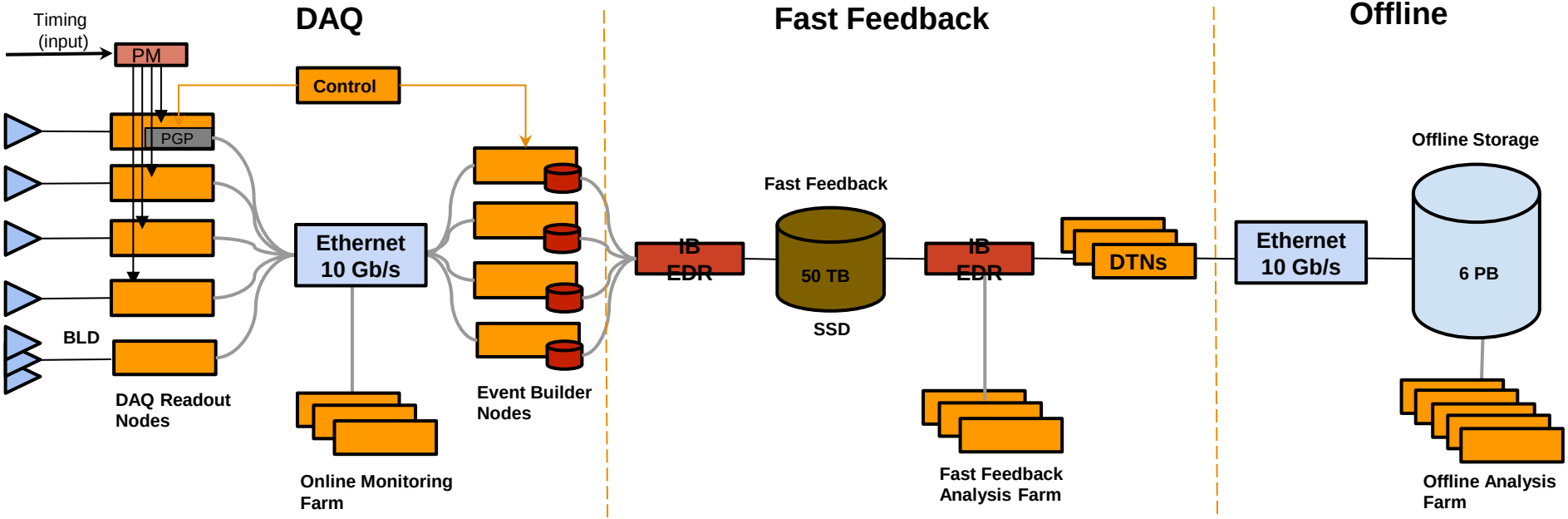
SLAC

- LCLS-II represents a significant data challenge in data **throughput**, **storage, network**, and **processing**
- **Data Reduction Pipeline** brings data throughput and storage needs to a manageable level
- LCLS-II Data System provides **analysis and visualization tools** with latencies that allow for direct, real-time tuning of the experiment as well as near real-time assessment of the scientific data quality
- **Algorithms** that run in the Data Reduction Pipeline and Fast Feedback layer for 20+ experimental techniques have been identified and their **performance quantified**

The LCLS-II Data System is scalable and flexible and addresses LCLS facility and user requirements

Backup Slides

LCLS Data Flow - Today



Hutch

1 - 5 GB/s
120 Hz

Experimental Hall

1 - 5 GB/s
1 - 10 TB per 12 hour shift

SLAC Bldg 50

1 - 50 TB per experiment
2.5 PB/year

DRP: Challenging Algorithms

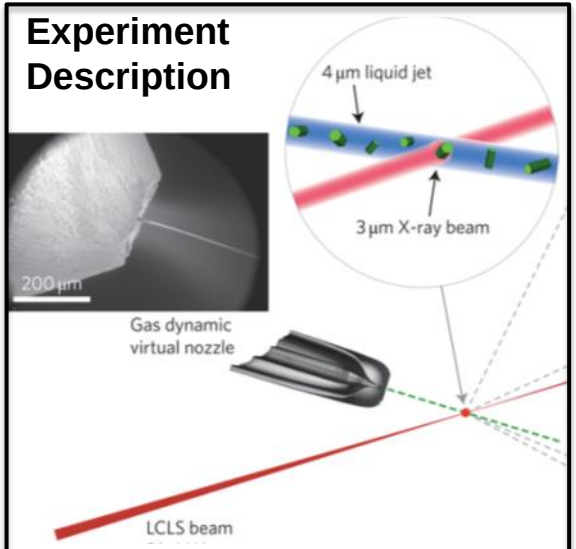
SLAC

- **XPCS** (X-ray Photon Correlation Spectroscopy)
  - Every event is a hit, photons are (often) dense
  - Either lossless compression, SZ compression or only saving speckles
  - Need to enumerate various cases more carefully (hard/soft x-ray, detector distance, bragg-spot/diffuse…)
- **FXS** (Fluctuation X-Ray Scattering: high concentration limit of SPI)
  - With good detector corrections and beam-center knowledge, believe we can compute angular correlations and sum resulting images
  - Working with CAMERA on this
- **TES** (Transition Edge Sensor) Detector
  - cross talk correction is computationally intensive, may need to be done in firmware, but not clear space is available in FPGA
  - event time-overlap complicates separation of data into events
- **SFX** (Serial Femtosecond Crystallography) in the unlikely multi-hit case

LCLS-II Data System Architecture: Single Particle Imaging Example

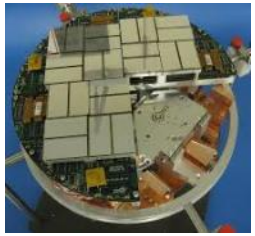
SLAC

Experiment Description



- Individual particles are injected into the focused LCLS pulses
- Scattering patterns are collected on a pulse-by-pulse basis
- Particle concentration dictates "hit" rate

Multi-megapixel detector



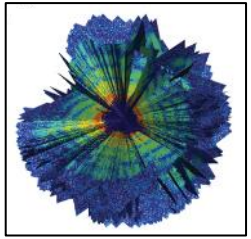
8 kHz in 2024 (4 MP)
40 kHz in 2027 (16 MP)

60 GB/s
1 TB/s

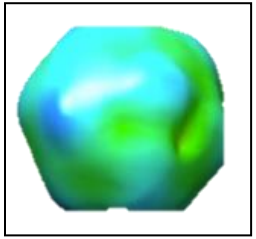
Coherent scattering image

6 GB/s
100 GB/s

Intensity map from multiple pulses



Interpretation of system structure / dynamics



Data Reduction
- Remove"no hits"
- >10x reduction

3 TFlops
16 TFlops

Data Analysis
- Orient patterns
- Average
- 3D intensity map
- Reconstruction

270 PFlops
1340 PFlops

Data reduction mitigates storage, networking, and processing requirements

LCLS-II and ATLAS: Similar but very different

SLAC

	LCLS-II 2022	LCLS-II 2026+	ATLAS Today	ATLAS 2026+
Wanted fraction of collisions	0.01 to 1.0	0.01 to 1.0	$< 10^{-6}$	$< 10^{-5}$
Typical experiment duration (same data-taking conditions)	3 days	3 days	3 years	3 years
24x7 availability of offline computing	Essential	Essential	Desirable	Desirable
Required turnround for data-quality checks	Seconds to minutes	Seconds to minutes	Hours to days	Hours to days
Raw digital data rate	200 GB/s	300+ GB/s	160 GB/s	1,000 GB/s
Zero-and-Junk-suppressed rate	10 GB/s	30+ GB/s	1.5 GB/s	20 GB/s
Storage need dominated by	Mainly raw data		Mainly simulated and derived data	
Role of Simulation	Growing in science analysis Growing in experiment design		Vital in physics analysis Vital in experiment design	
Analysis, Simulation and Workflow Software development community	Individuals (in the past) → Organized effort		~100 organized collaborators (mainly research physicists)	

LCLS-II data volume similar to ATLAS

Design Drivers: Self-Describing In-memory data format

Format	World Standard	Performant Access from C++/Python	Corresponding Robust File Format	No serialization, deserialization for transport	Well supported
EPICS	✔	✔	✘	✘	✔
Apache Arrow	✔	✔	✘	✔	✔
Protobufs	✔	✔	✘	✘	✔
Flatbufs	✔	✘	✘	✔	✘ (python)
XTC	✘	✔	✔	✔	✔

DRP: Parallelization

SLAC

- Hand out fseek-offsets to different cores
- Used this pattern to run on 30,000 cores at NERSC using MPI in 2017
- Also supporting Stanford “Legion” parallelization, which is expected to behave better than MPI at the ExaScale (e.g. fault tolerance, startup time, efficiency, GPU usage)
- Some extra overhead to support both parallelization schemes

Small-ish change (for performance):

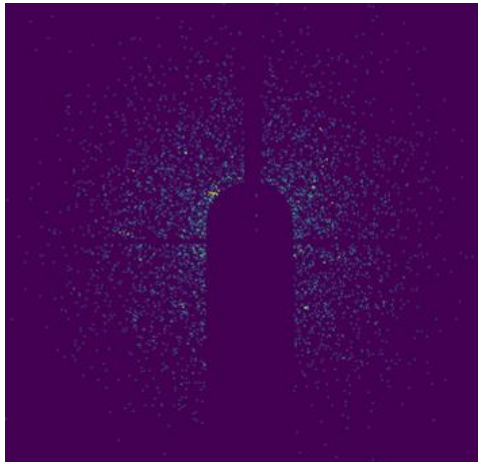
- Current model: each core fetches “small data” to decide whether or not to fetch large data using fseek-offsets
- Future model: master core fetches all small data for improved efficiency. Filters, then distributes fseek-offsets of interesting large-data events to slave cores.
- Requires interface change for separate “filtering” process

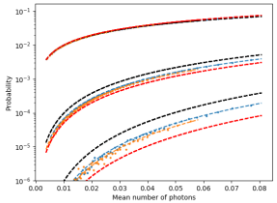
Design Drivers: Supporting Tools

- Beam center finding
- Detector calibration
  - Psana detector corrections:
    - Pedestals
    - Bad pixel masks
    - Common-mode noise corrections
    - Per-pixel gain
  - Have automated deployment of more corrections (e.g. common-mode parameters for pnCCD)
  - Used in most LCLS analyses for many detectors (Epix, CsPad, Jungfrau, pnCCD) so confident that our automated determination of these is robust.
  - Geometry is a significant problem: need more standardization of techniques, and automated tools for computing multi-detector geometry
- Critical but not started yet: Graphical control of DRP using real-time graphical monitoring tools (AMI)

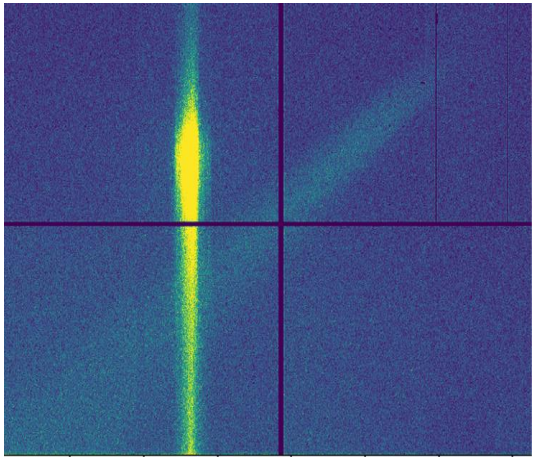
Common FFB/Offline Example: Photon finding

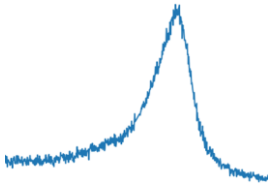
X-ray Photon Correlation Spectroscopy (XPCS)





X-ray Emission Spectroscopy



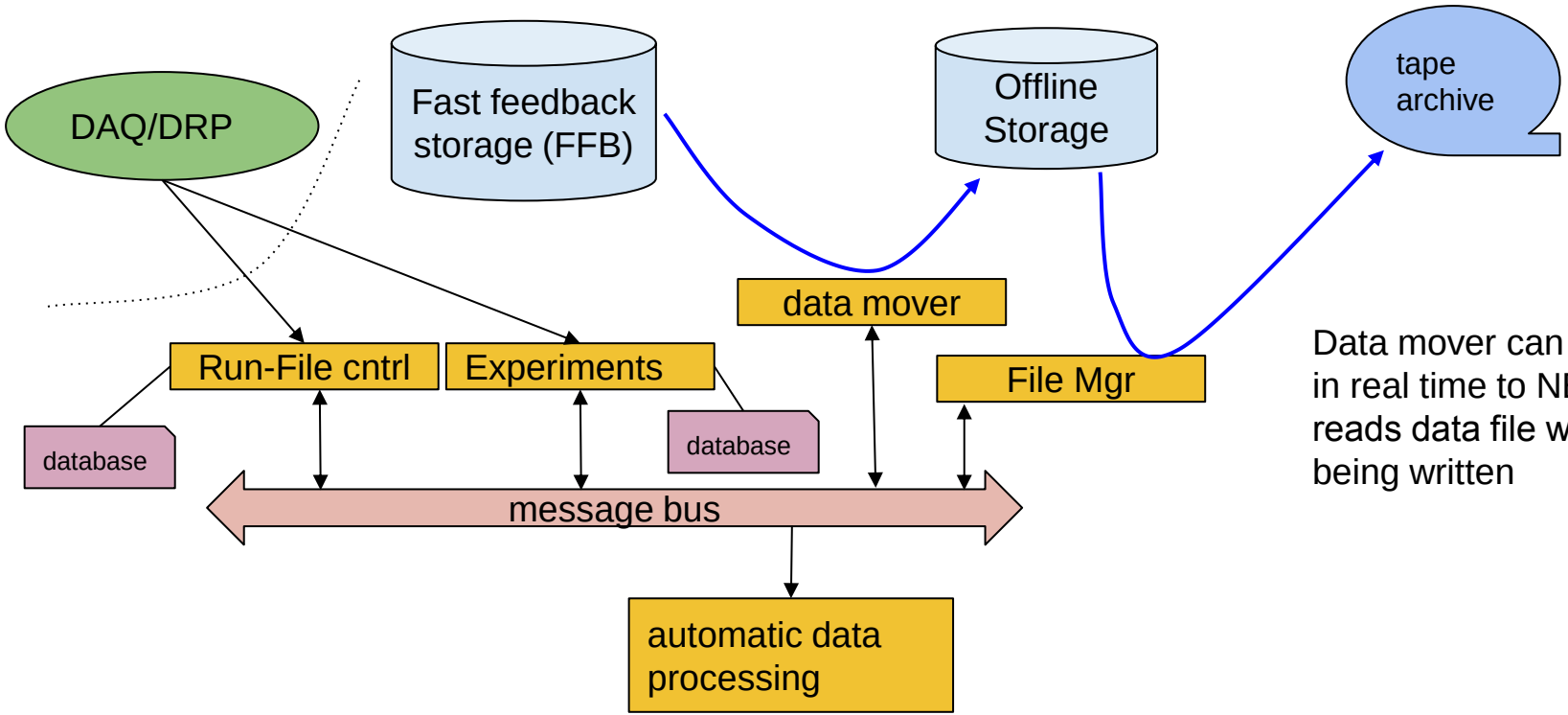


Reconstructing photon hits on image detector is important algorithm for many experiments

- 2 Threshold droplet algorithm
- 50 ms processing time for 1 MPix Camera (including detector corrections)
- 70 TFlops for 0.5 MPix @ 40kHz

Example of an algorithm that runs on Fast Feedback Layer

Data Management System Overview



Data mover can stream data in real time to NERSC; reads data file while it's being written