ExaFEL: Achieving Real-Time XFEL Data Analysis using Exascale Hardware

(in alphabetic order)

**NERSC:** Katie Antypas, Bill Arndt, Debbie Bard, Wahid Bhimji, Shane Canon, Bjoern Enders, Lisa Gerhardt, Laurie Stephey, Rollin Thomas, Felix Wittwer

**LBNL:** Aaron Brewster, Asmit Bhowmick, Anna Giannakou, Daniel Paley, Lavanya Ramakrishnan, Nick Sauter, Elyse Schriber

**SLAC:** Wilko Kroeger, Derek Mendez, Amedeo Perazzo, Murali Shankar, Elliott Slaughter, Monarin Uervirojnangkoorn, Jana Thayer, Chuck Yoon

*All the great folks at OLCF and ALCF*
HPC is Evolving
HPC is Evolving
HPC is Evolving

Applications -> Workflows

Storage -> Configurable Storage

Compute -> More Compute

Services

APIs

Networking
Superfacility, and Integrated Research Infrastructure
Bringing HPC “Closer” to Experiments

• Data collection rate at XFEL light sources expected to increase by 400x

• Expected to outpace local computing resources

• Large experiments require tight coupling between data collection and analysis ⇒ superfacility model

https://arxiv.org/abs/2206.11992
New Generation of Workflows

Similar need across many science domains! Eg.:

1. Astrophysics and Cosmology: 
   *LSST DESC, CMB/TOAST, DESI, DUNE*
2. High-Energy and Particle Physics: 
   *ATLAS, CMS, LZ*
3. Electron Microscopy and Nanoscience: 
   *NCEM*
4. Nuclear Fusion Experiments: 
   *KSTAR*
New Generation of Workflows

Similar need across many science domains! Eg.:

1. Astrophysics and Cosmology: 
   *LSST DESC, CMB/TOAST, DESI, DUNE*
2. High-Energy and Particle Physics: 
   *ATLAS, CMS, LZ*
3. Electron Microscopy and Nanoscience: 
   *NCEM*
4. Nuclear Fusion Experiments:  
   *KSTAR*
New Generation of Workflows

Similar need across many science domains! Eg.:

1. Astrophysics and Cosmology: 
   LSST DESC, CMB/TOAST, DESI, DUNE
2. High-Energy and Particle Physics: 
   ATLAS, CMS, LZ
3. Electron Microscopy and Nanoscience: 
   NCEM
4. Nuclear Fusion Experiments: 
   KSTAR
Outline

1. The Human Perspective: XFEL + Superfacility
2. The Engineering Nuts and Bolts: Performance Optimization Towards the Exascale
3. Lessons Learned
The Human Perspective:
XFEL + Superfacility
XFEL: Serial Crystallography
XFEL: Serial Crystallography
XFEL: Serial Crystallography
XFEL: Serial Crystallography

(Thomas White, CFEL)
XFEL: Serial Crystallography

(Thomas White, CFEL)

(Chapman, Nature 2011)
When should I move onto the Next Sample?

- Beamtime is scarce!

- Critical live feedback:
  - Does the beam hit the sample?
  - Do we see crystals?
  - Does the data make sense?
  - What is the quality of the data?

- Can I move on to the next sample?
Experimentalists Are In The Driver's Seat
Live Data Analysis for Experiments using the NERSC+LCLS Superfacility
Demo!
Example: Mithrene derivatives

- **Thiorene** (sulfur)
- **Mithrene** (selenium)
- **Tethrene** (telurium)

Types:
- **Inactive**
- **Photo-active**
number of threads should not exceed about 65536. -g sets the reflection buffer size. This depends on the CPU cache size but will rarely need changing.

-g sets the number of reflection groups used for calculating R-comp. This must be greater than 1 but smaller than the total number of reflections for refinement. It is also currently reflection buffer size. This may be not be less than the sum of reflection groups for command line flags override other ways defining space-R reflecting.

The -m value is also used as a seed for the WIGL pseudo-random number.

-t sets the number of threads for each is set to the appropriate number of CPUs. For optimal performance on hyperthreading systems -t should be set to a little more than half the number of CPUs, e.g. -t4 or -t5 for an Intel X7 processor.

Running 8 threads on 8 processors.

Read instructions and data

** WARNING: Input data appear to be merged: CIF file will be incorrect.**

Data: 276 unique, 100 suppressed (int) = 0.0900 (ring signal).

** Cell contents from UNIT instructions: Tset = 0.0000 and

Goof = S = 12.794

Mean shift/psd = 0.091 Max shift = 0.202

w2 = 0.6561 before cycle 4 for 276 data and 9 parameters

Goof = S = 4.533 Restrained Goof = 0.000

Mean shift/psd = 0.0899 Max shift = 0.1877

w2 = 0.6514 before cycle 4 for 276 data and 9 parameters

Goof = S = 4.2811 Restrained Goof = 0.000

Mean shift/psd = 0.0555 Max shift = 0.2214

w2 = 0.6400 before cycle 5 for 276 data and 9 parameters

Goof = S = 4.1904 Restrained Goof = 0.000

Mean shift/psd = 0.0431 Max shift = 0.226 for 276 parameters

max shift = 0.044 A for AgO3 restraints CIF file has been saved with the meta-data.cif file
XFEL structures reveal why thiorene is not photo-active

How is the NERSC + LCLS Superfacility Used?
How is the NERSC + LCLS Superfacility Used?

• Difficult to define what “interactivity” means
  o Eg: Interactive can mean exploration, vs human-in-the-loop
  o Each use HPC differently

• ExaFEL: Case study of an interactive HPC workflow

• NERSC staff deploy code mine log files, and instrument workflows during runtime in order to monitor the whole workflow
(Realtime) Data Analysis Workflow

1. Data Acquisition
2. Automatic Data Transfer
3. On-site storage
4. Responsive Queue
5. HPC Data Analysis
6. Realtime Feedback
7. Generate “Tasks”
8. Interpret Results
9. Experimental Decisions
(Realtime) Data Analysis Workflow

1. How is data getting to NERSC?

2. How do users interact with workflow in real time?

3. How responsive is the job queue?
1. How is Data Getting to NERSC?

- Files are automatically transferred using XRootD immediately after data is collected 15 TB/day
1. How is Data Getting to NERSC?

• Tail latency can disrupt the entire workflow!

• Choice of file system matters
2. How do Users Interact with the Workflow?

[Diagram showing workflow with stages: Measure, Adjust Experiment, Measure, Adjust Experiment, Measure, Adjust Experiment, Collect Data / Tweak, Experiment, Analyze Data, Compute, Interpret, Compute, Interpret, Compute, Interpret, Data Transfer]
2. How do Users Interact with the Workflow?

• An HPC center bridges 3 time scales:
  o Hours => Experiment, Job Scheduler
  o Minutes => Humans
  o Milliseconds => Compute Nodes

• Objective: measure the human timescale
  o How much of the workflow requires “constant” human interaction?
2. How do Users Interact with the Workflow?

- Most images are processed only once
  - This step can be automated once tuned

- Reprocessing as a “fingerprint” of interactivity
Interactive Engagement with Data Sets

- Repeated reprocessing indicates user is exploring data (troubleshooting?)
- Details different from experiment to experiment
Interactive Engagement with Data Sets

• Every beamtime is unique!

• Only ~5% of images reprocessed > 10x
Interactive Engagement with Data Sets

- Most reprocessing takes place on a 25-50 min cadence
- In line with the cadence of measurement runs at LCLS
3. How Responsive is the Job Queue?

- We don’t know when data will need to be analyzed
- Once data is available, queue needs to respond ASAP

- Use reservations
  - 60-80 Nodes
- Idle reservations “waste” compute => try sharing with preemptible jobs
Preemptible Reservations

- Allow preemptible jobs to use reservation:
  - MaxStartDelay=${MINUTES}
    Eg: MaxStartDelay=5

- Make a job preemptible
  - #SBATCH --signal=R:INT@$SECONDS
    Eg. #SBATCH --signal=R:INT@300

- Jobs submitted to the regular QOS where:
  $MINUTES * 60 >= $SECONDS
  are allowed to run in the reservation
  - A job being preempted will receive SIGINT and then has $SECONDS many seconds
to shut down gracefully (before being killed)
Testing Preemptible Reservations
Testing Preemptible Reservations
Testing Preemptible Reservations

- Preemptible jobs don’t leave reservation quickly enough
Conclusion: XFEL + Superfacility

- On-site storage
- Automatic Data Transfer
- Data Acquisition
- Responsive Queue
- Realtime Feedback
- Experimental Decisions
- Interpret Results
- Monitor Everything!
- Reservations
- HPC Data Analysis
- Tail latency can disrupt whole workflow

- ~5% of all data
- Every 5-30 mins

Most images processed only once

50% of reprocessing occurs within 30 mins

Preemption can make use of idle reservation, but issues with scheduler (slurm)
Engineering Nuts and Bolts:
Performance Optimization Towards the Exascale
Deploying CCTBX at NERSC
XRootD automatically transfers data to NERSC DTNs
Job sentinel regularly queries the LCLS REST API for new experiments. It then compares the experiment list with the record of completed data analyses.
If new jobs need to be submitted, the job sentinel constructs and submits jobscripts.
Jobs run in Shifter / Podman containers on compute nodes.

As data is analyzed, the progress is updated in real time to a SQL database.
Users (AoE) can connect to the SQL database, to observe the progress of the data analysis. Users also commit new parameters to be analyzed.
Data Movement XRootD clusters

- eLog
  LCLS metadata manager
- datamvr
  xrdcp process
- kafka
- cmsd
- xrootd
- cmsd
- xrootd
- psexport
- dtn
- cmsd
- xrootd
- cmsd
- xrootd
- cmsd
- xrootd
- ESNET-SENSE API
  used to reserve bandwidth

ana-filesystems
Lustre

SLAC/LCLS
NERSC

scratch
Lustre

Community
file-system

XRootD redirectors
XRootD data servers

xrdcp process
find file, get server to use
get file loc to write
request transfer
get file loc to

xrootd
get new file avail for transfer
register transferred files

ESNET-SENSE API
used to reserve bandwidth
Workflow Coordination hosted on Spin

- Spin was capable of handling over 7k transactions/sec
- Transactions and connections need to be pooled when exceeding 4000 connections (ranks)
Data Analysis

- Data analysis follows sequential stages:
  - spotfinding
  - indexing Bragg spots
  - model refinement
  - integrating Bragg spots
Performance Optimization For The Exascale
How’s the Computation Weather Today?

- Computational Weatherplot:
  - Each line shows work done by one MPI rank
  - There is no “the cctbx.xfel workload”
How’s the Computation Weather Today?

- **Computational Weatherplot:**
  - Each line shows work done by one MPI rank
  - There is no “the cctbx.xfel workload”

![Graph showing MPI-Communication Bound](image)
How’s the Computation Weather Today?

- Computational Weatherplot:
  - Each line shows work done by one MPI rank
  - There is no “the cctbx.xfel workload”

---

Start-Up and I/O (PSANA)
Spot Detection (DIALS)
Indexing (DIALS)
Refinement (DIALS)
Integrating (DIALS)
How’s the Computation Weather Today?

- Computational Weatherplot:
  - Each line shows work done by one MPI rank
  - There is no “the cctbx.xfel workload”

```
Start-Up and I/O (PSANA)
Spot Detection (DIALS)
Indexing (DIALS)
Refinement (DIALS)
Integrating (DIALS)
```
Performance Portability

- Use Kokkos ([https://github.com/kokkos](https://github.com/kokkos)) to generate code for each architecture
- Kokkos kernel abstractions are hardware independent
Performance Portability

- Kokkos-generated code outperformed original CUDA kernels
  - Kokkoss generated kernels better at filling available resources on GPU

- Hide workflow and I/O latency by sharing GPU among several MPI ranks
  - Ideal number of ranks/GPU depend on how much of the GPU each kernel can use

Special Issue of the 2021/2022 Cray User Group Conference (CUG 2021/2022), Concurrency and Computation: Practice and Experience
Vendors + Libraries

- Kokkos does not provide portable interfaces to vendor libraries (eg. cuBLAS, cuFFT/rockBLAS, rocFFT)
- Requires active **engagements** with vendors and developer community to design portable and performant libraries

- [https://github.com/elliottslaughter/cufinufft](https://github.com/elliottslaughter/cufinufft)
- [https://github.com/JBlaschke/PybindGPU](https://github.com/JBlaschke/PybindGPU)

Load Balancing
Load Balancing

Fork-Join Communication Pattern: Rank with the most work is holding up the rest during MPI_Barrier
Load Balancing
Load Balancing

- Simple heuristic load-balancing (based on number of active pixels) results in very good load balancing

- Quick change results in 20% speedup
Lesson Learned:
Performance + Portability
Planning
Packaging
Priority
Packets
Performance + Portability

- When developing code of HW accelerators:
  - Keep portability a high priority: [https://github.com/kokkos/kokkos](https://github.com/kokkos/kokkos)
  - Kernel abstractions are a useful tool for getting good performance out of unfamiliar hardware (on a budget)

- Measure Everything! (profiling, debugging, tail latencies)
  - Instrumentation libraries (e.g. [https://github.com/LLNL/GOTCHA](https://github.com/LLNL/GOTCHA), [https://github.com/NERSC/timemory](https://github.com/NERSC/timemory)) do this with minimal effort
  - Workflow managers come with some monitoring tools out of the box

- Fast Feedback!
  - Measured performance data needs to communicated to users in a timely manner

- Porting ExaFEL is still a high-touch effort
Planning: Workflow Coordination

• New class of workflows include multiple data processing steps - Central **scalable** management and persistent state becomes necessary
  o Often hosted on login nodes ⇒ scaling becomes an issue
  o CCTBX: MySQL database for logging process state (completed, running, waiting)
  o Others: MongoDB (fireworks), or sometimes just writing to file

• Collaborative workspaces to facilitate dataset access and analysis
  o NERSC features **collaborative** accounts (users from a particular group can login with the collaborative account credentials)

• Use a workflow manager! (or a **data center’s API**) 
  o Data centers can customize these to account for their system needs
Packaging: Containers

- Design images to be portable:
  - Can Streamline cross-facility workflows by standardizing image build process
  - Optimize dependency handling, data analysis workflows tend to have complex dependencies
  - Shifter / Podman mounts NERSC-specific libraries into image
  - Isolate Anaconda’s conflicts with OS

- Significantly lower import/loading times:
  - Image cached to node-local storage
  - Improved scalability even for few MPI Ranks

Dynamic linker’s symbol resolution uses node-local storage ⇒ many ranks do not flood file-system.
Priority: Job Queue and Urgent Computing

- Real-time data analysis workloads are bursty:
  - Experimental operators require analysis to make decisions about future runs
  - Regular queue turnaround time insufficient for real-time workflow
- Reserving nodes for bursty use is wasteful:
  - Nodes would sit idle
  - Solution: share reserved nodes with preemptible jobs
- Future: Explore flexible resource allocation (e.g. malleable jobs)
Packets: Network and I/O

• In data analysis workflows, file systems and network can become bottlenecks

• I/O Optimization is Crucial:
  o Optimize software (eg. python logger) for high-frequency parallel I/O
  o Write logs to Burst Buffer

• Experience during Beamtimes:
  o Transfers ran smoothly (most of the time), can switch redirect destination in emergencies
  o **FS performance can limit the transfer rate**

• Future Work:
  o Explore in-memory burst buffers, eg [https://github.com/LLNL/UnifyFS](https://github.com/LLNL/UnifyFS)
  o Offer more fine-grained control over which data is transferred and when
  o Better monitoring and alerting
Packets: Network and I/O

- In data analysis workflows, file systems and network can become bottlenecks

- I/O Optimization is Crucial:
  - Optimize software (e.g., python logger) for high-frequency parallel I/O
  - Write logs to Burst Buffer

- Experience during Beamtimes:
  - Transfers ran smoothly (most of the time), can switch redirect destination in emergencies
  - FS performance can limit the transfer rate

- Future Work:
  - Explore in-memory burst buffers, e.g., https://github.com/LLNL/UnifyFS
  - Offer more fine-grained control over which data is transferred and when
  - Better monitoring and alerting
Extra Bits
Sharing GPUs Between MPI Ranks

Time to simulate N images

Best fit: Wall clock = 1.07 s + 1.22 s * N

Time to simulate each image

N = 1 rank

N = 5 ranks

100 ms

Host to device memcpy
Kernel execution
Example Issue: LCLS Slow File Transfer

1. Files transfer from LCLS via
   LCLS SCRATCH => ESNet => NERSC Scratch

2. Most files transfer quickly

3. Some files come in slowly

4. Open Work Items:
   a. Why?
      i. LDMS on DTN (identify slow FS)?
      ii. Work with ESNet?
   b. Accept slow files
      => Design workflow so that slow files don’t hold up other work
Fireworks Workflow Manager

```python
qadapter_1 = CommonAdapter(
    q_type = "SLURM",
    rocket_launch=f"rlaunch --json -l '{lp_str}' -w '{fw_str_1}' singleshot",
    constraint="gpu",
    account="nstaff",
    walltime='00:02:00',
    qos="regular",
    nodes="2"
)

fw_1 = Firework(ScriptTask.from_str(
    "srun -n 2 cctbx.python -m mpi4py.bench helloworld"
), spec={"_category": "n2", "_fworker": "mpi_2_fworker"}, name="mpi_2")

workflow = Workflow(
    [fw_1, fw_2],
    {fw_1: fw_2},
    name="multi_queue"
)

launchpad.add_wf(workflow)
rapidfir(launchpad, fworker_1, qadapter_1, sleep_time=60, reserve=True)
```
This is really the end