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

NERSC

(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

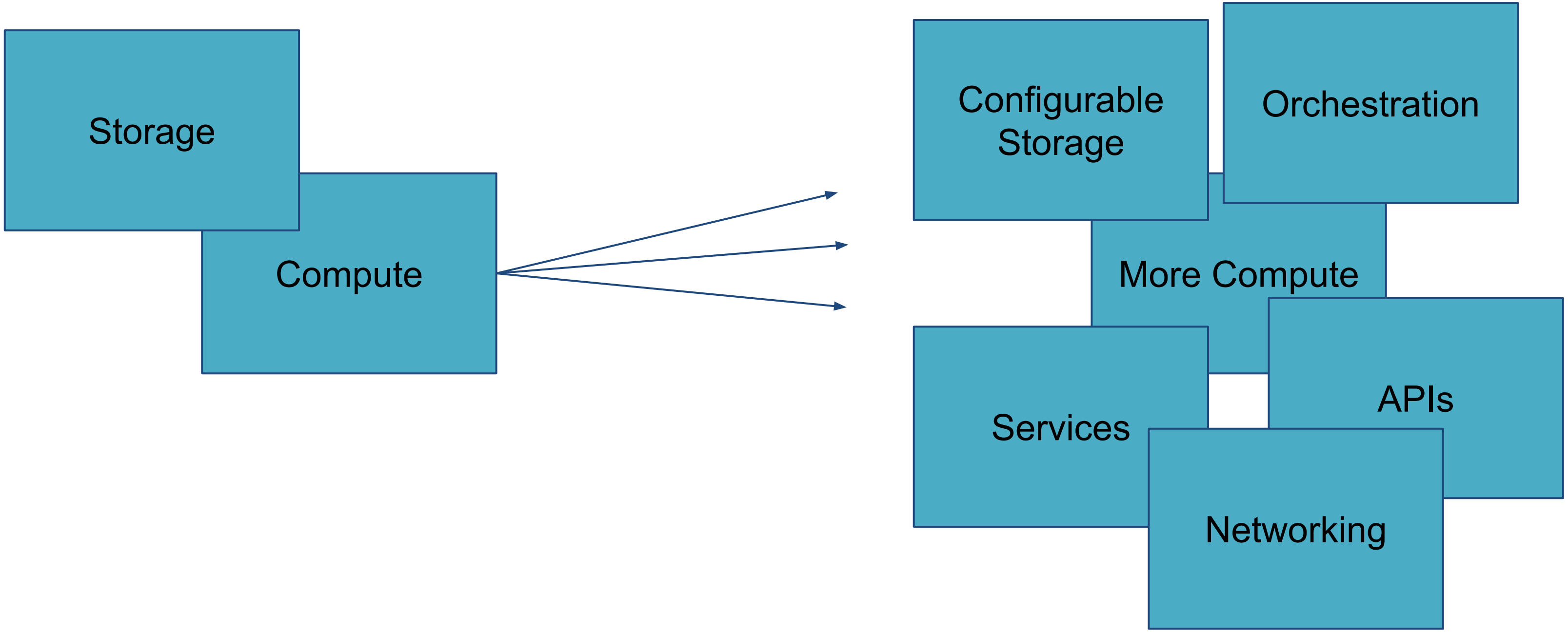
+ All the great folks at OLCF and ALCF

Johannes Blaschke
Data Science Engagement Group, NERSC

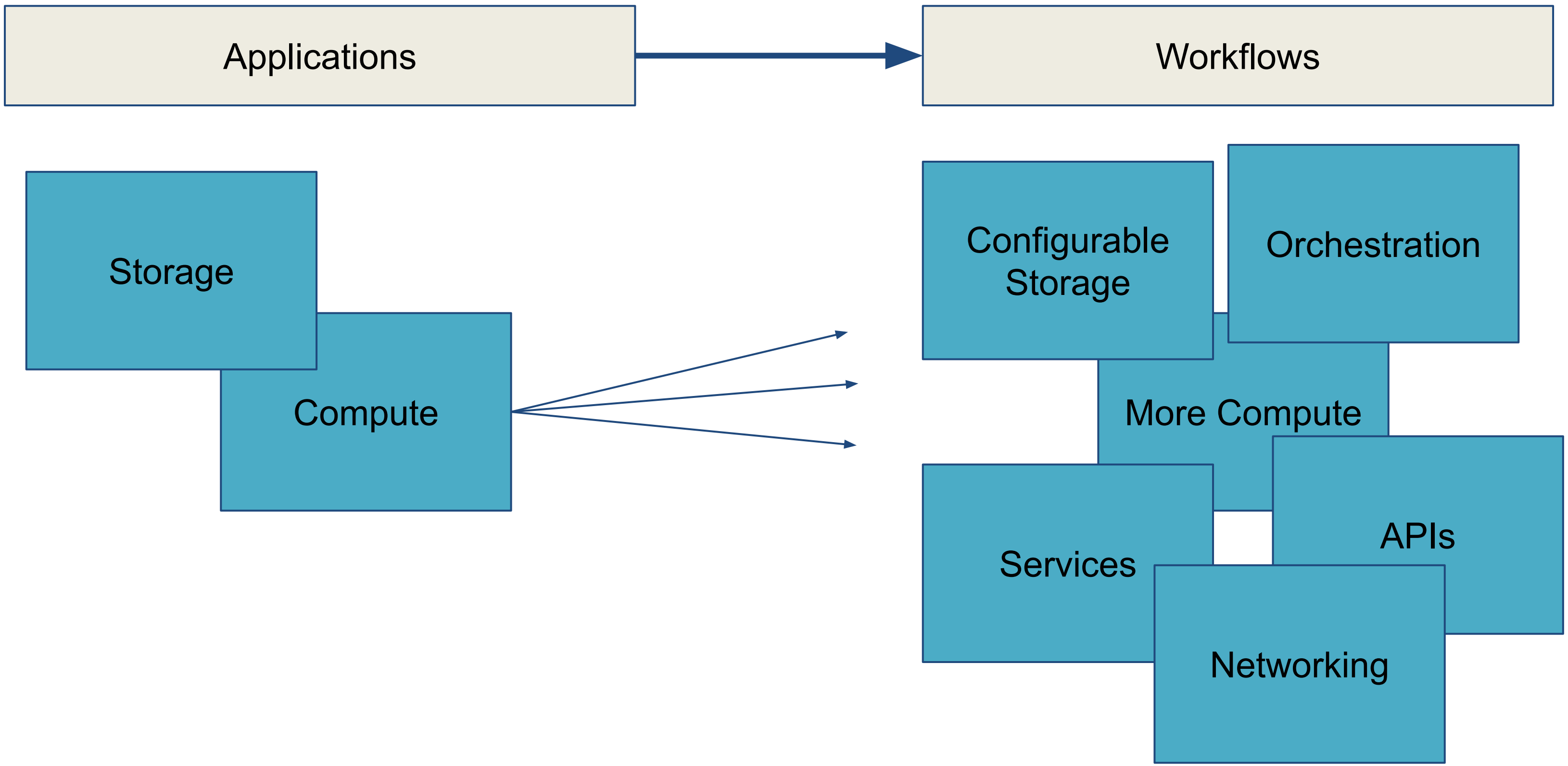
IPAM, NMEW Workshop May 4, 2023

HPC is Evolving

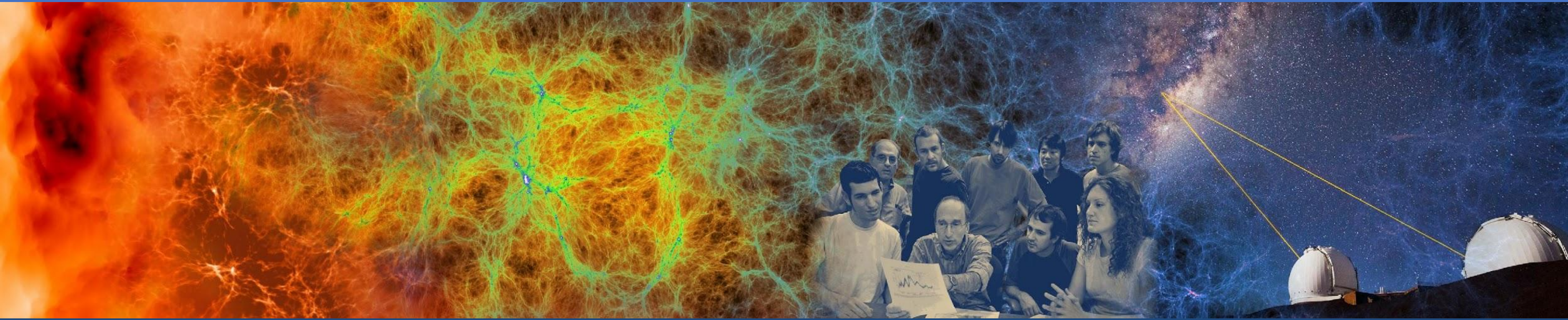
HPC is Evolving



HPC is Evolving

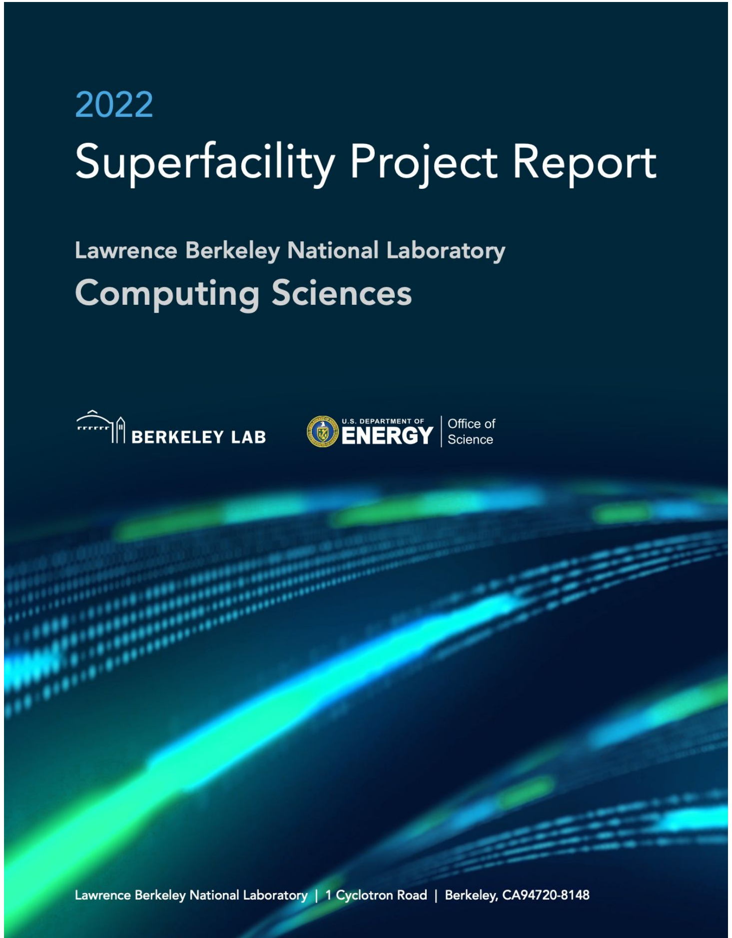


Superfacility, and Integrated Research Infrastructure



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

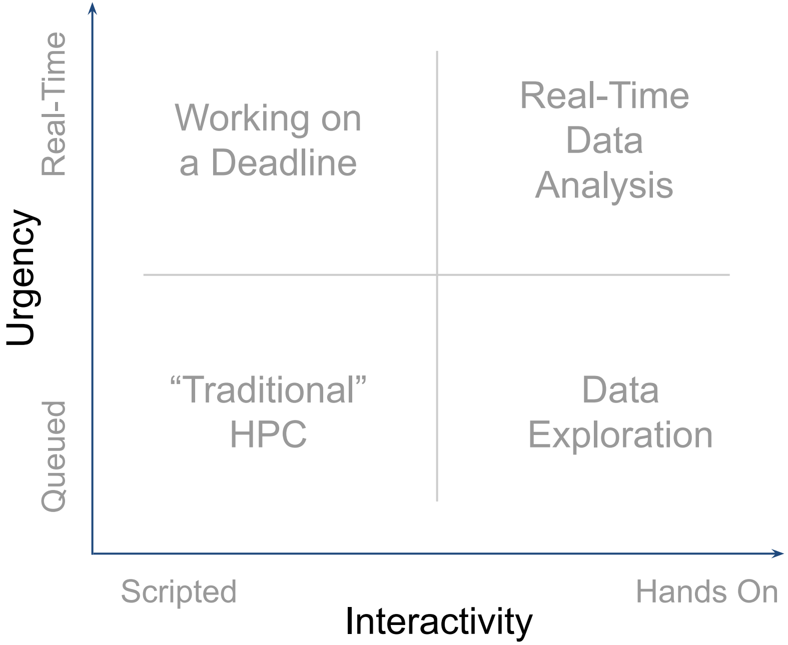
Similar need across many science domains! Eg.:

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.:

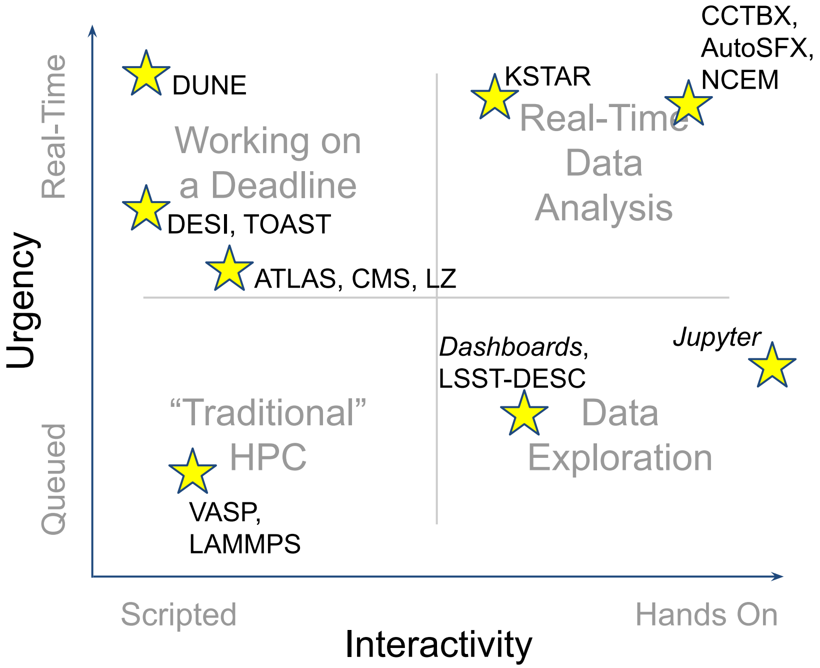
1. Astrophysics and Cosmology:
   LSST DESC, CMB/TOAST, DESI, DUNE
2. High-Energy and Particle Physics:
   ATLAS, CMS, LZ
3. Electron Microscopy and Nanoscience:
   NCEM
4. Nuclear Fusion Experiments:
   KSTAR

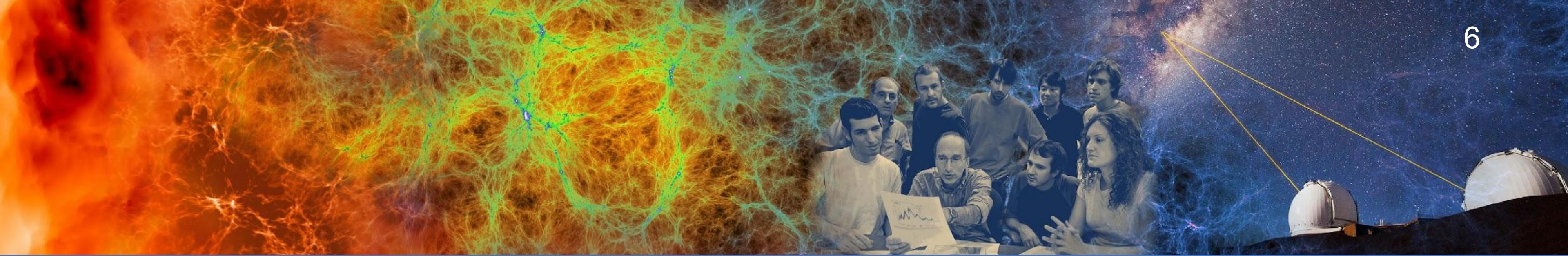


New Generation of Workflows

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



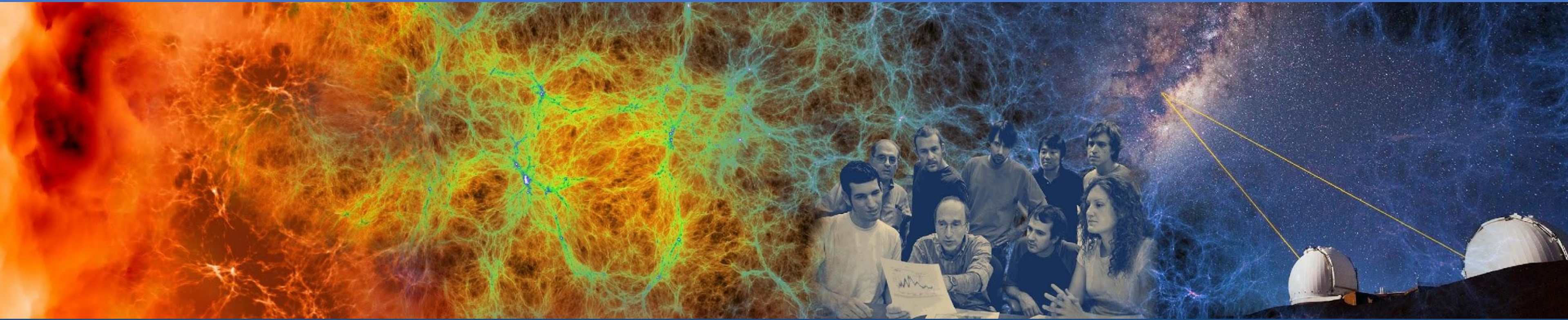


Outline

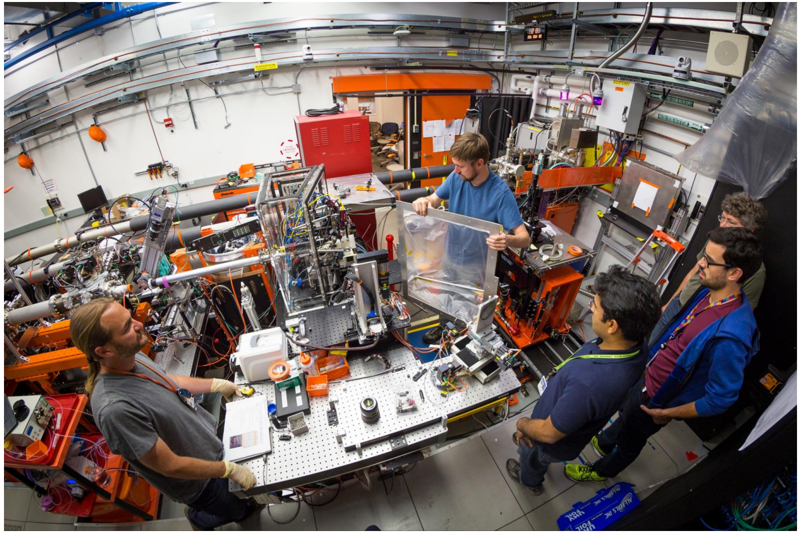
1. The Human Perspective: *XFEL + Superfacility*
2. The Engineering Nuts and Bolts: *Performance Optimization Towards the Exascale*
3. Lessons Learned

The Human Perspective:

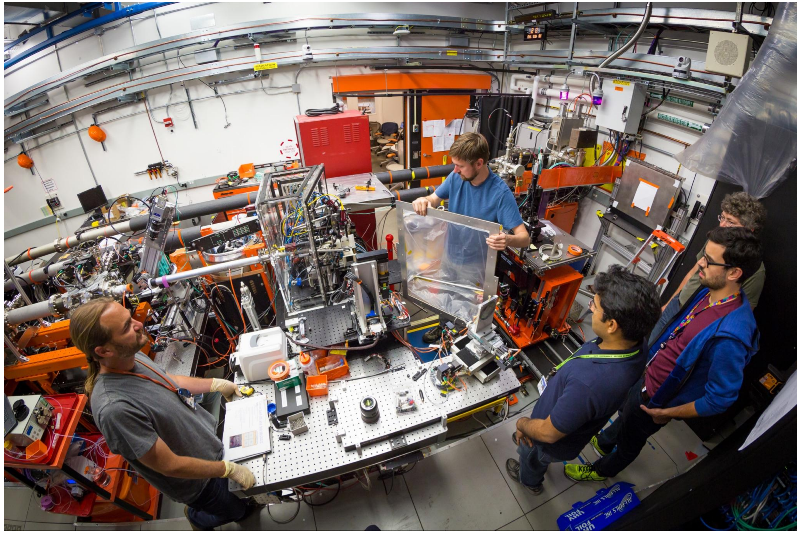
XFEL + Superfacility

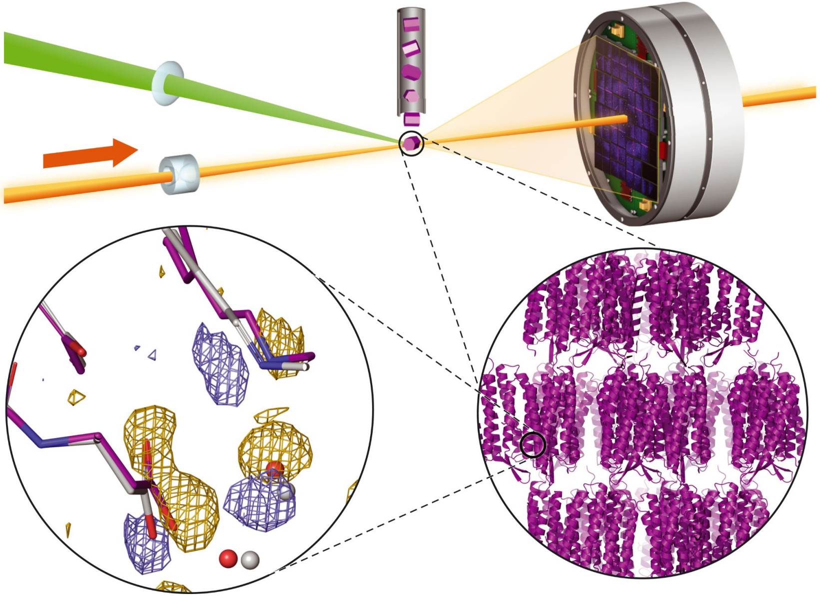


XFEL: Serial Crystallography



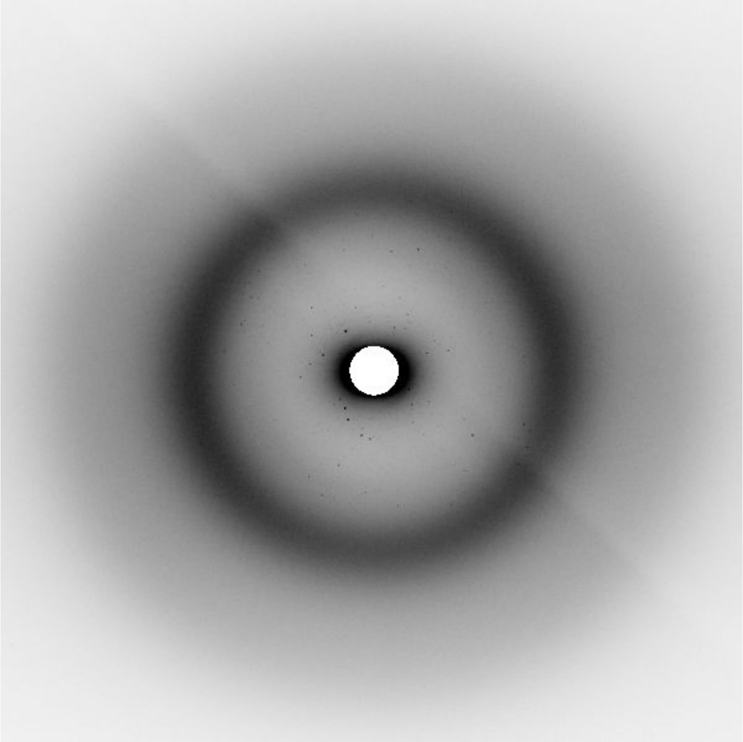
XFEL: Serial Crystallography



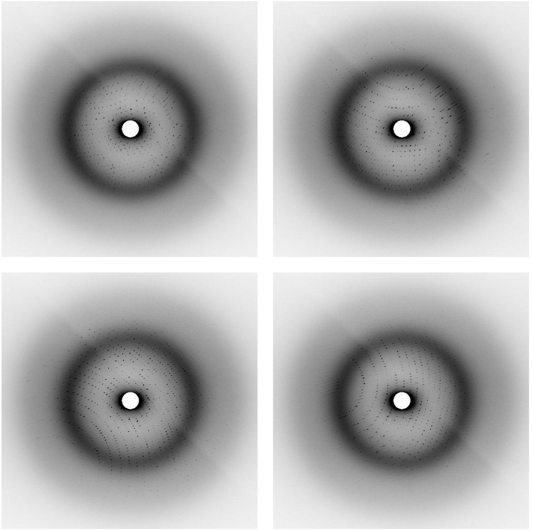


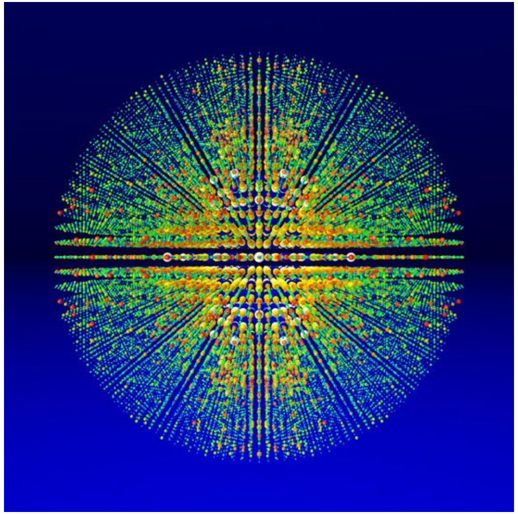
(Brändén, Science 2021)

XFEL: Serial Crystallography



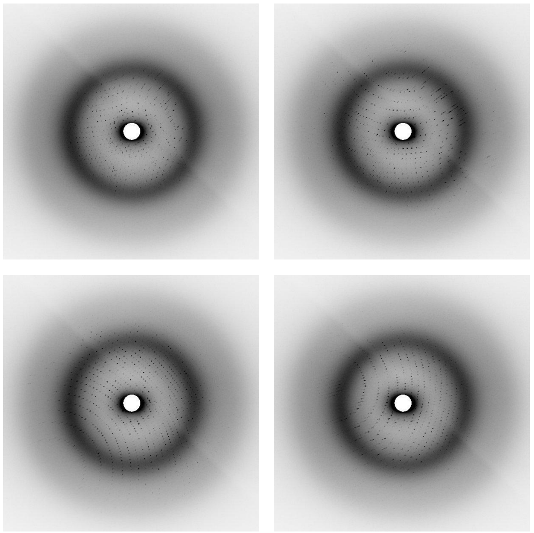
XFEL: Serial Crystallography

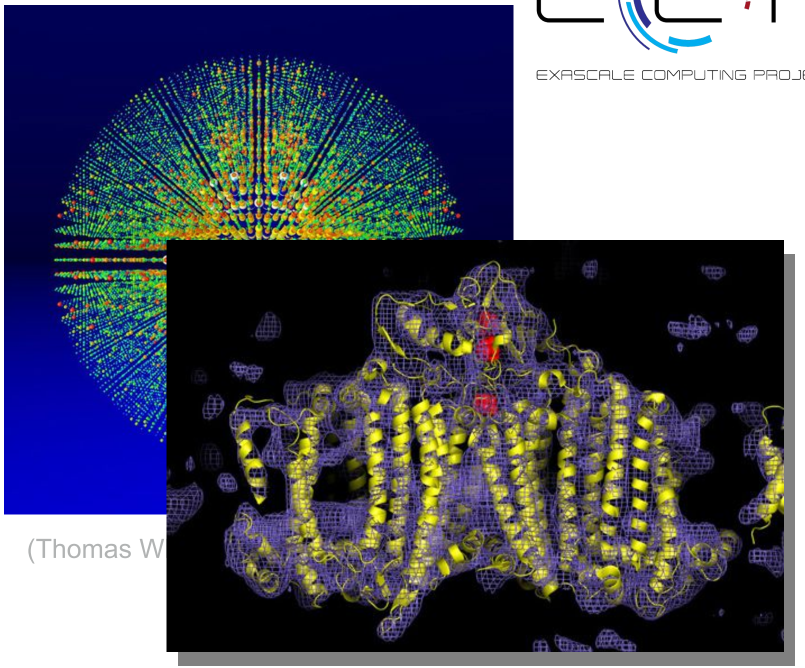




(Thomas White, CFEL)

XFEL: Serial Crystallography



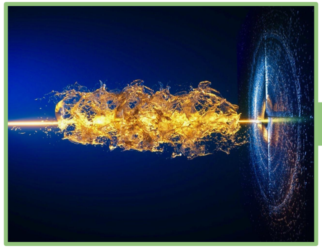


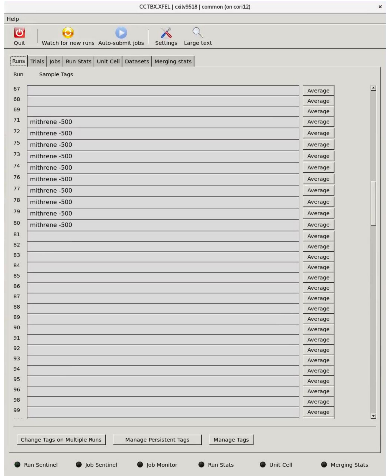
(Thomas W

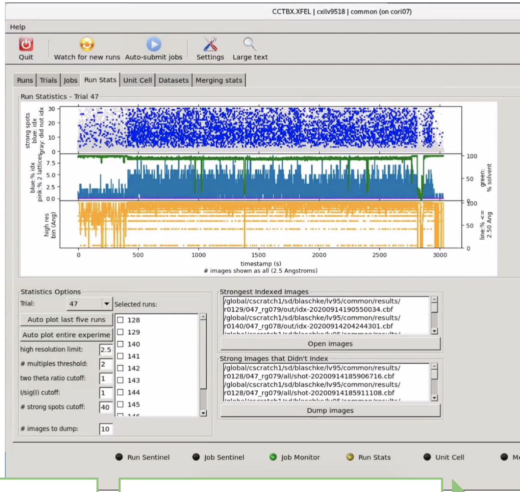
(Chapman, Nature 2011)

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?





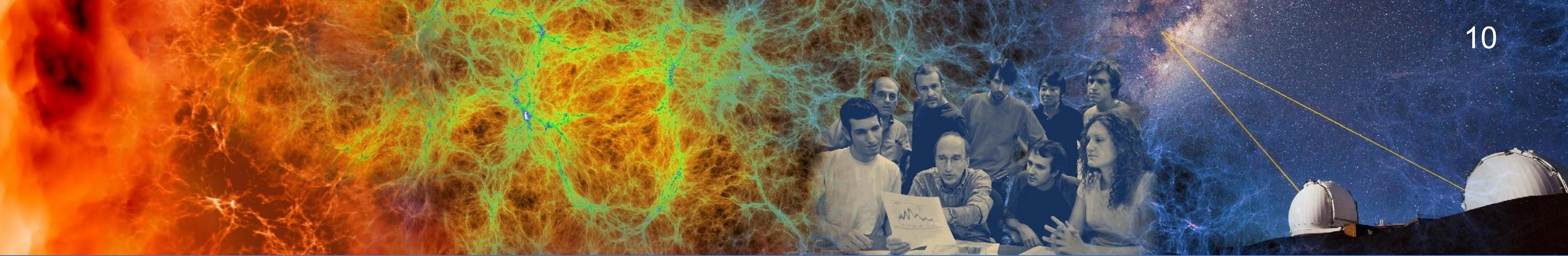


Incoming data → Build jobs → Montor Analysis

NERSC

BERKELEY LAB
Bringing Science Solutions to the World

U.S. DEPARTMENT OF ENERGY
Office of Science

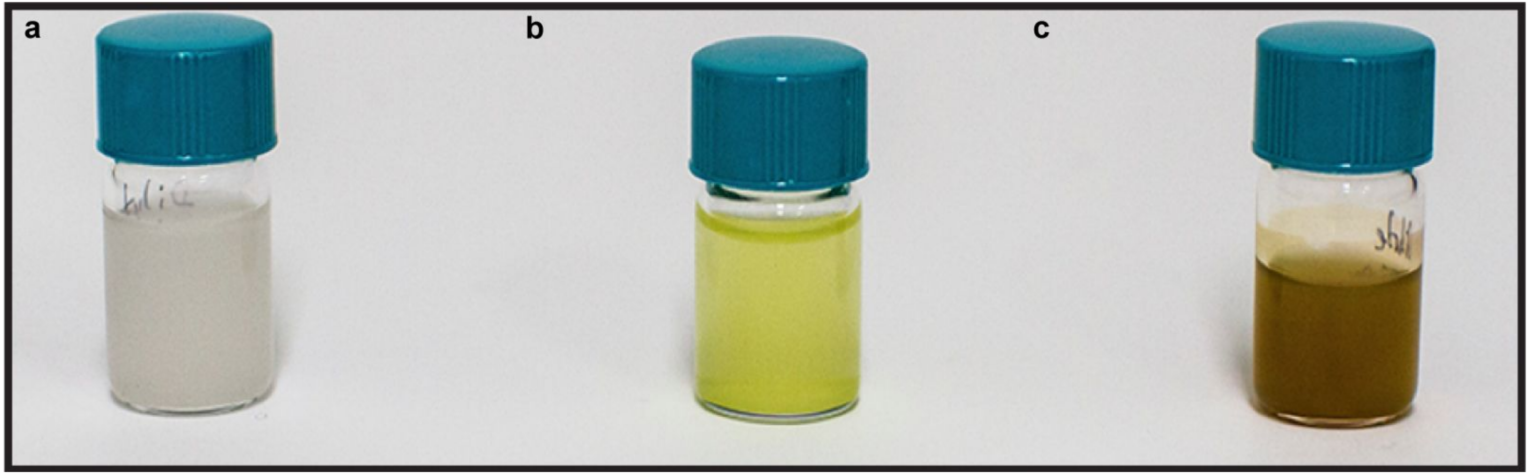


Experimentalists Are In The Driver's Seat

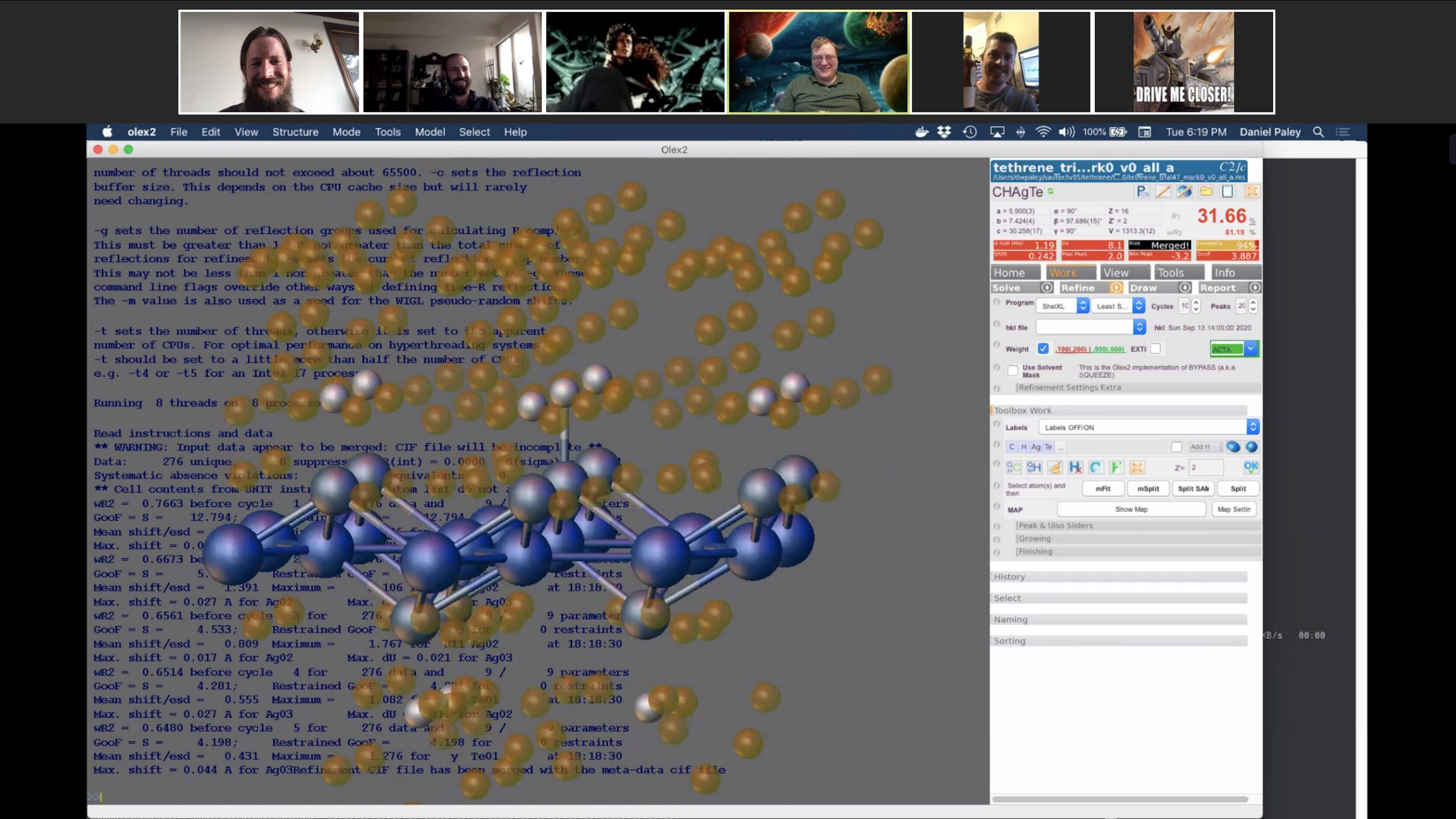
Live Data Analysis for Experiments using the NERSC+LCLS Superfacility

Example: Mithrene derivatives

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

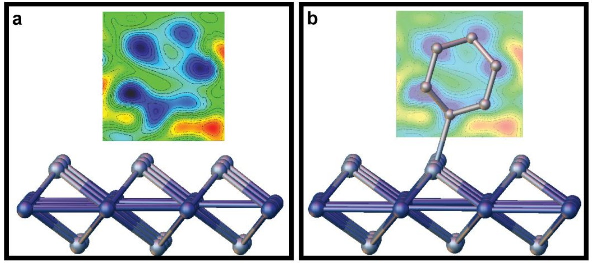


Inactive   Photo-active

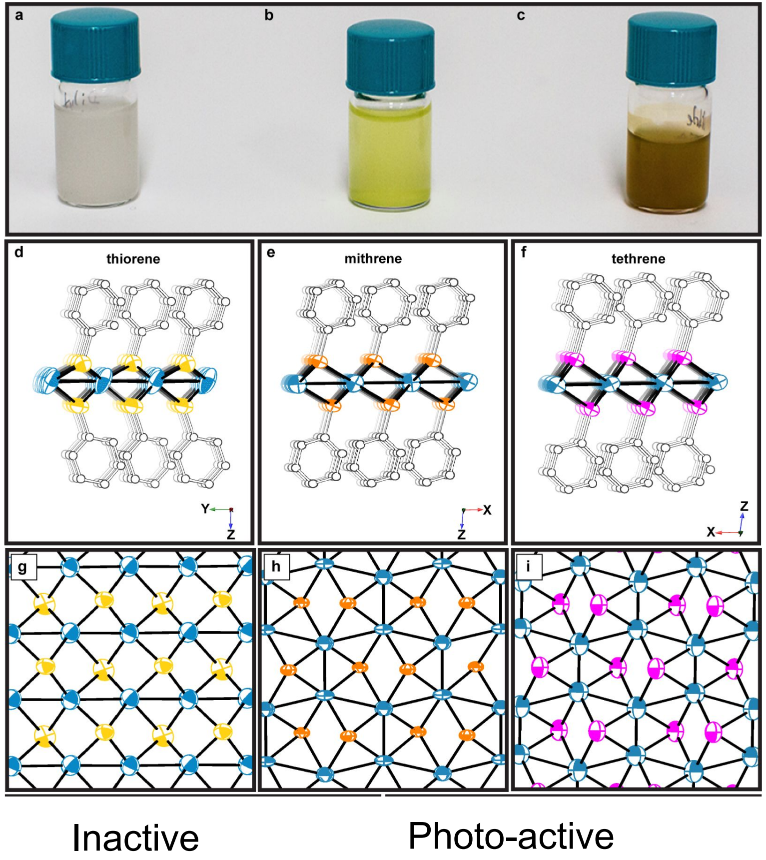


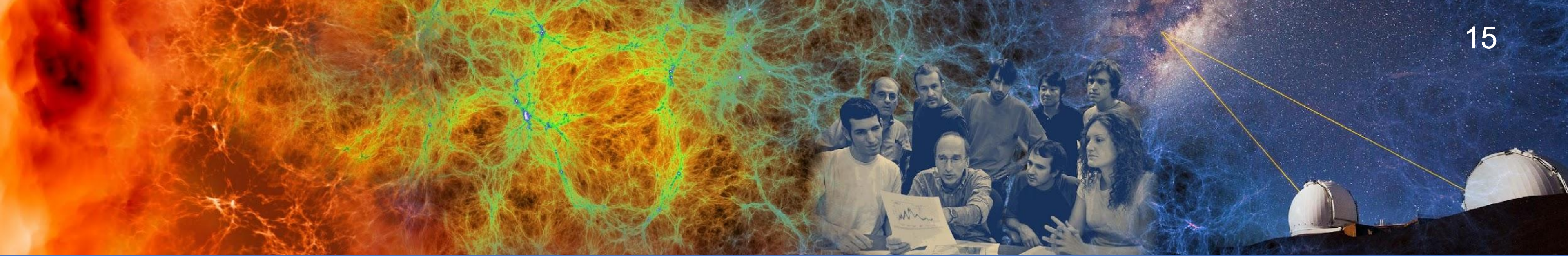
DRIVE ME CLOSER!
olex2 File Edit View Structure Mode Tools Model Select Help
100% Tue 6:19 PM Daniel Paley
Olex2
number of threads should not exceed about 65500. -c sets the reflection
buffer size. This depends on the CPU cache size but will rarely
need changing.
The -m value is also used as a seed for the WIGL pseudo-random shifts.
-t sets the number of threads, otherwise it is set to the apparent
number of CPUs. For optimal performance on hyperthreading systems
-t should be set to a little more than half the number of CPUs;
Read instructions and data
GooF = S = 4.533; Restrained GooF =
Mean shift/esd = 0.809 Maximum = 1.767
Max. shift = 0.017 A for Ag02 Max. dU = 0.021 for Ag03
wR2 = 0.6514 before cycle 4 for 276
Mean shift/esd = 0.555 Maximum =
Max. shift = 0.027 A for Ag03
wR2 = 0.6480 before cycle 5 for 276
GooF = S = 4.198;
Mean shift/esd = 0.431
Max. shift = 0.044 A for Ag03Refinement CIF file has been merged with the meta-data cif file
tethrene_tri...rk0_v0_all_a C2/c
CHAgTe
a = 5.900(3) α = 90° Z = 16
b = 7.424(4) β = 97.686(15)° Z' = 2
c = 30.258(17) γ = 90° V = 1313.3(12)
31.66 %
61.19 %
1.19 8.1 Merged! 94%
0.242 2.0 -3.2 3.887
Home Work View Tools Info
Solve Refine Draw Report
Program ShelXL Least S... Cycles Peaks
hkl file hkl: Sun Sep 13 14:00:00 2020
Weight EXTI ACTA
Use Solvent Mask This is the Olex2 implementation of BYPASS (a.k.a. SQUEEZE)
Refinement Settings Extra
Toolbox Work
Labels Labels OFF/ON
C H Ag Te Add H
Select atom(s) and then mFit mSplit Split SAN Split
MAP Show Map Map Settin
Peak & Uiso Sliders
Growing
Finishing
History
Select
Naming
Sorting

XFEL structures reveal why thiorene is not photo-active



Schriber EA, Paley DW *et. al.* (2021) “Chemical Crystallography by Serial Femtosecond X-ray Diffraction.” Nature **601**, 360-365.





How is the NERSC + LCLS Superfacility Used?

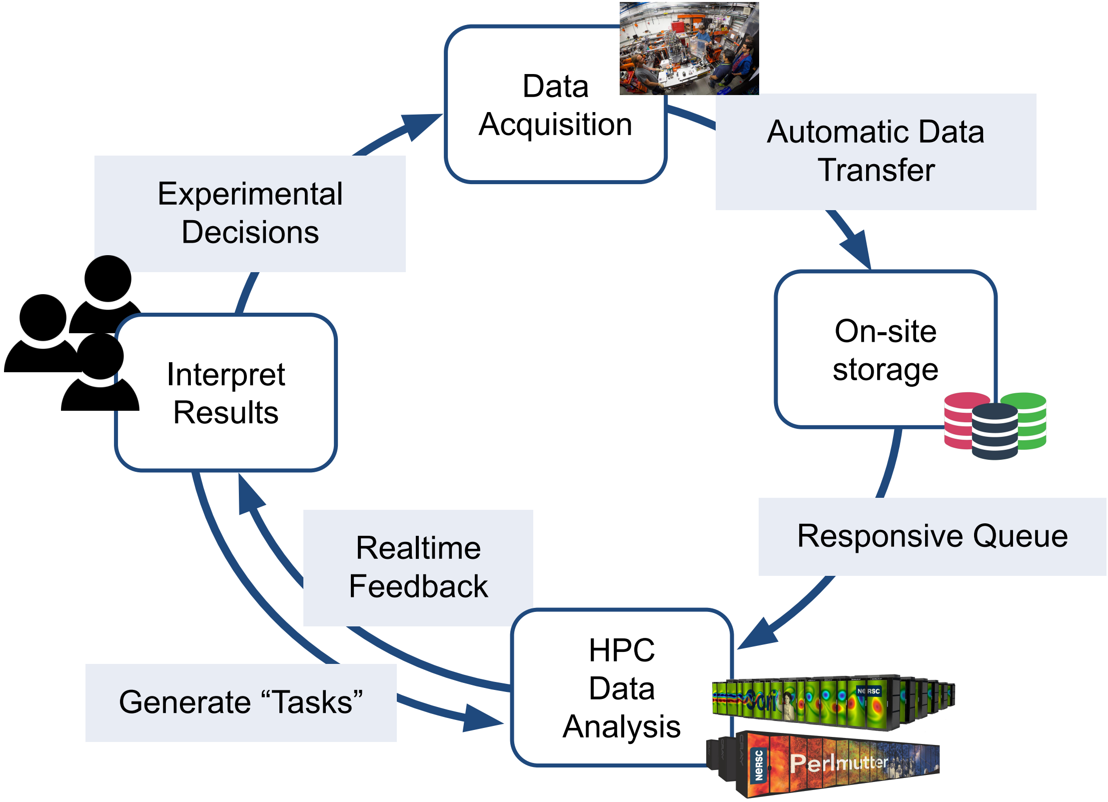
How is the NERSC + LCLS Superfacility Used?

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

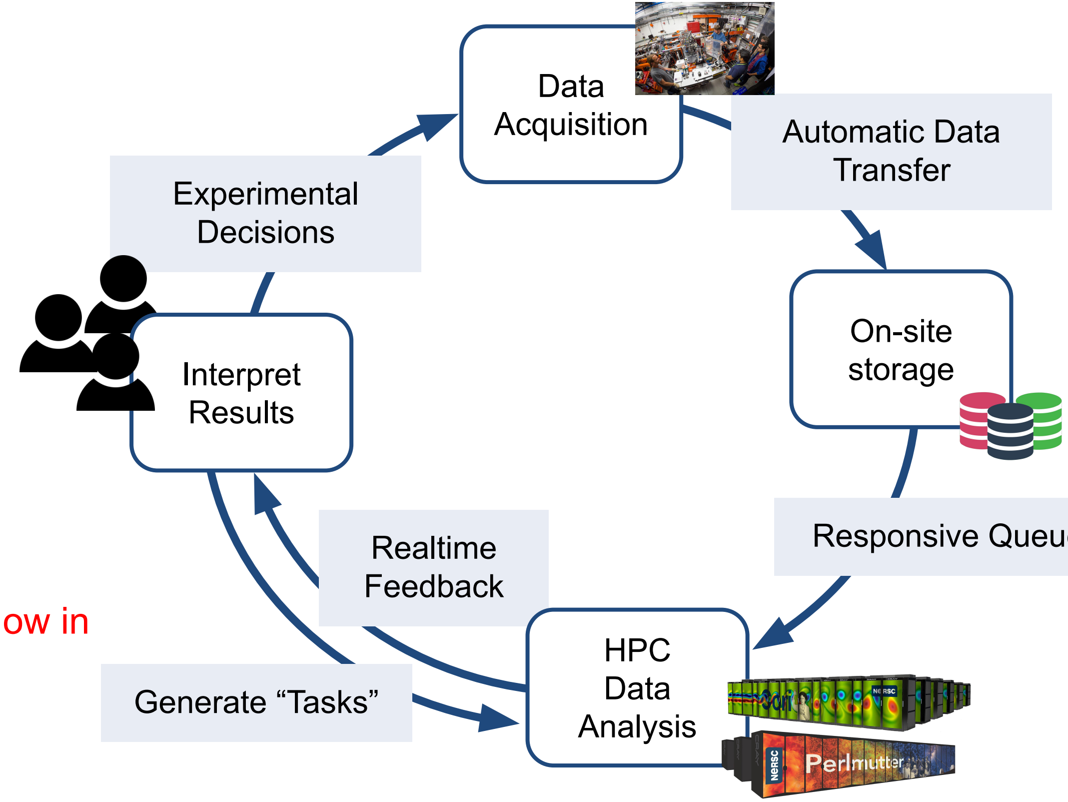
- ExaFEL: Case study of an interactive HPC workflow

- NERSC staff deploy code mine log files, and instrument workflows during runtime in order to monitor the whole workflow

(Realtime) Data Analysis Workflow



(Realtime) Data Analysis Workflow



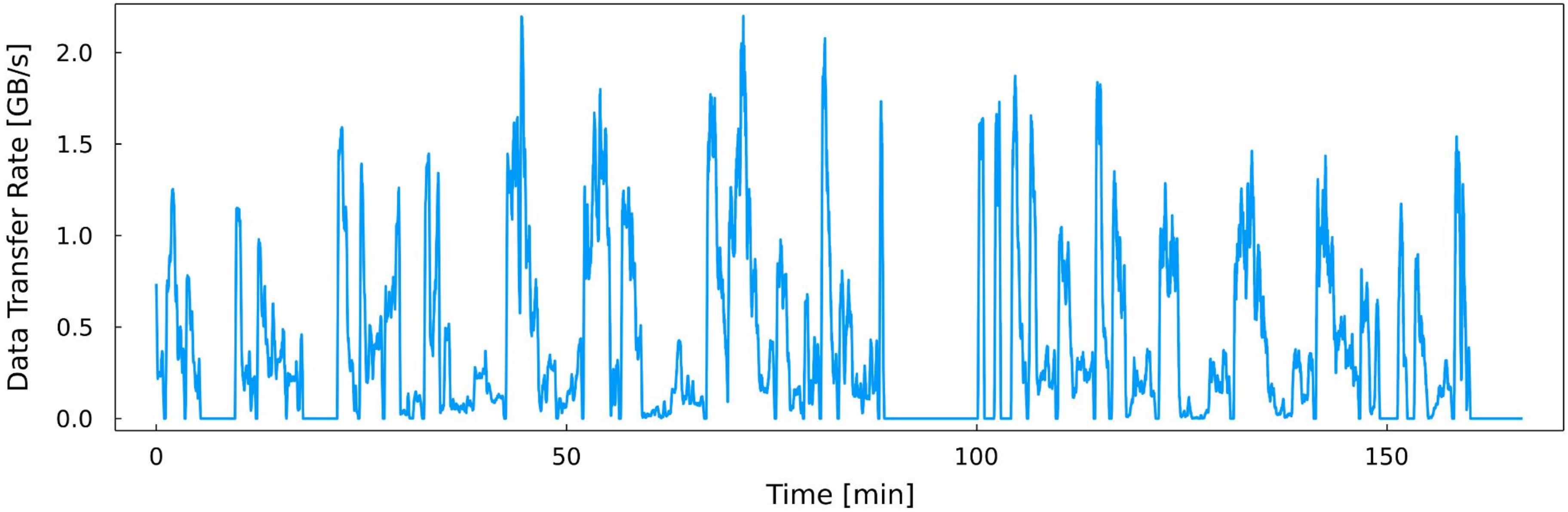
1. How is data getting to NERSC?

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



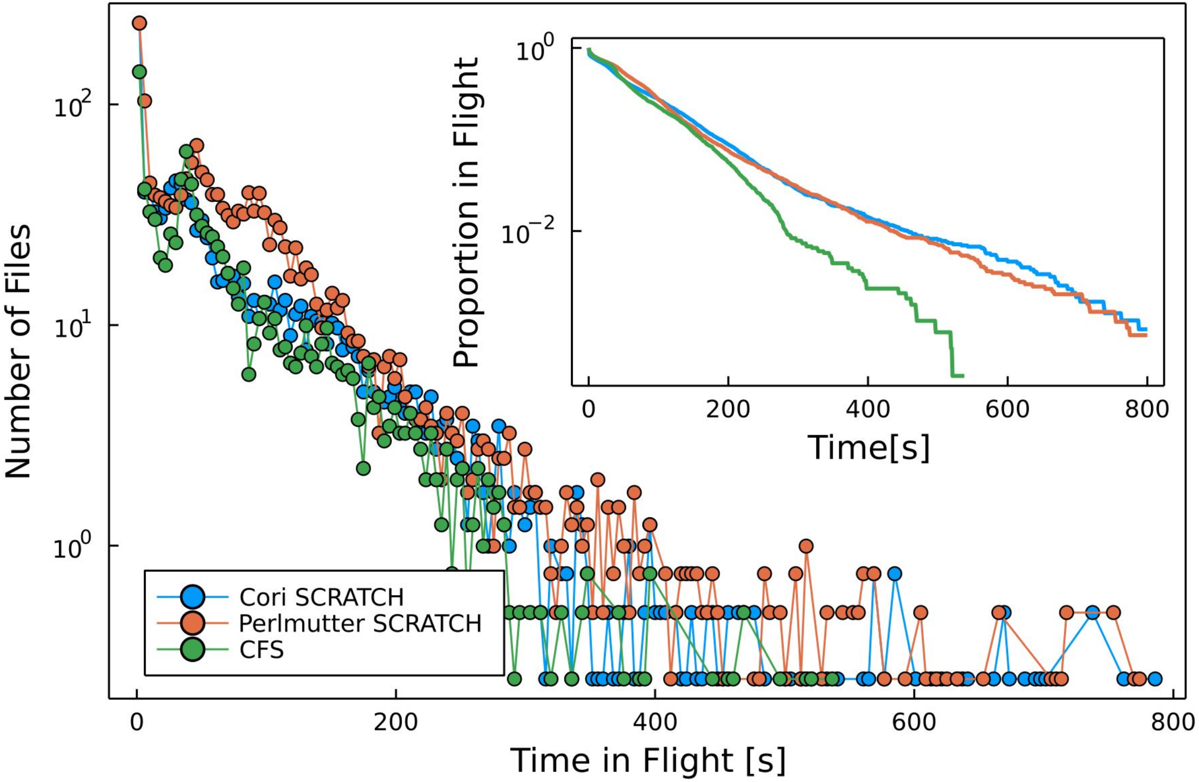
NERSC

BERKELEY LAB
Bringing Science Solutions to the World

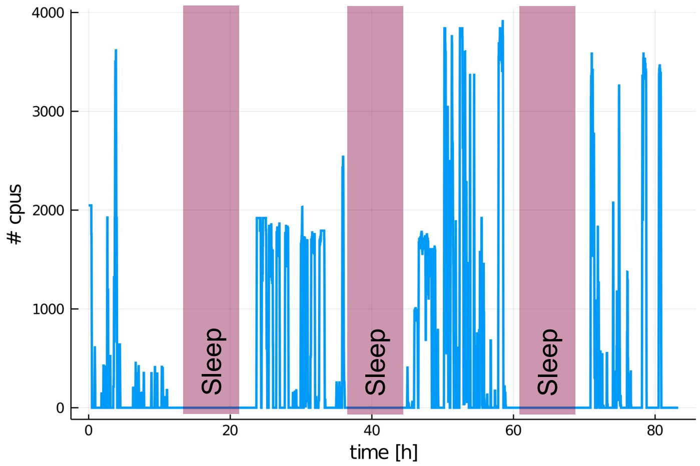
U.S. DEPARTMENT OF ENERGY
Office of Science

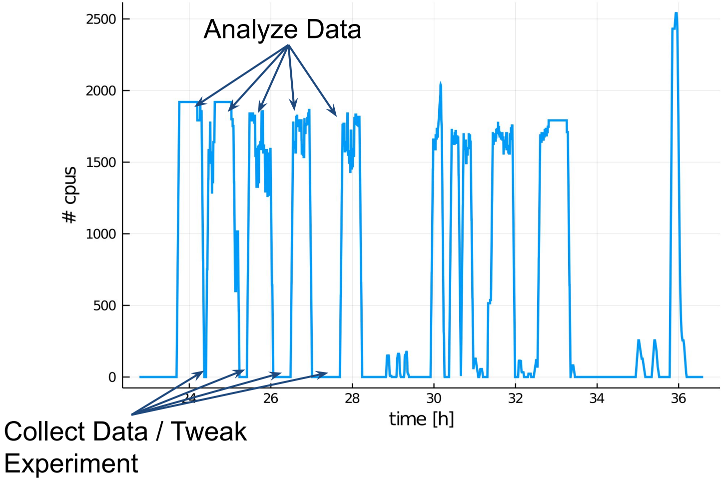
1. How is Data Getting to NERSC?

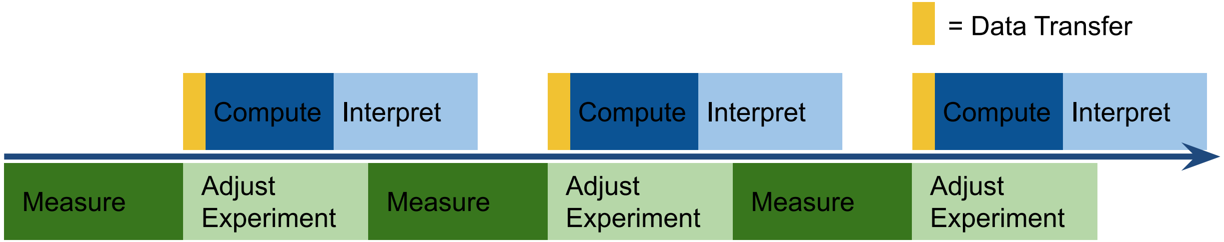
- Tail latency can disrupt the entire workflow!
- Choice of file system matters



2. How do Users Interact with the Workflow?







NERSC BERKELEY LAB Bringing Science Solutions to the World U.S. DEPARTMENT OF ENERGY Office of Science

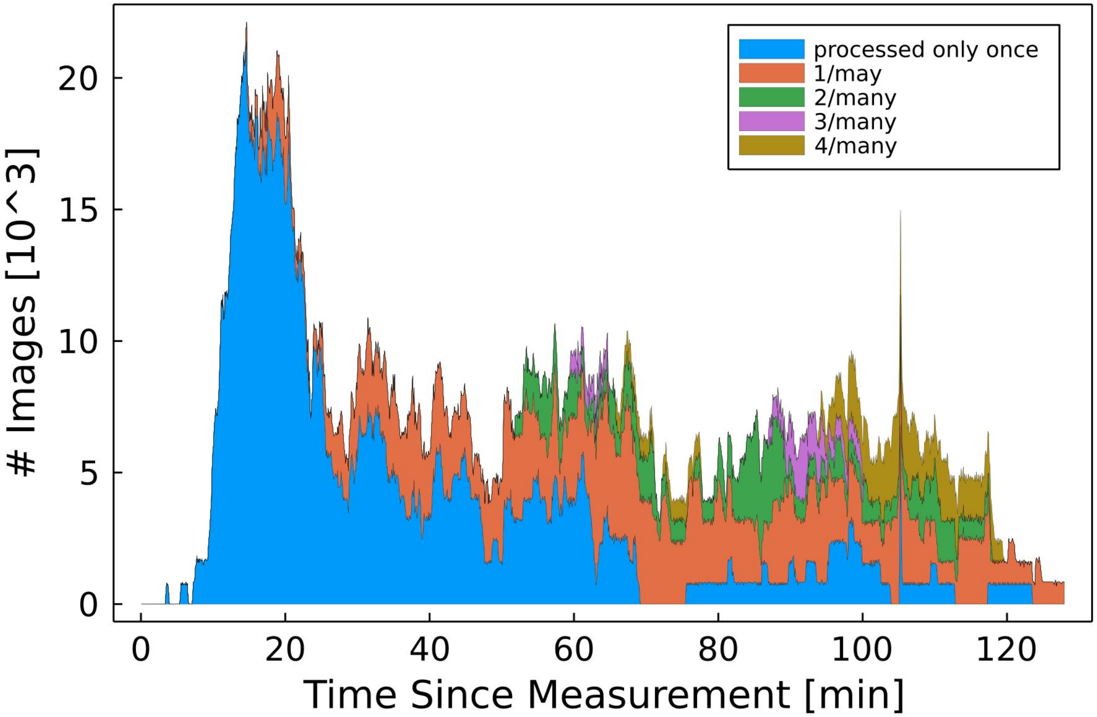
2. How do Users Interact with the Workflow?

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

- Objective: measure the human timescale
  - How much of the workflow requires “constant” human interaction?

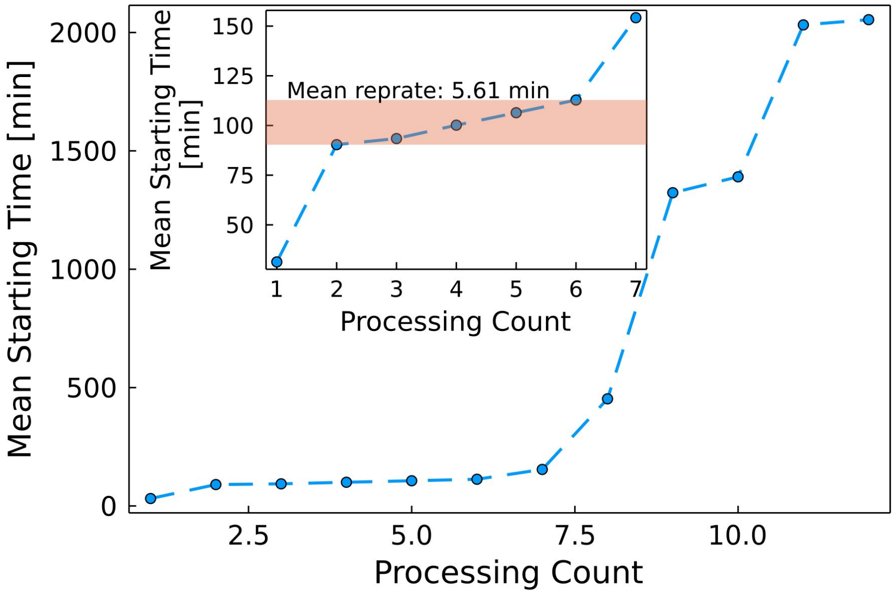
2. How do Users Interact with the Workflow?

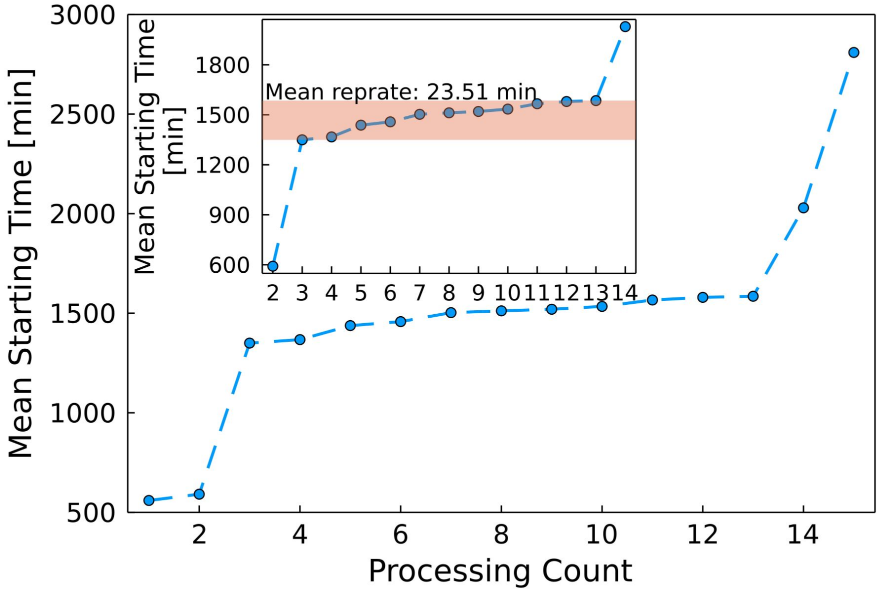
- Most images are processed only once
  - This step can be automated once tuned
- Reprocessing as a "fingerprint" of interactivity



Interactive Engagement with Data Sets

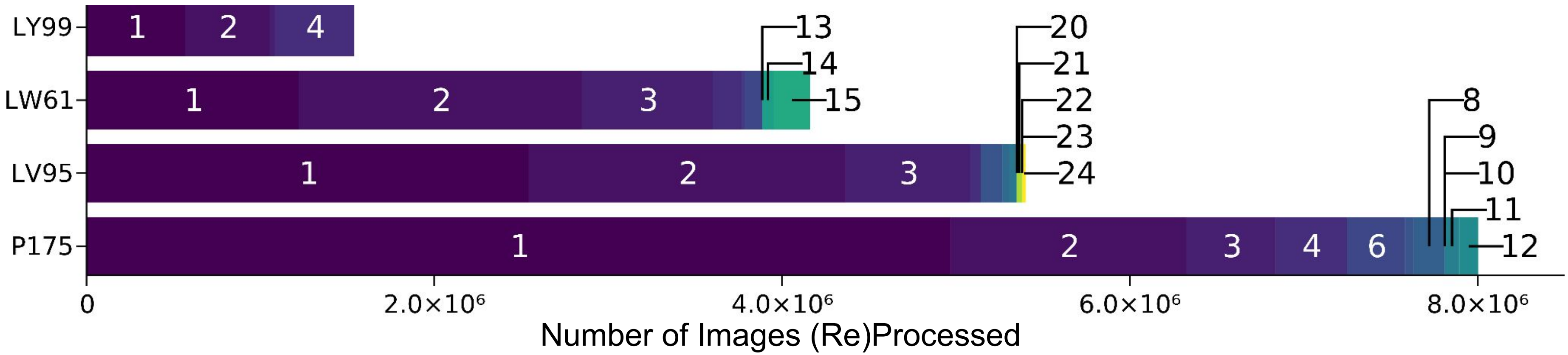
- Repeated reprocessing indicates user is exploring data (troubleshooting?)
- Details different from experiment to experiment





Interactive Engagement with Data Sets

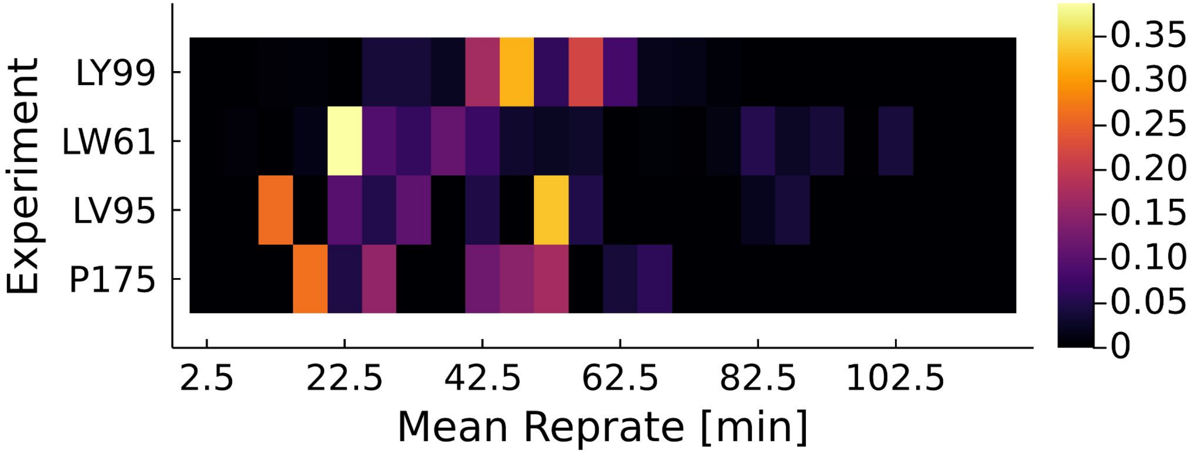
- Every beamtime is unique!

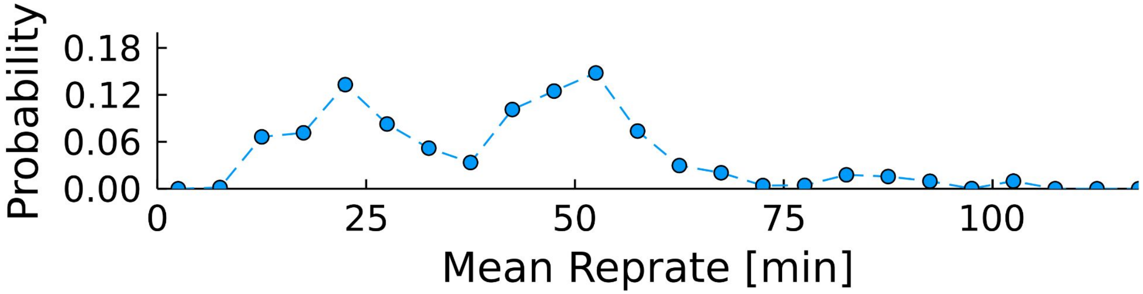


- Only ~5 % of images reprocessed > 10x

Interactive Engagement with Data Sets

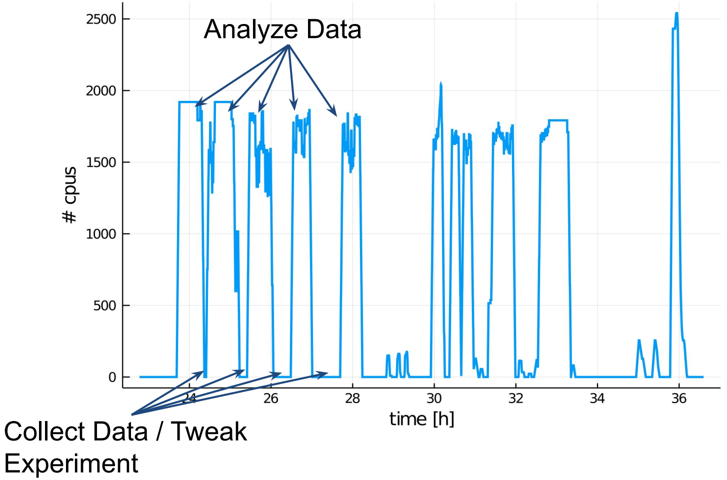
- Most reprocessing takes place on a 25-50 min cadence
- In line with the cadence of measurement runs at LCLS





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

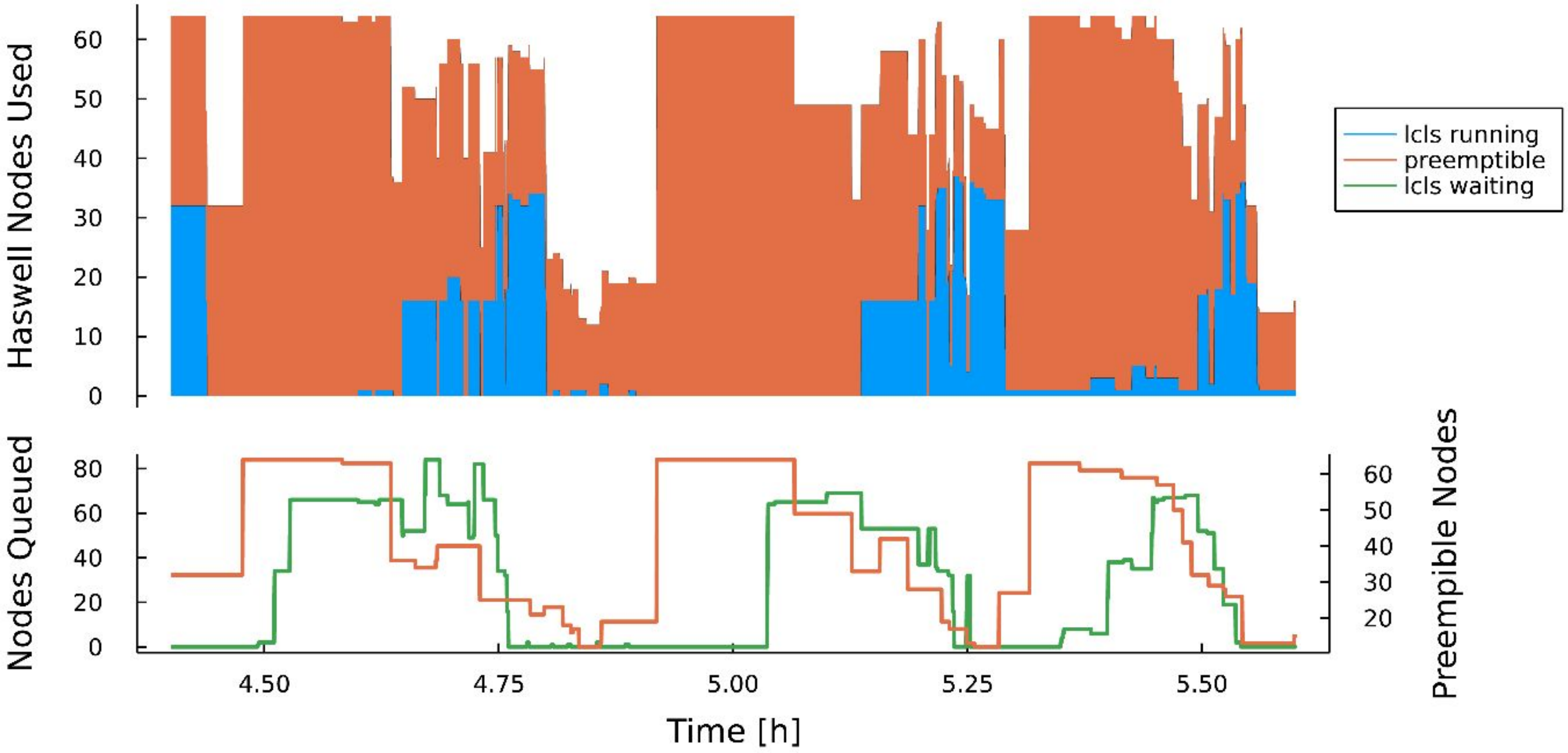
Preemptible Reservations

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

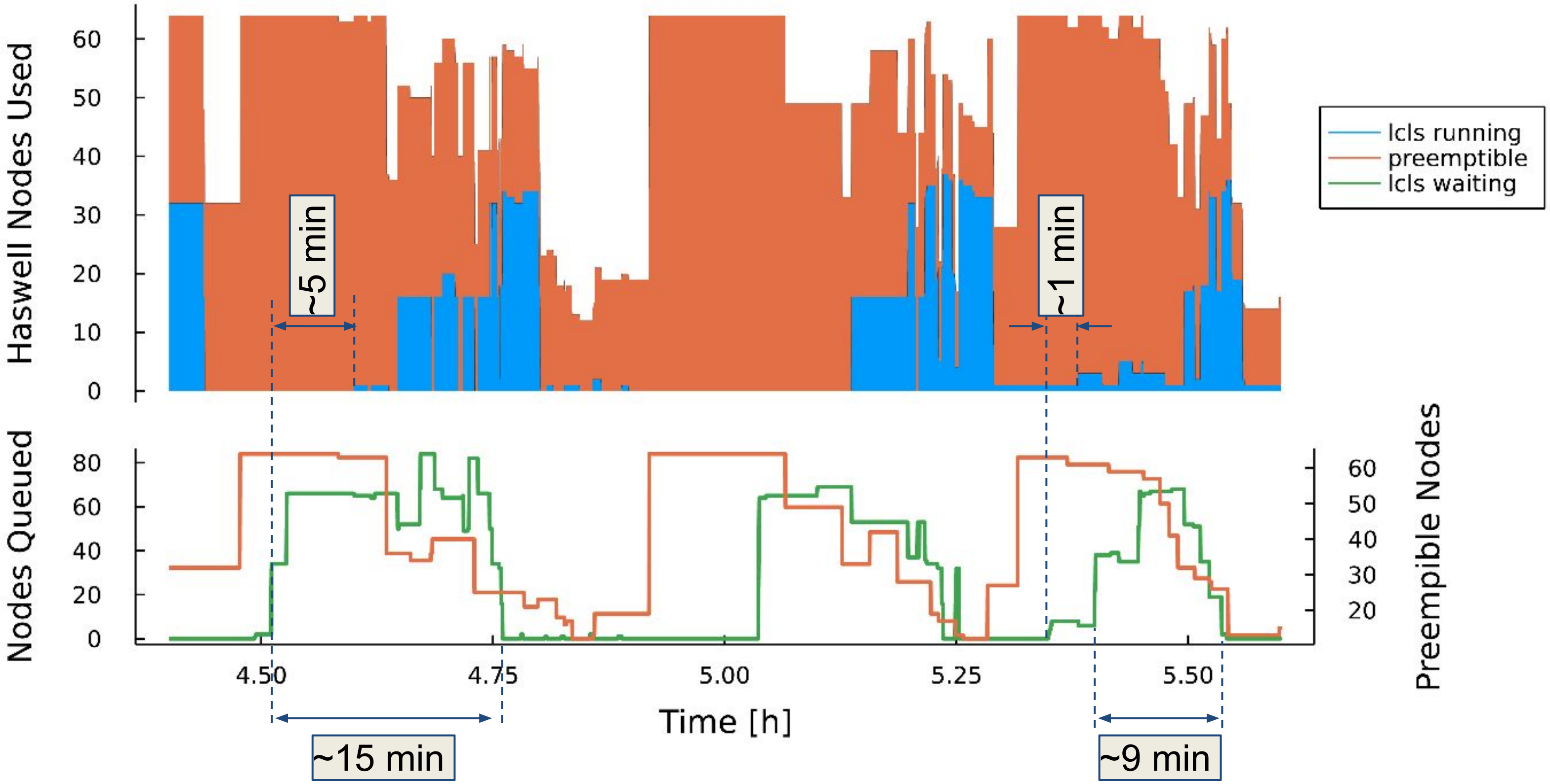
- Make a job preemptible
  - `#SBATCH --signal=R:INT@${SECONDS}`
    Eg. `#SBATCH --signal=R:INT@300`

- 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

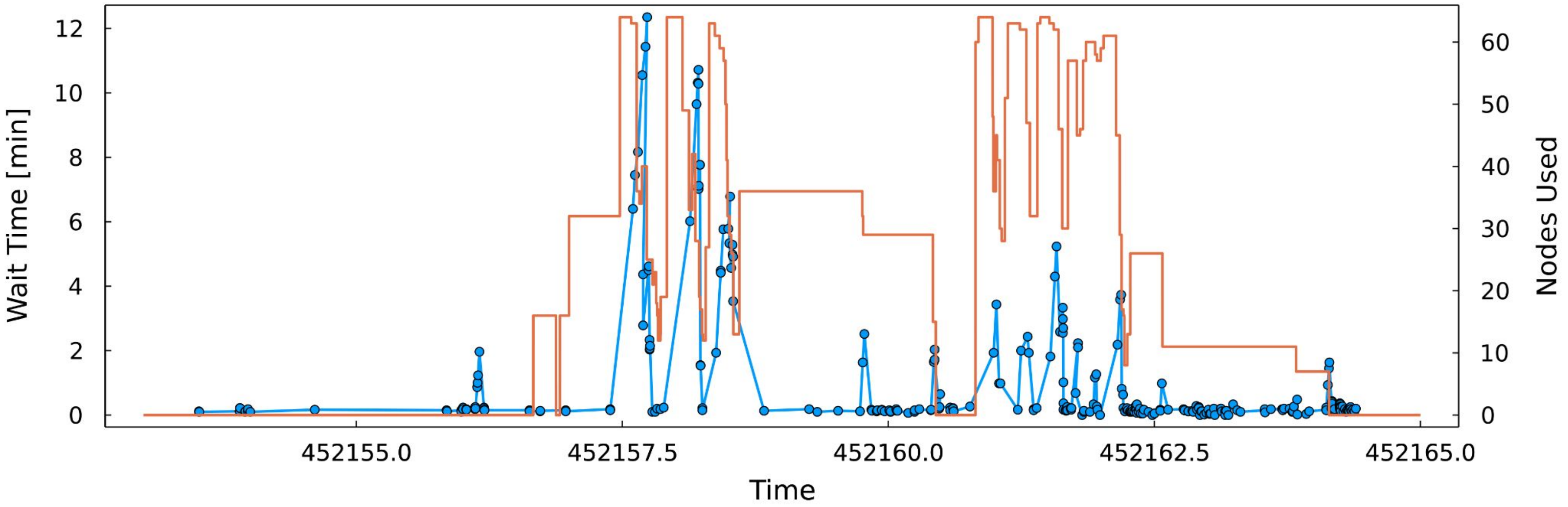


Testing Preemptible Reservations

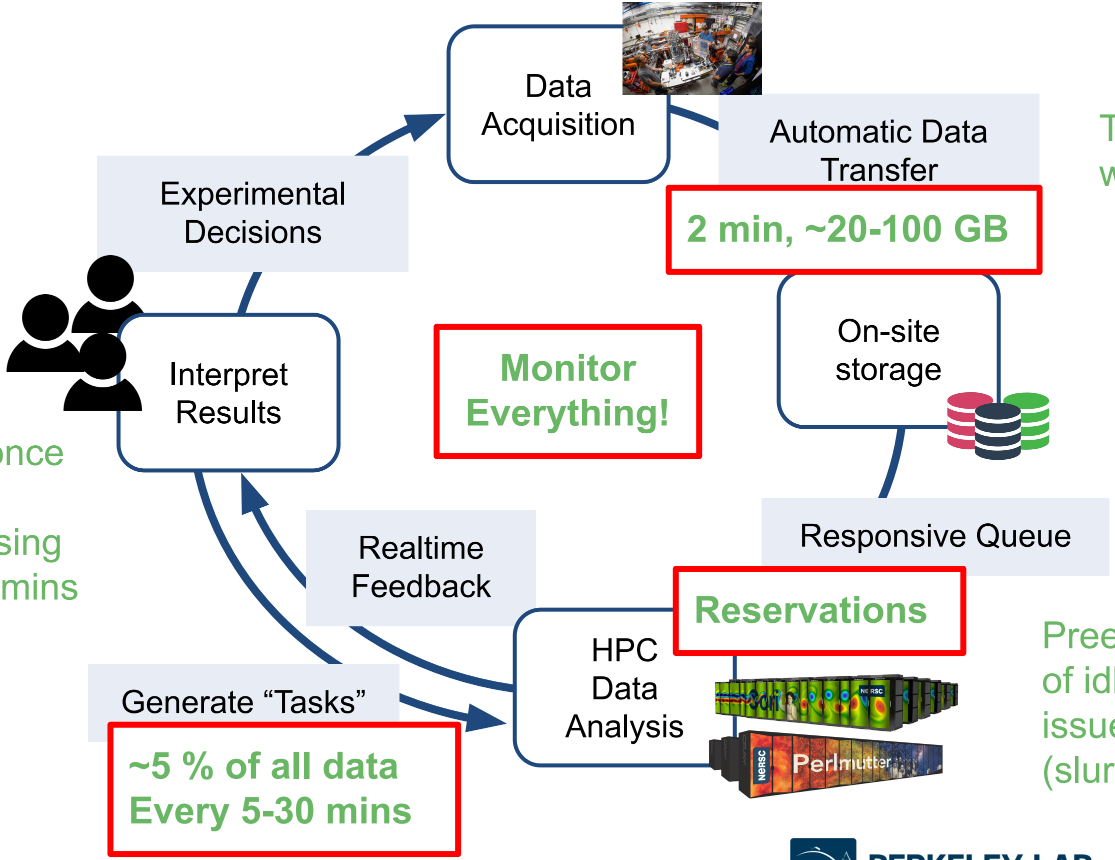


Testing Preemptible Reservations

- Preemptible jobs don't leave reservation quickly enough



Conclusion: XFEL + Superfacility



Tail latency can disrupt whole workflow

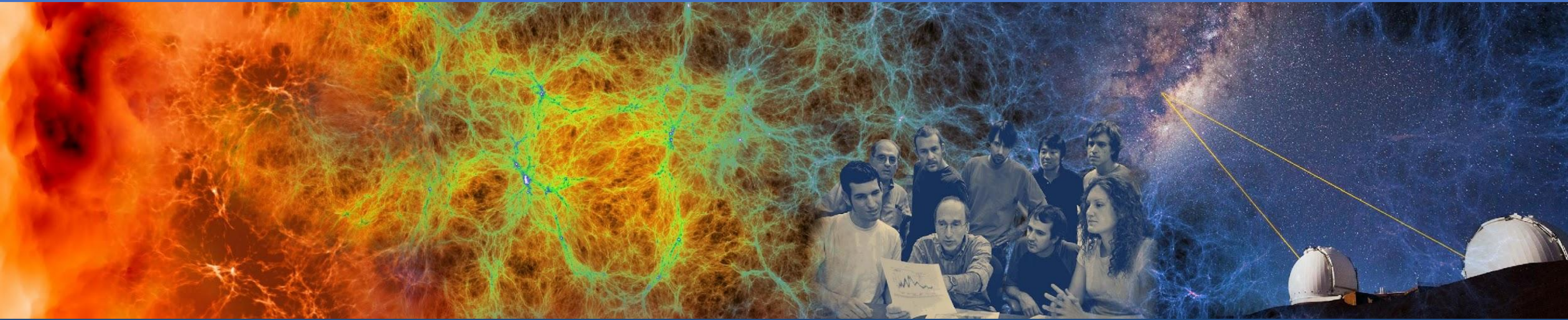
Most images processed only once

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

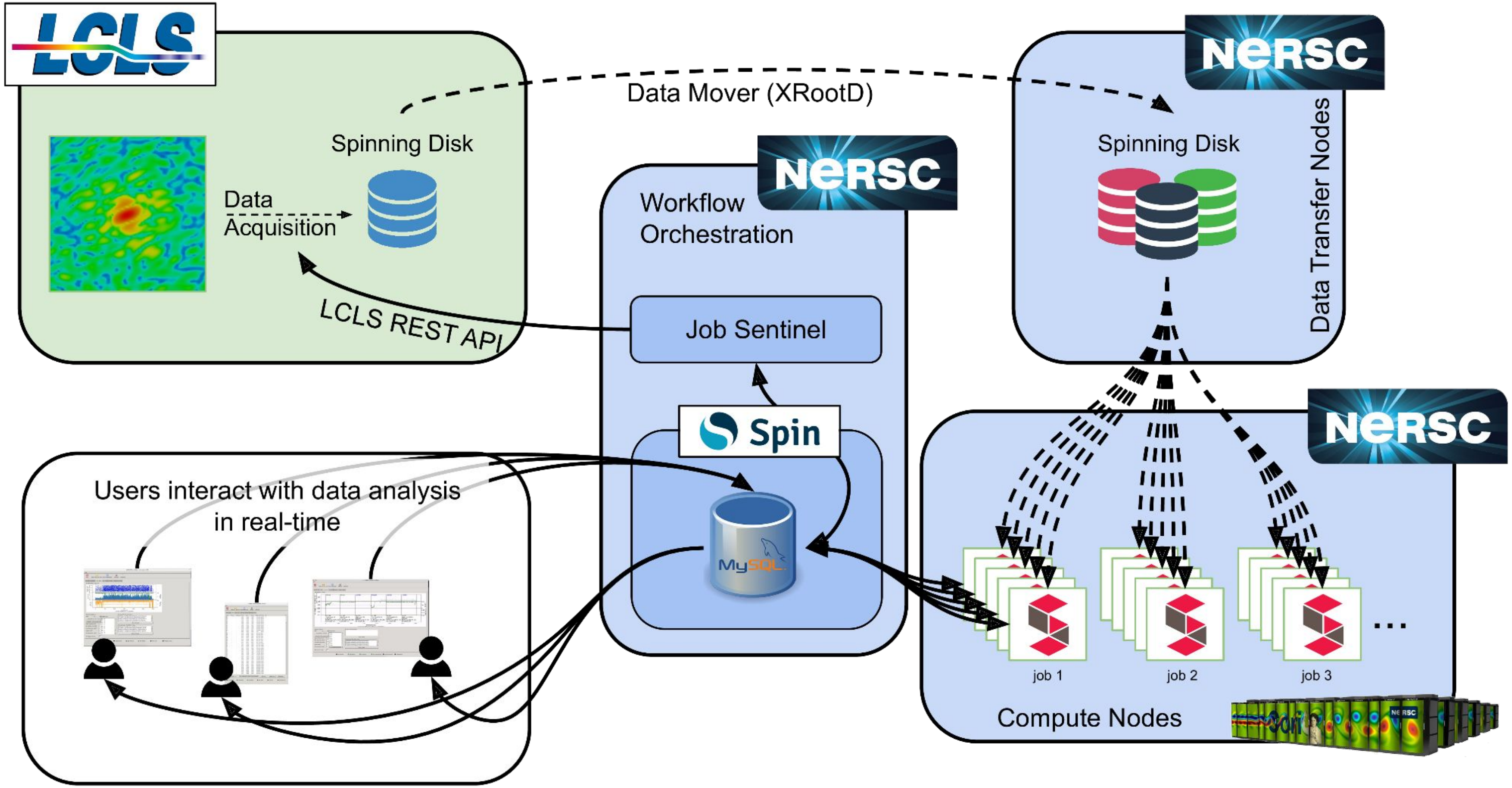


NERSC

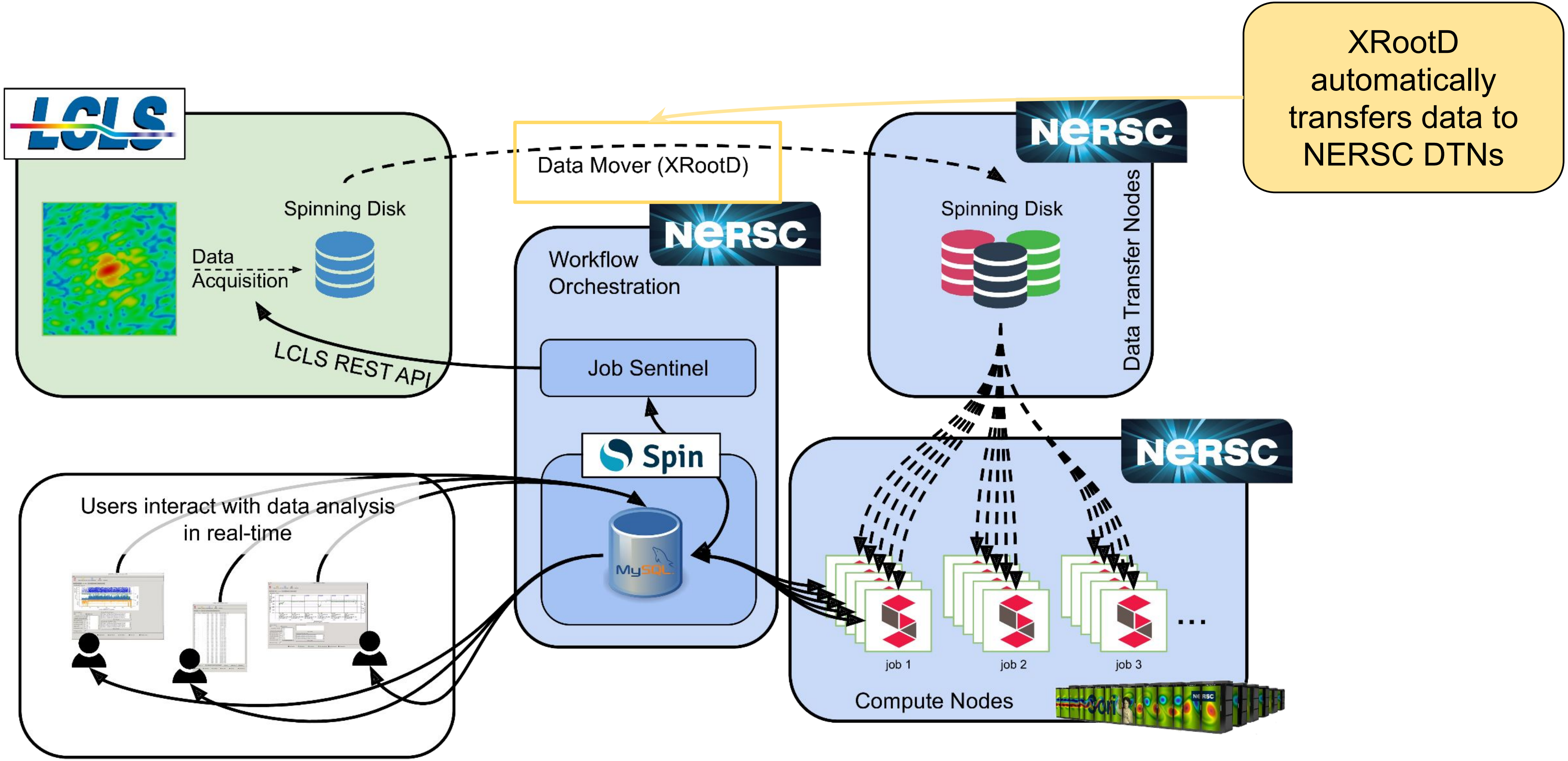
BERKELEY LAB

U.S. DEPARTMENT OF ENERGY | Office of Science

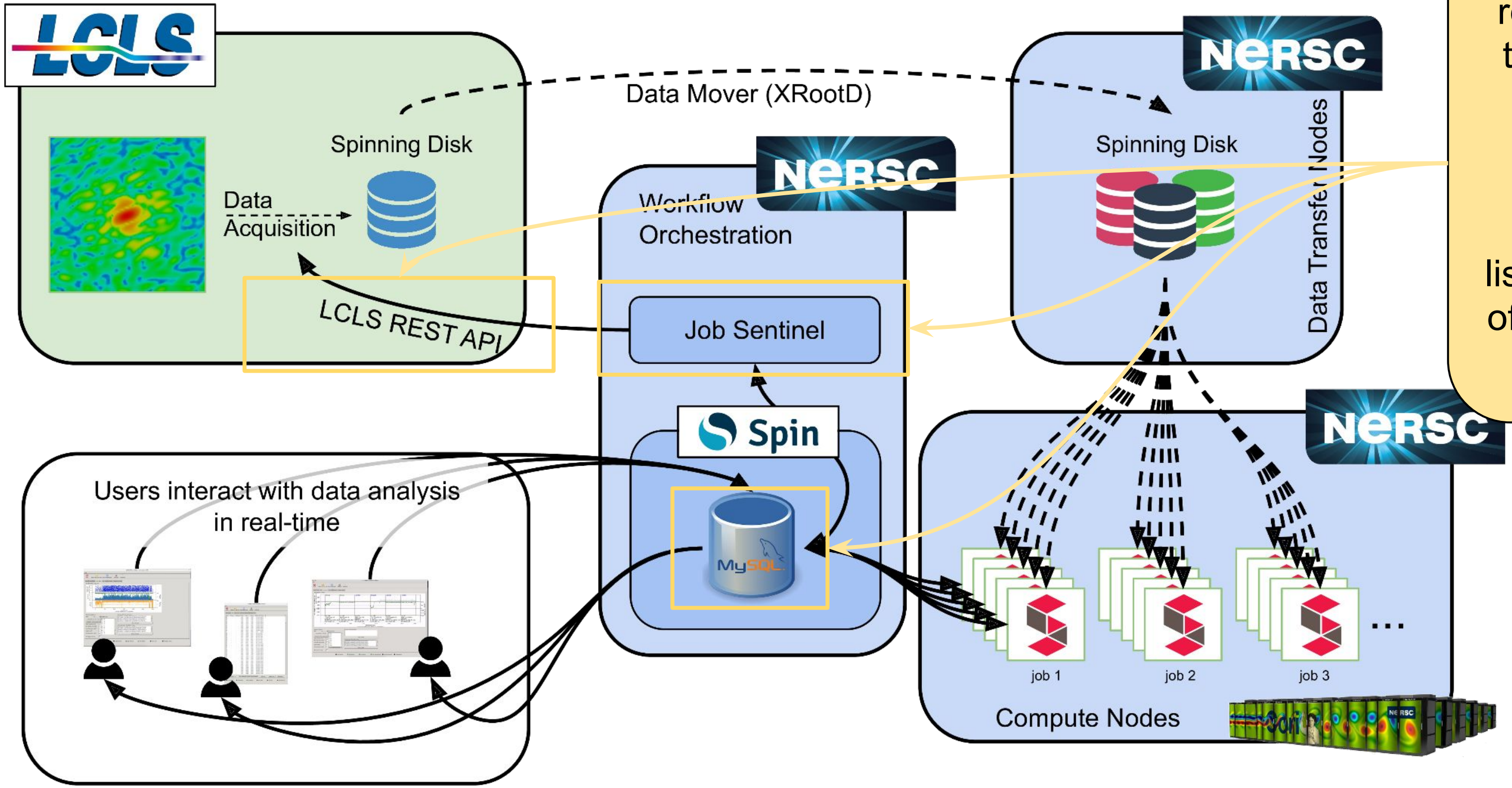
Deploying CCTBX at NERSC



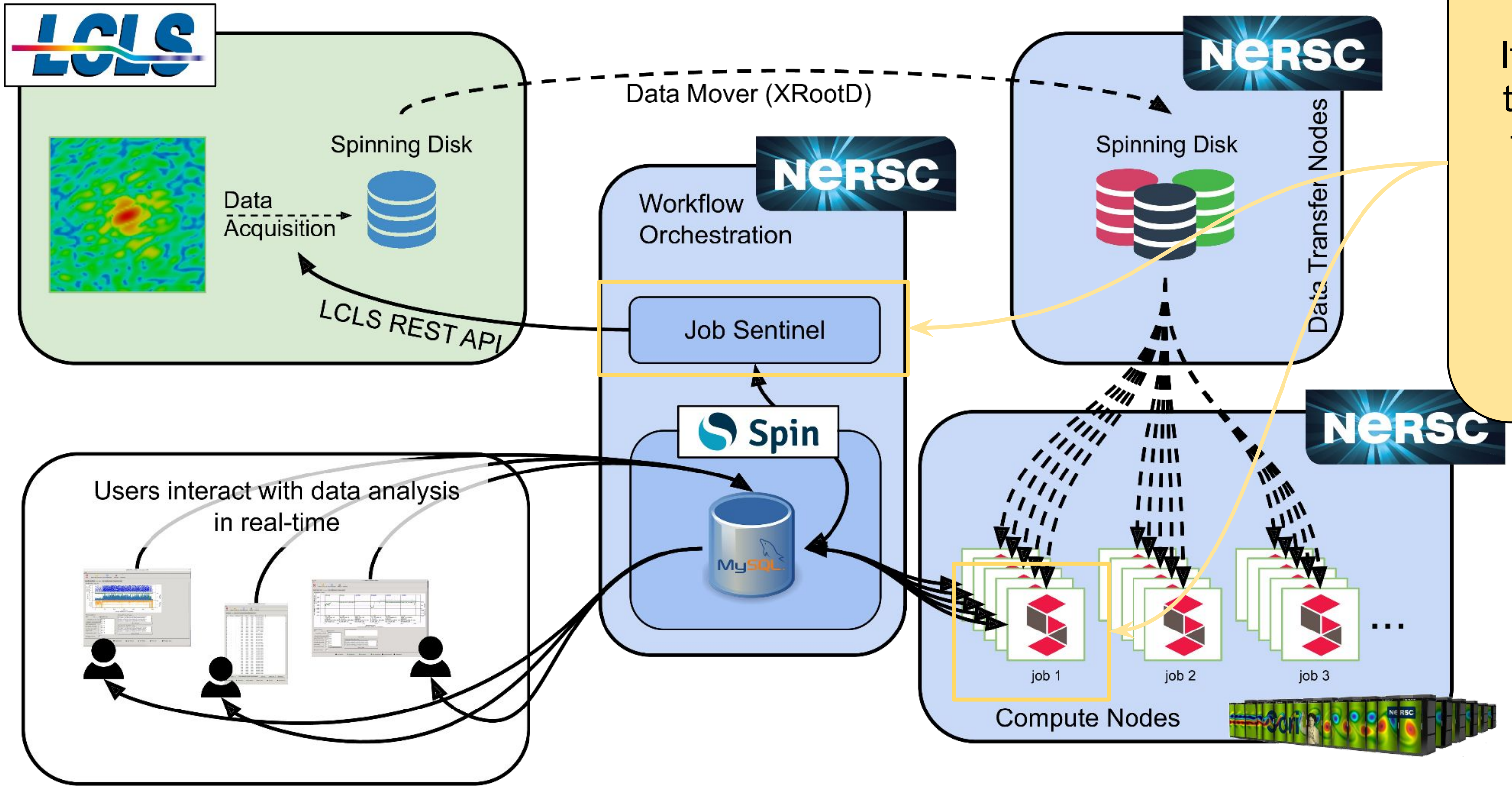
LCLS
Data Mover (XRootD)
Spinning Disk
Data
Acquisition
LCLS REST API
NERSC
Workflow
Orchestration
Job Sentinel
Spin
MySQL
Spinning Disk
Data Transfer Nodes
Users interact with data analysis
in real-time
job 1
job 2
job 3
...
Compute Nodes



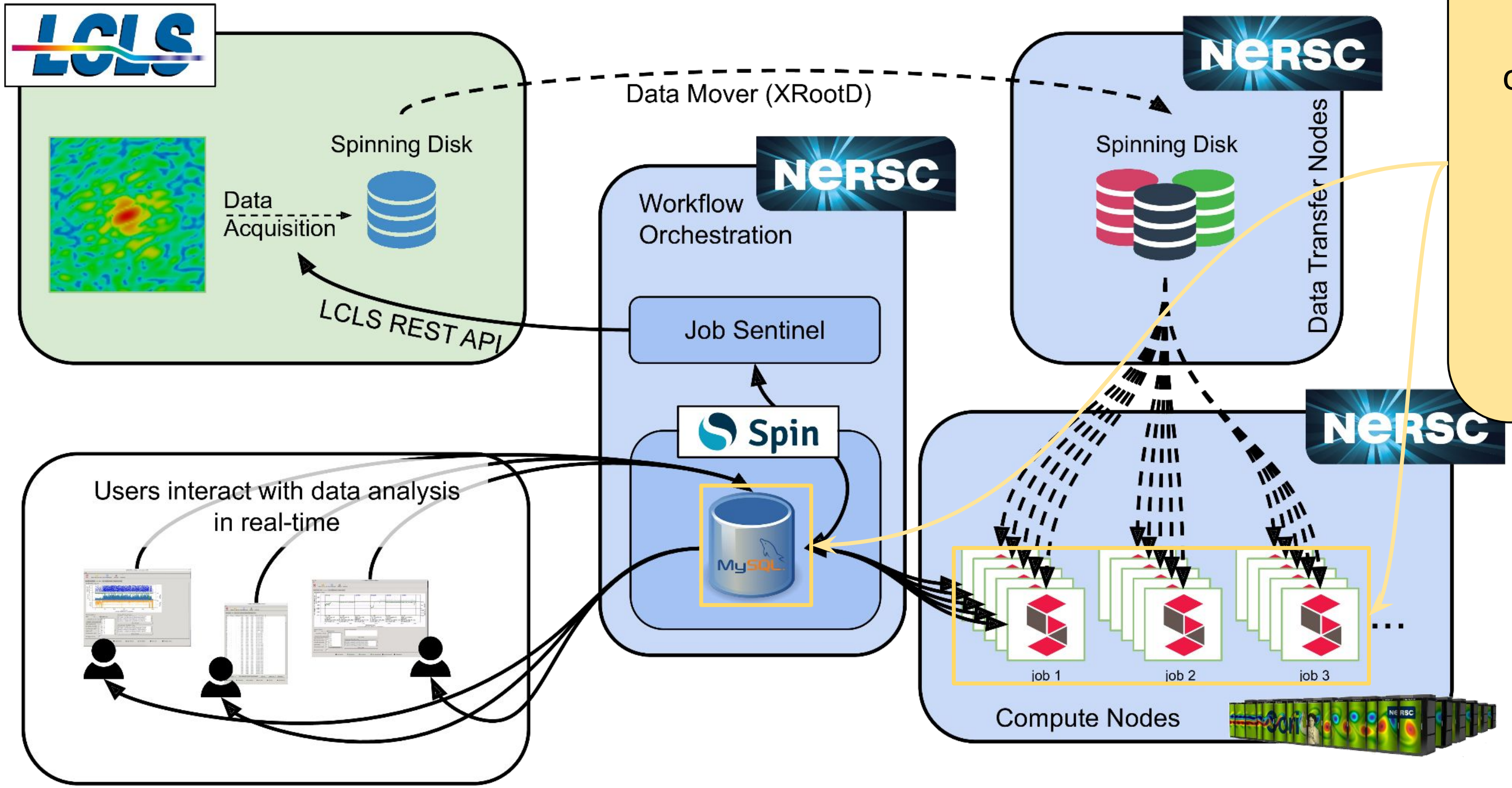
XRootD automatically transfers data to NERSC DTNs
Data Mover (XRootD)
Spinning Disk
Data Acquisition
LCLS REST API
Workflow Orchestration
Job Sentinel
Spin
MySQL
Spinning Disk
Data Transfer Nodes
job 1
job 2
job 3
...
Compute Nodes
Users interact with data analysis in real-time



Job sentinel regularly queries the LCLS REST API for new experiments. It then compares the experiment list with the record of completed data analyses.

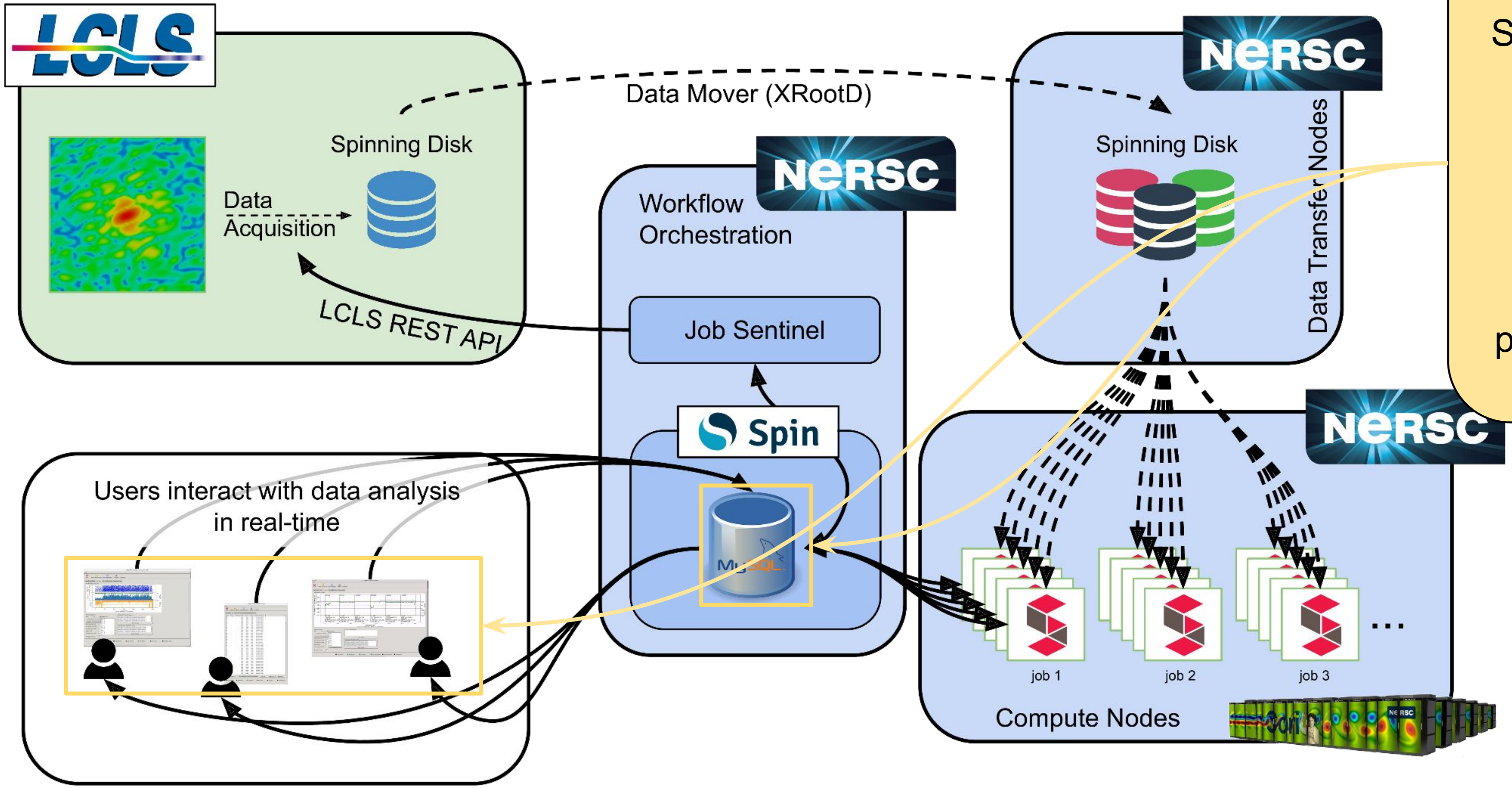


If new jobs need to be submitted, the job sentinel constructs and submits jobscripts.



Jobs run in Shifter / Podman containers on compute nodes.

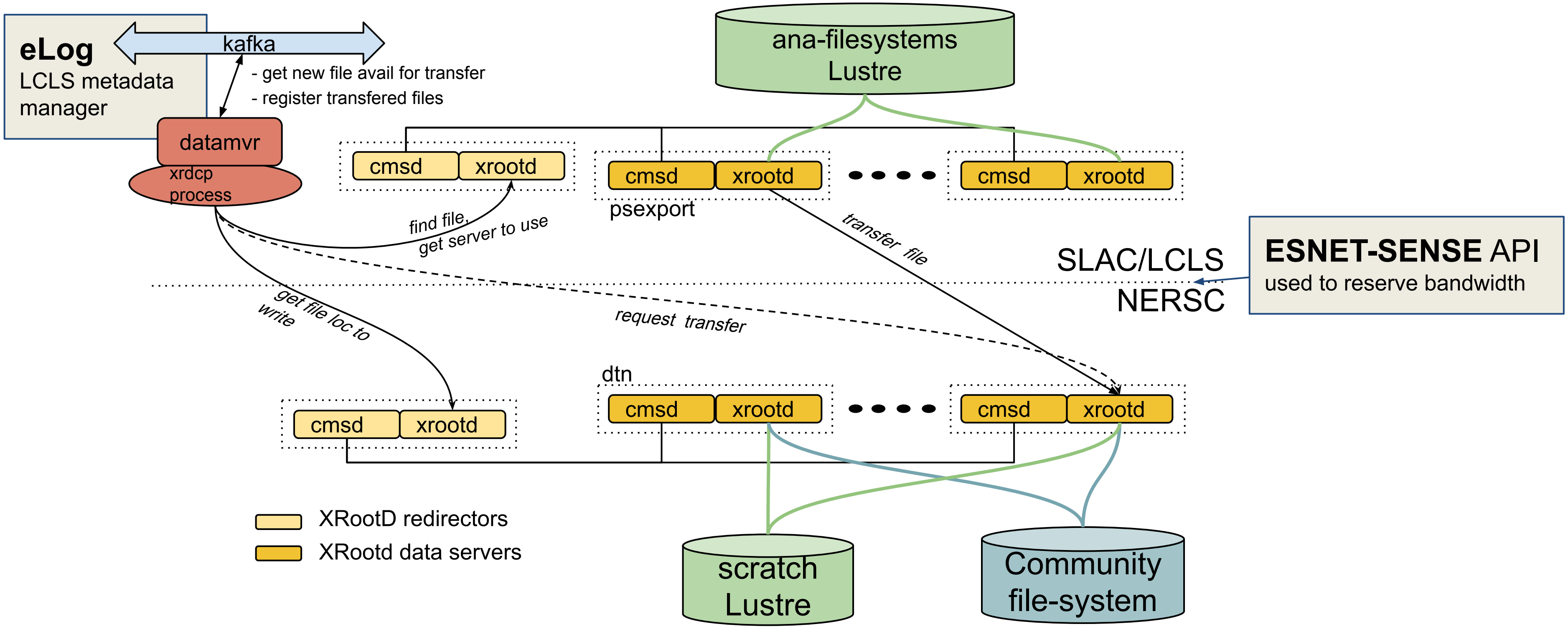
As data is analyzed, the progress is updated in real time to a SQL database



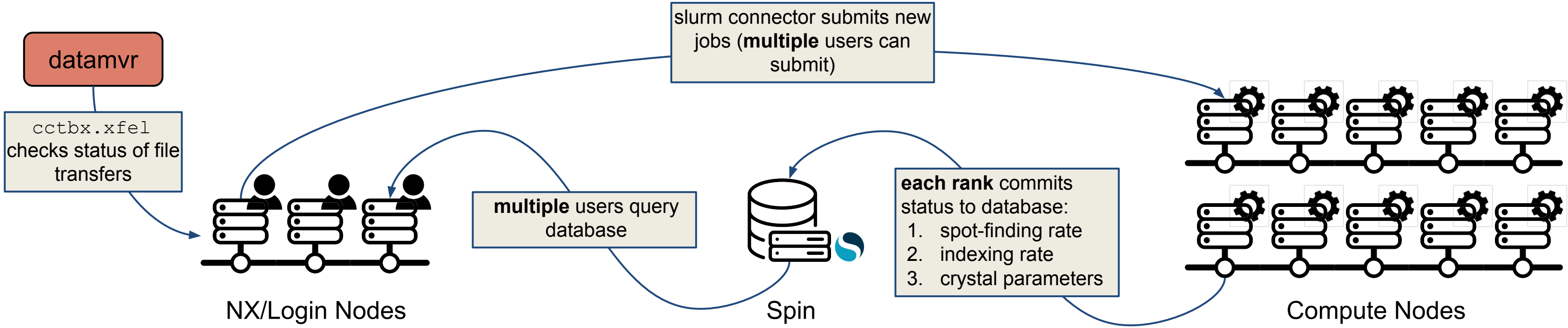
Users (AoE) can connect to the SQL database, to observe the progress of the data analysis.

Users also commit new parameters to be analyzed

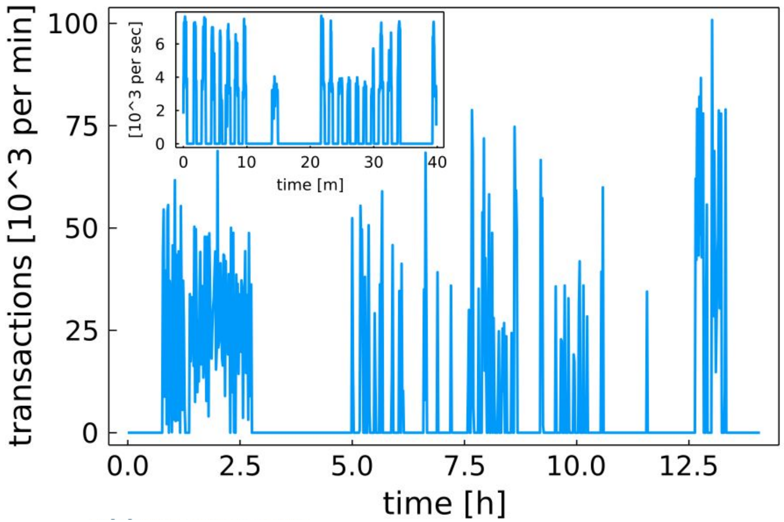
Data Movement XRootD clusters



Workflow Coordination hosted on Spin

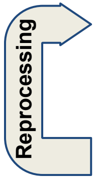


- Spin was capable of handling over 7k transactions/sec
- Transactions and connections need to be pooled when exceeding 4000 connections (ranks)

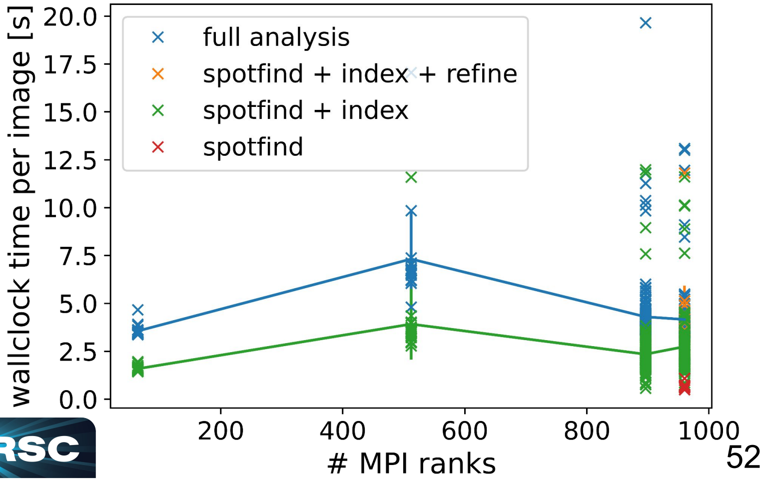


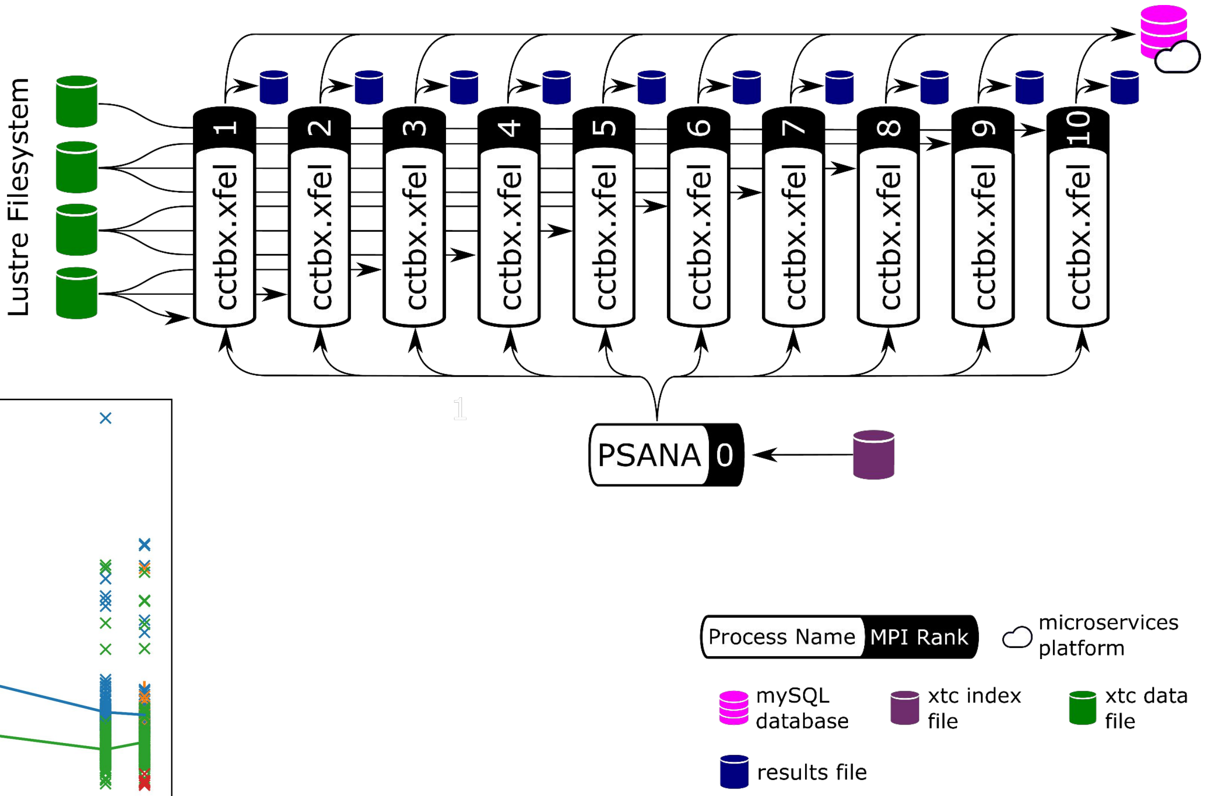
Data Analysis

- Data analysis follows sequential stages:



  - spotfinding
  - indexing Bragg spots
  - model refinement
  - integrating Bragg spots



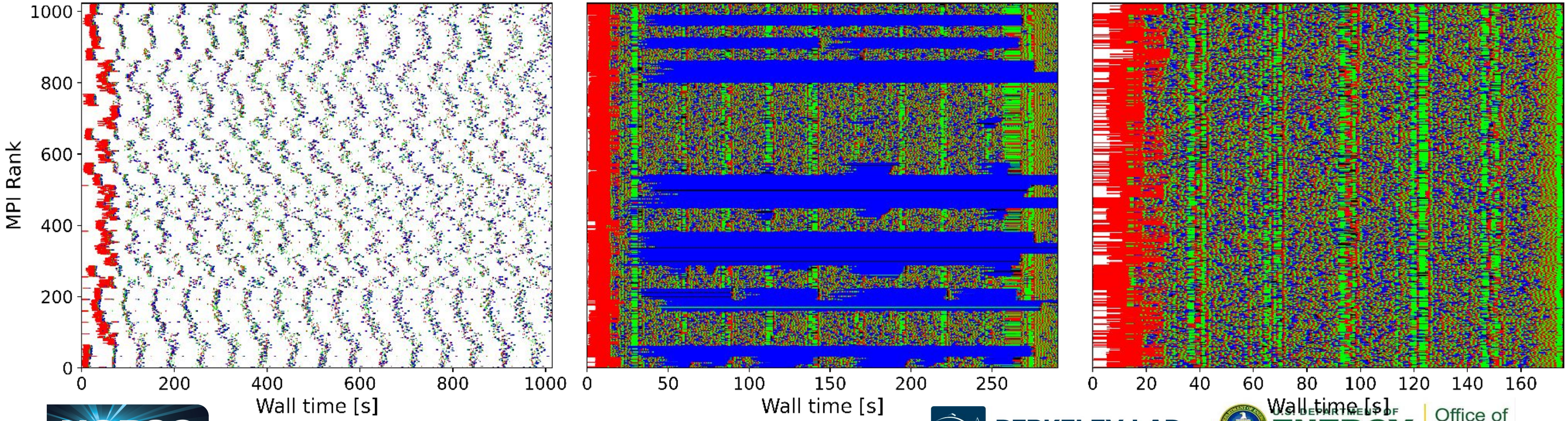


Performance Optimization For The Exascale

How's the Computation Weather Today?

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

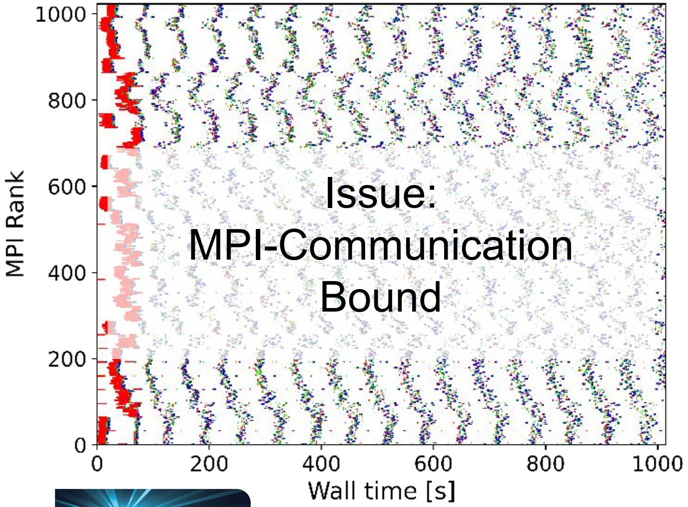
Start-Up and I/O (PSANA)
Spot Detection (DIALS)
Indexing (DIALS)
Refinement (DIALS)
Integrating (DIALS)

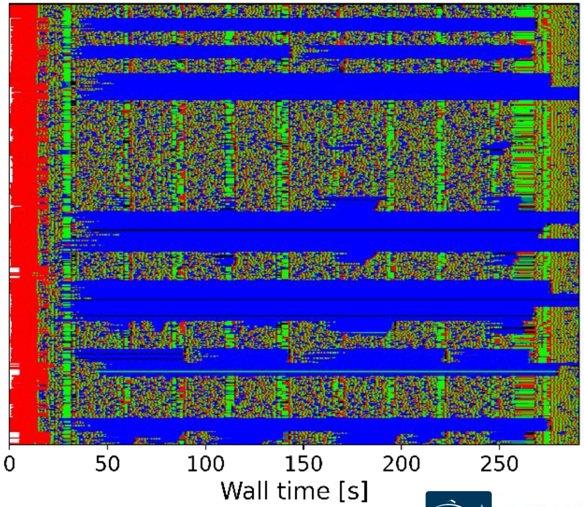


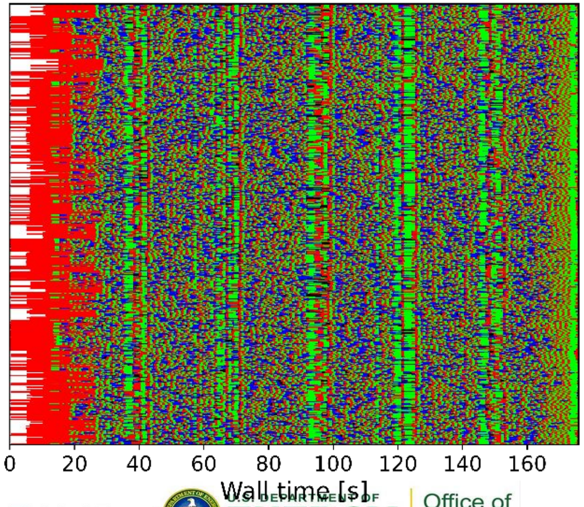
How’s the Computation Weather Today?

- 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)



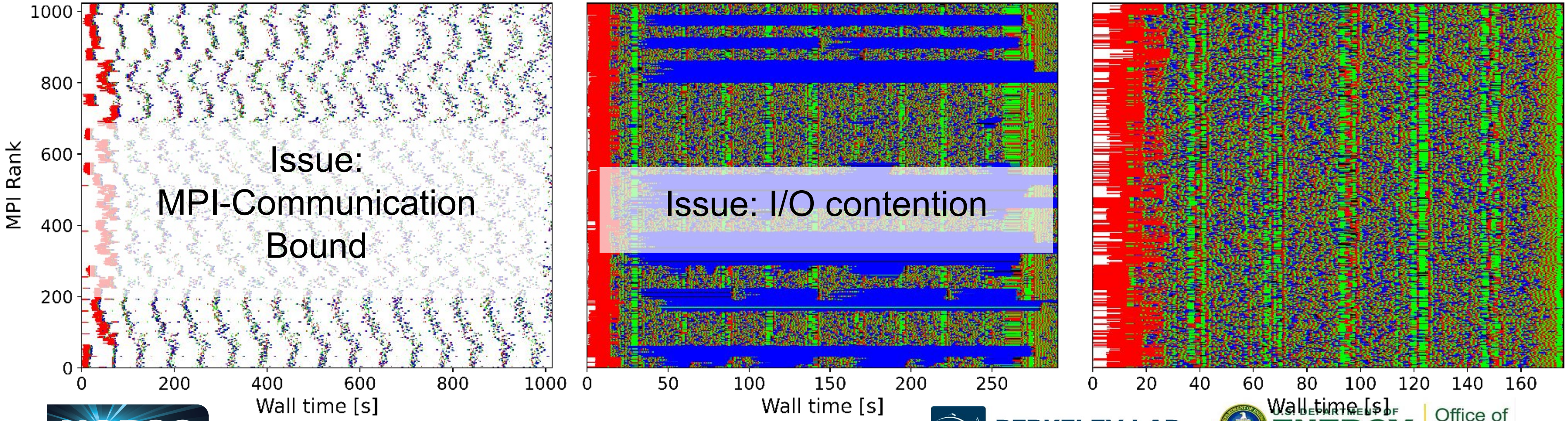




How’s the Computation Weather Today?

- 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)



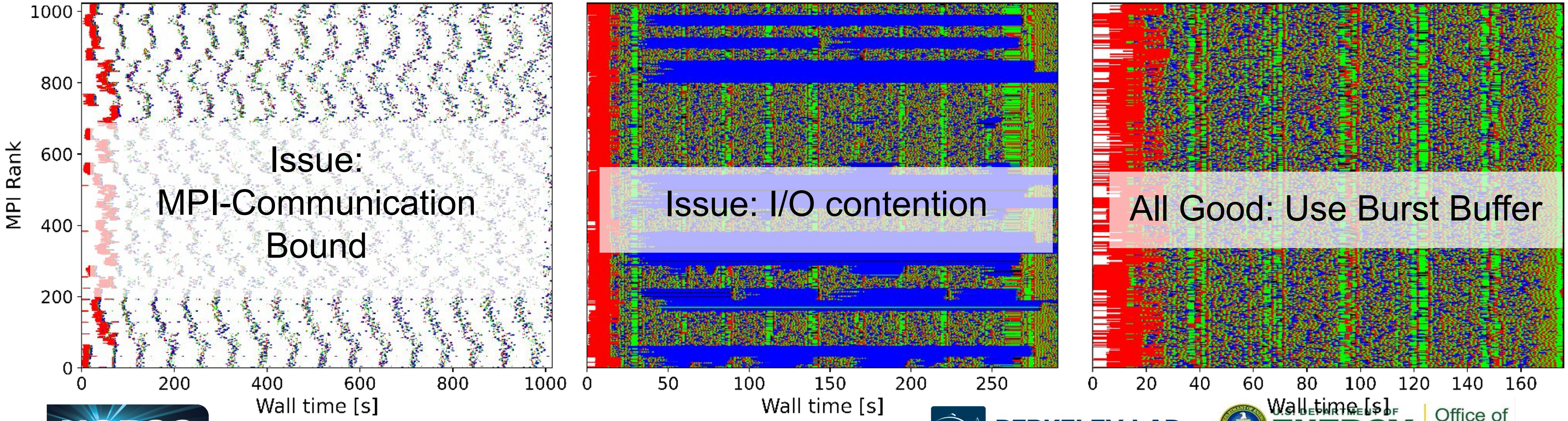
NERSC

BERKELEY LAB
Bringing Science Solutions to the World
U.S. DEPARTMENT OF ENERGY
Office of Science

How’s the Computation Weather Today?

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

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



Performance Portability

Work by Felix Wittwer

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

add_array.cu

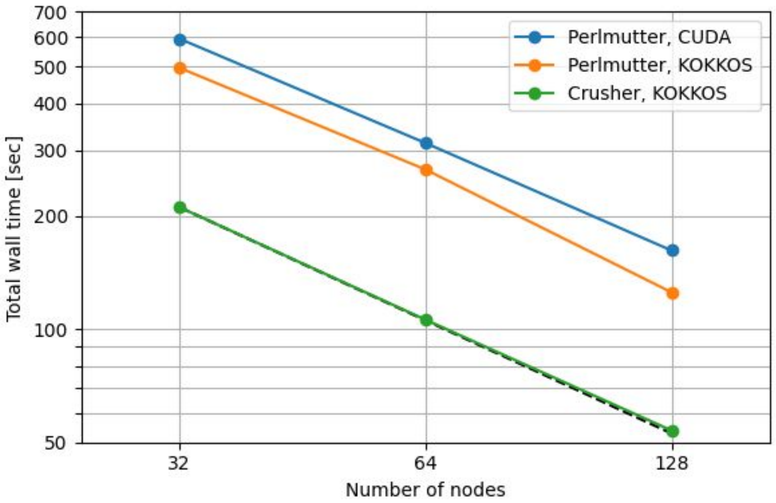
```
__global__
void addArray(double* lhs,
              float* rhs,
              int total_pixels) {
    int j = blockDim.x * blockIdx.x + threadIdx.x;
    if (j < total_pixels) {
        lhs[j] = lhs[j] + (double) rhs[j];
    }
}
```

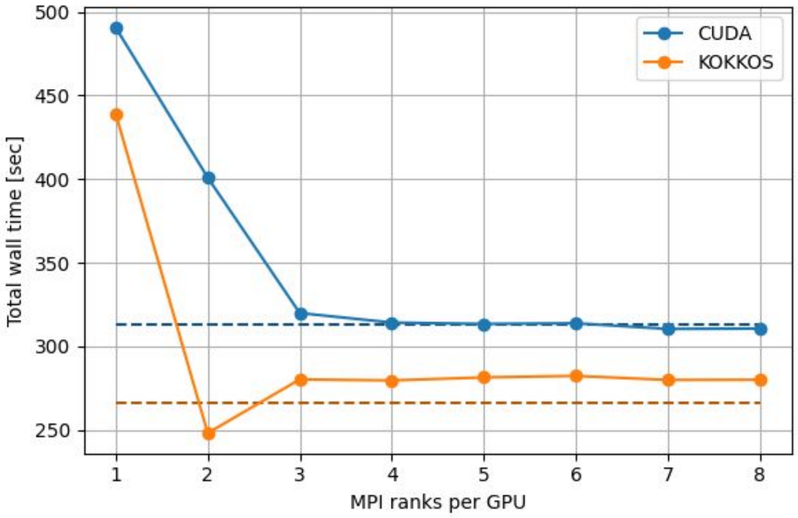
add_array.cpp

```
void addArray(Kokkos::View<double*> lhs,
              Kokkos::View<float*> rhs,
              int total_pixels) {
    Kokkos::parallel_for(
        "addArray", total_pixels,
        KOKKOS_LAMBDA (const int& j) {
            lhs(j) = lhs(j) + (double) rhs(j);
        }
    );
}
```

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

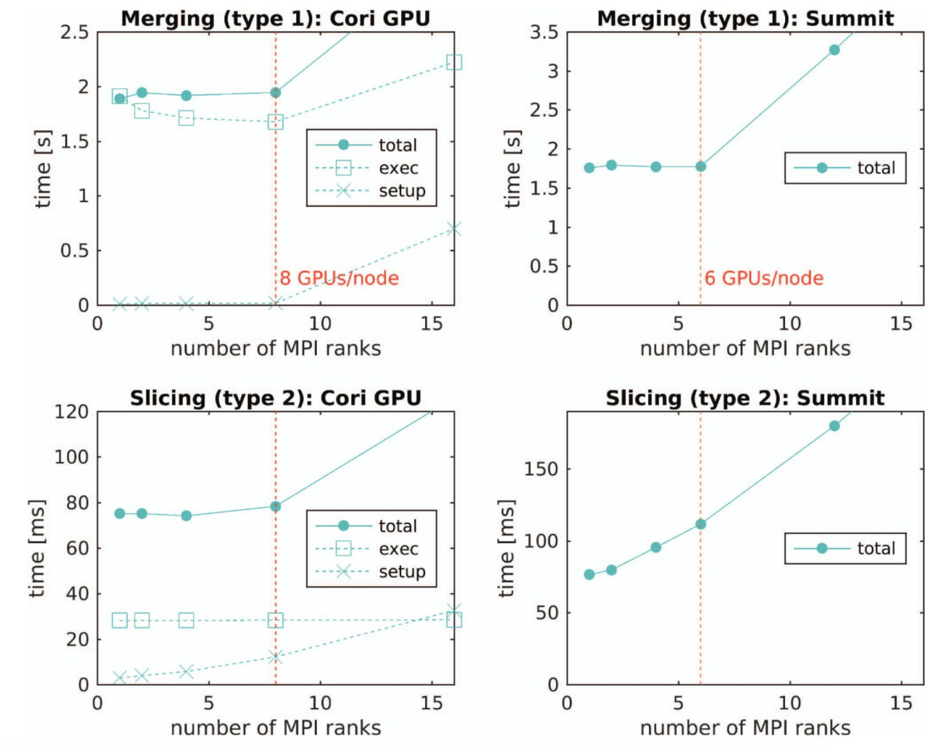




Special Issue of the 2021/2022 Cray User Group Conference (CUG 2021/2022), Concurrency and Computation: Practice and Experience

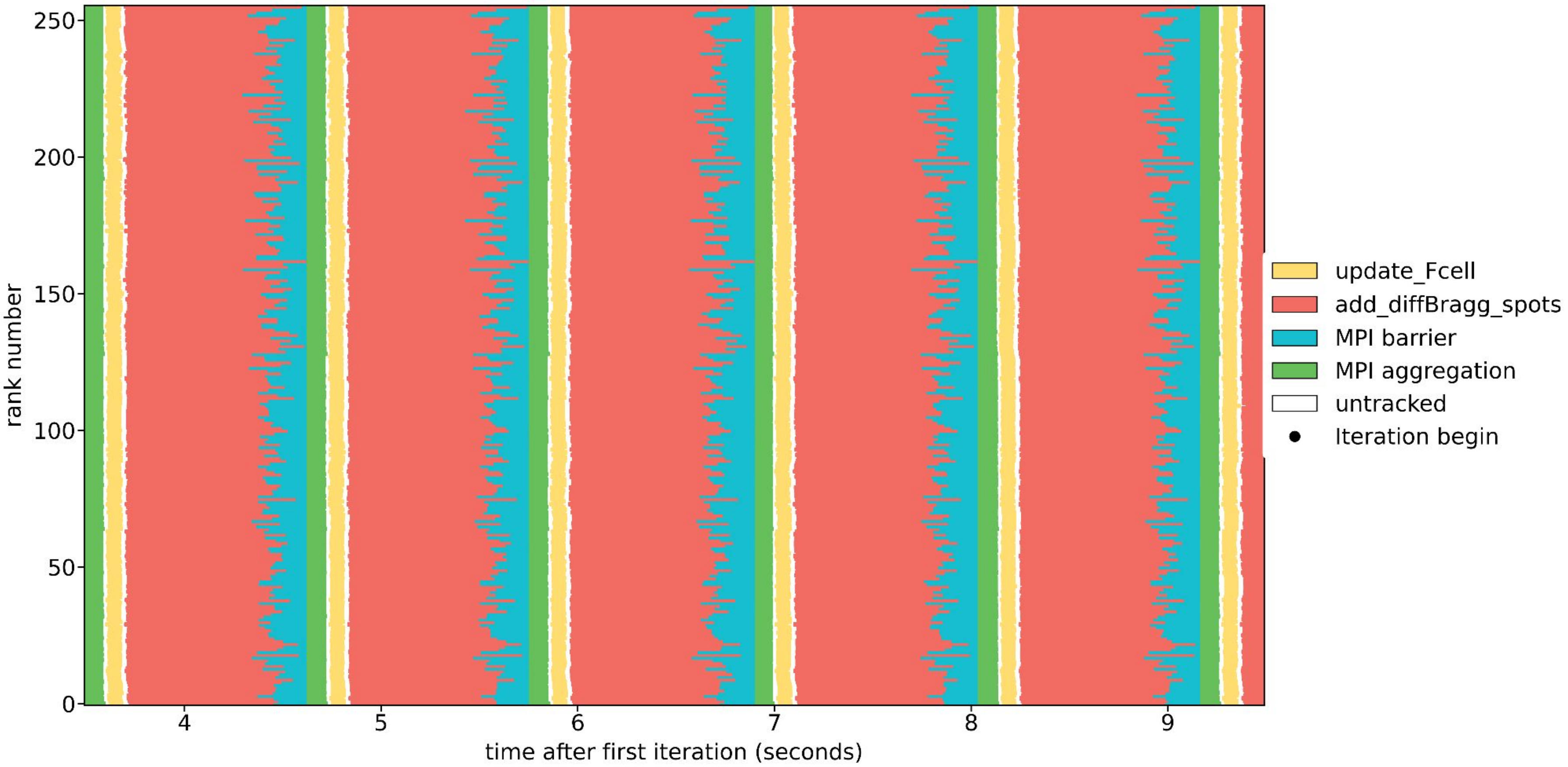
Vendors + Libraries

- 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/JBlaschke/PybindGPU

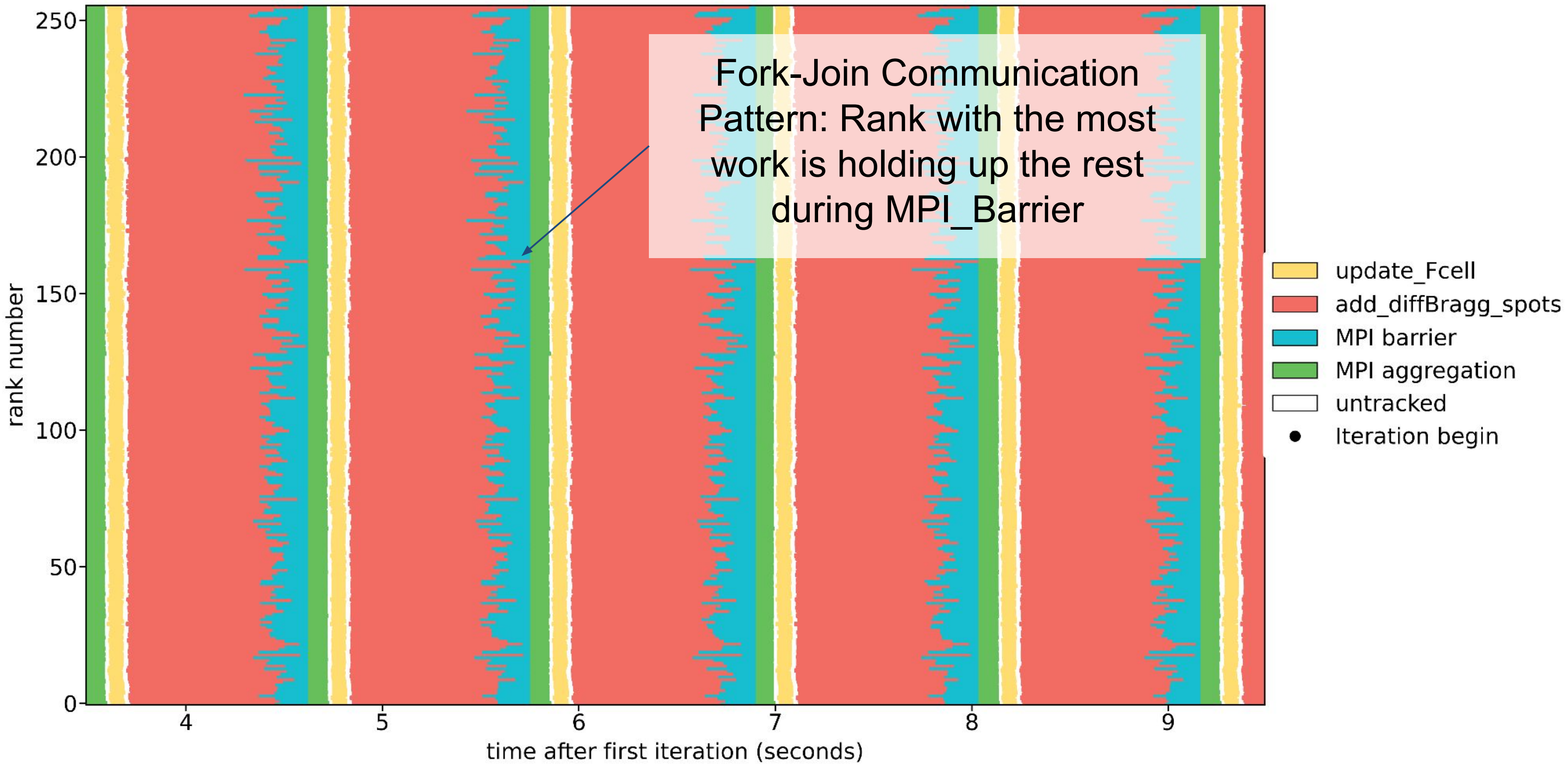


https://arxiv.org/abs/2102.08463

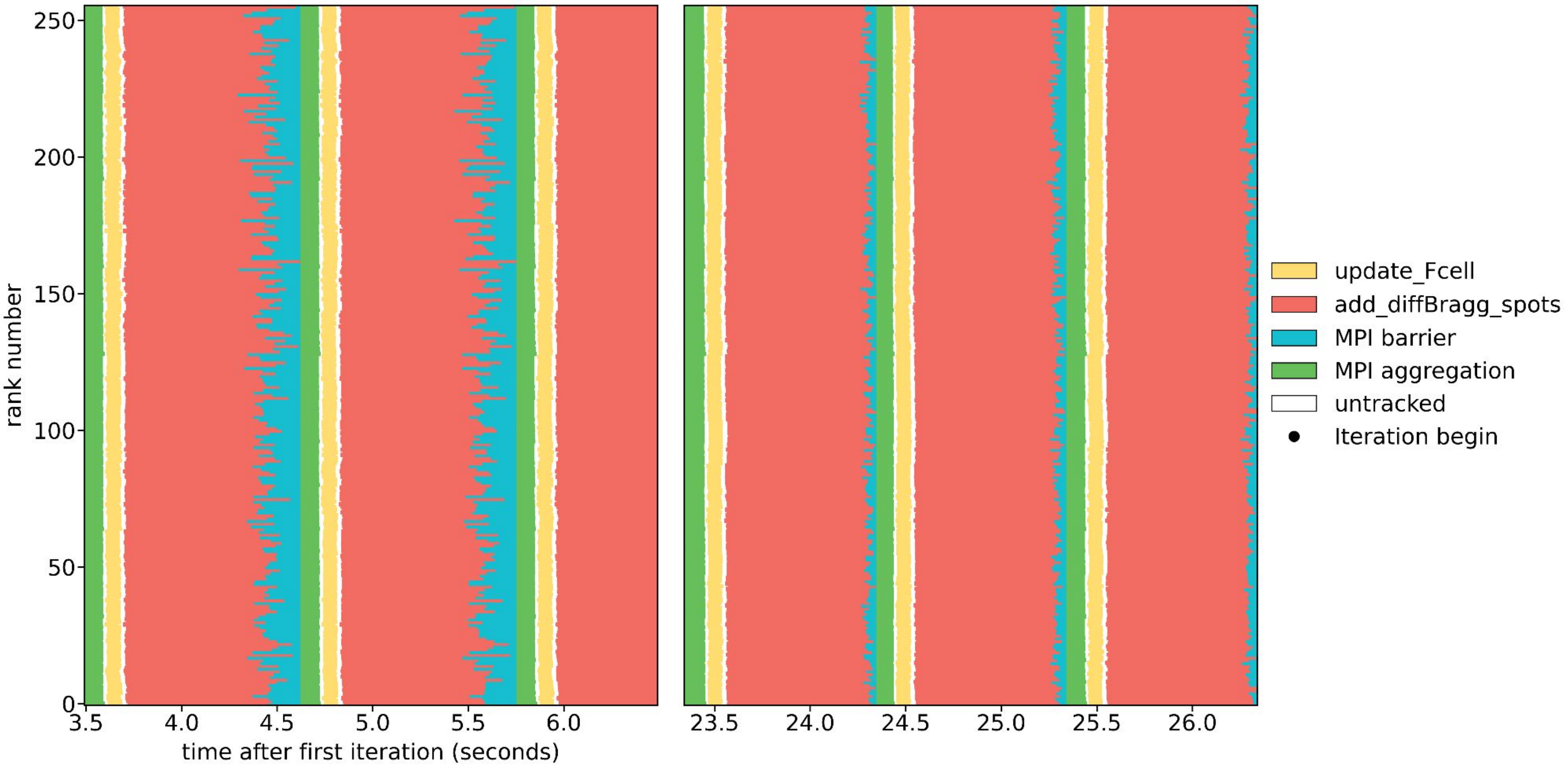
Load Balancing



Load Balancing



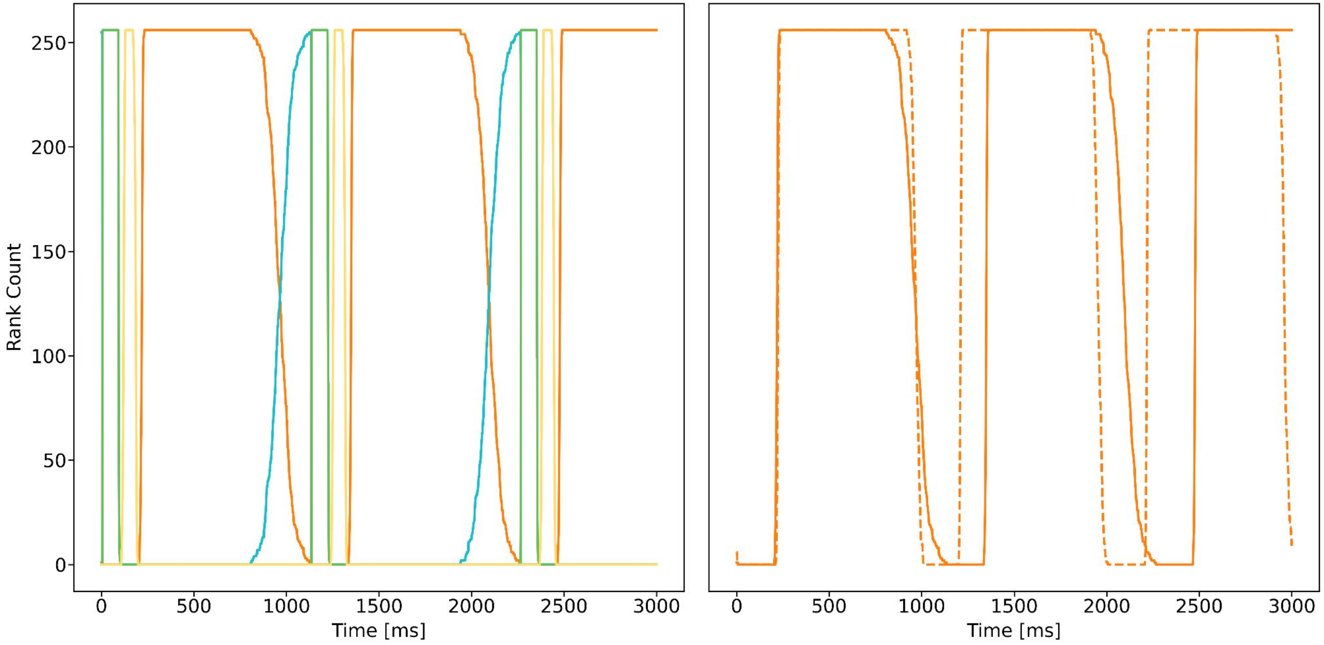
Load Balancing

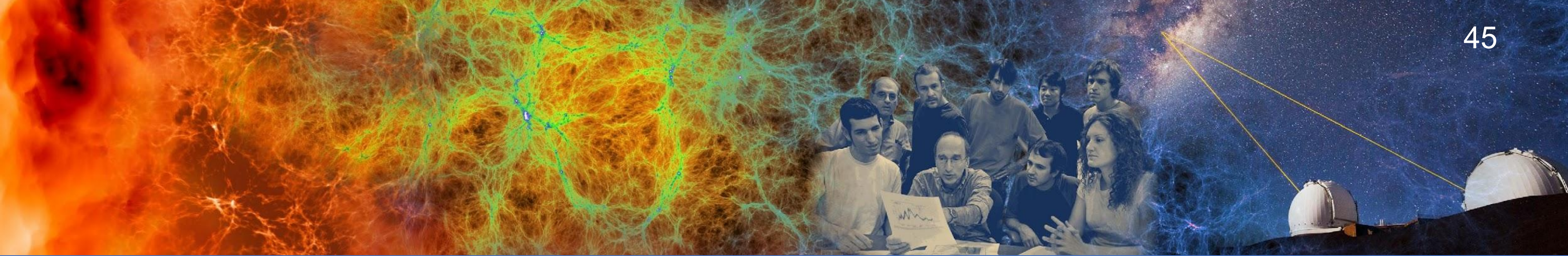


Load Balancing

- 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
  - 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/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

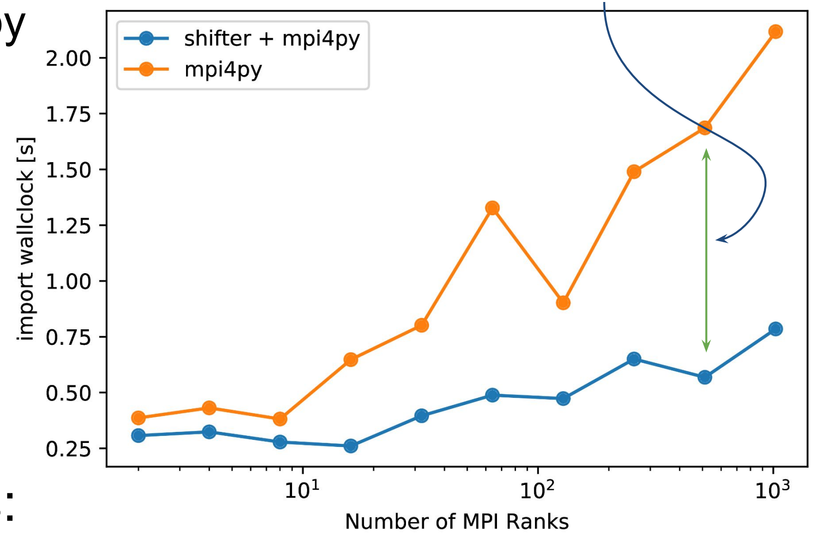
Planning: Workflow Coordination

- New class of workflows include multiple data processing steps - Central **scalable** management and persistent state becomes necessary
  - Often hosted on login nodes ⇒ scaling becomes an issue
  - CCTBX: MySQL database for logging process state (completed, running, waiting)
  - Others: MongoDB (fireworks), or sometimes just writing to file
- Collaborative workspaces to facilitate dataset access and analysis
  - 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**)
  - Data centers can customize these to account for their system needs

Packaging: Containers

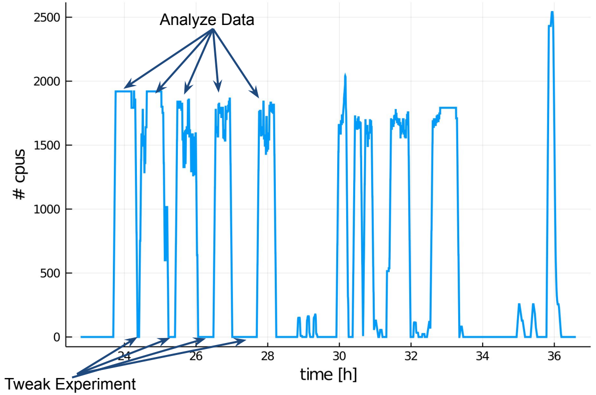
- Design images to be portable:
  - Can Streamline cross-facility workflows by standardizing image build process
  - Optimize dependency handling, data analysis workflows tend to have complex dependencies
  - Shifter / Podman mounts NERSC-specific libraries into image
  - **Isolate Anaconda’s conflicts with OS**
- Significantly lower import/loading times:
  - Image cached to node-local storage
  - Improved scalability even for few MPI Ranks

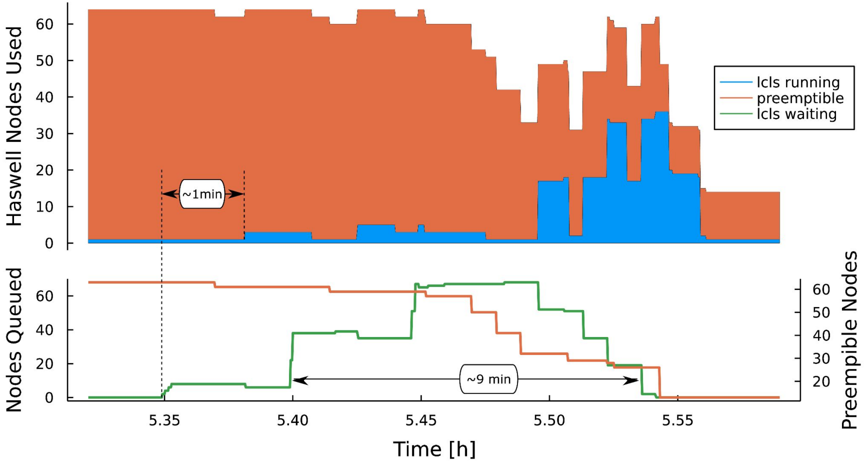
Dynamic linker’s symbol resolution uses node-local storage ⇒ many ranks do not flood file-system.



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 (eg. malleable jobs)



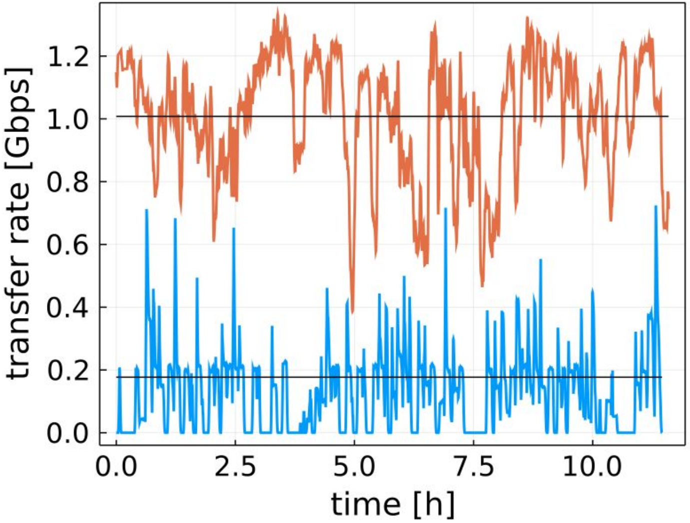


Packets: Network and I/O

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

- I/O Optimization is Crucial:
  - Optimize software (eg. 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**



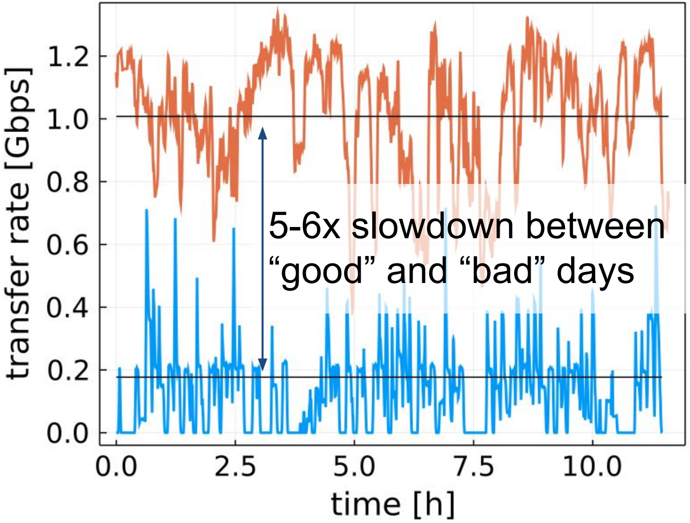
- Future Work:
  - Explore in-memory burst buffers, eg https://github.com/LLNL/UnifyFS
  - Offer more fine-grained control over which data is transferred and when
  - Better monitoring and alerting

Packets: Network and I/O

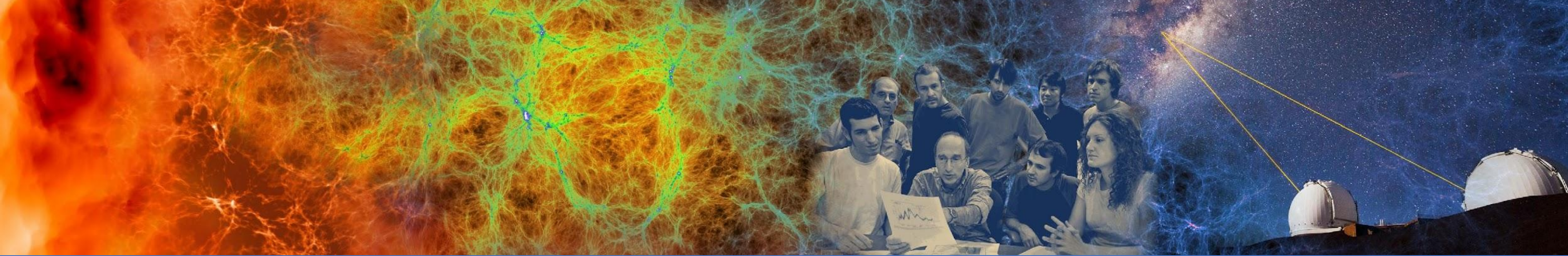
- In data analysis workflows, file systems and network can become bottlenecks

- I/O Optimization is Crucial:
  - Optimize software (eg. 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**

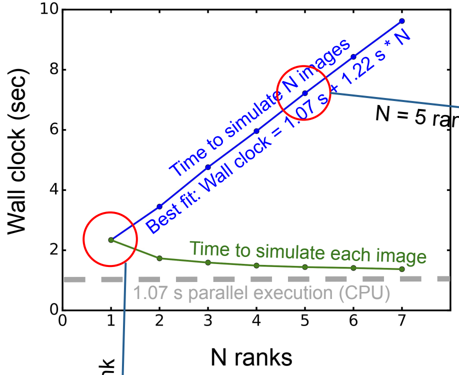


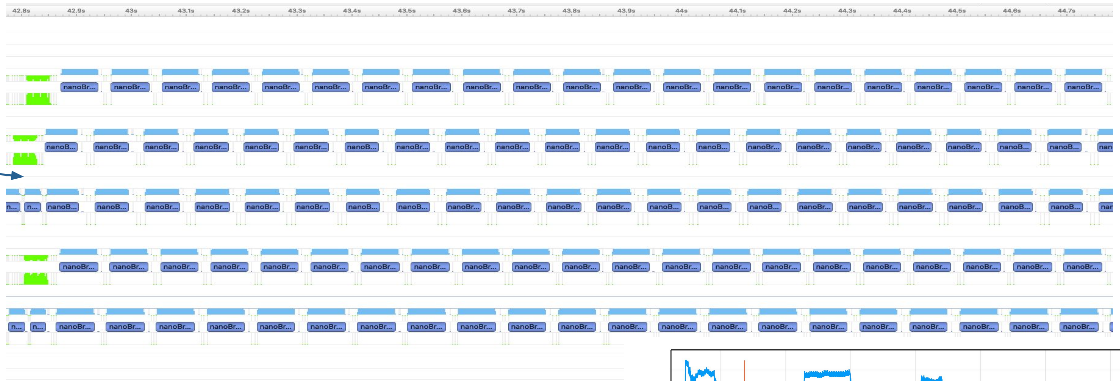
- Future Work:
  - Explore in-memory burst buffers, eg https://github.com/LLNL/UnifyFS
  - Offer more fine-grained control over which data is transferred and when
  - Better monitoring and alerting



Extra Bits

Sharing GPUs Between MPI Ranks

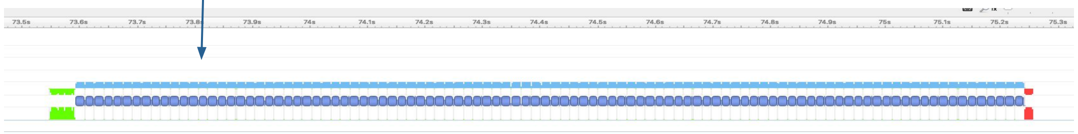


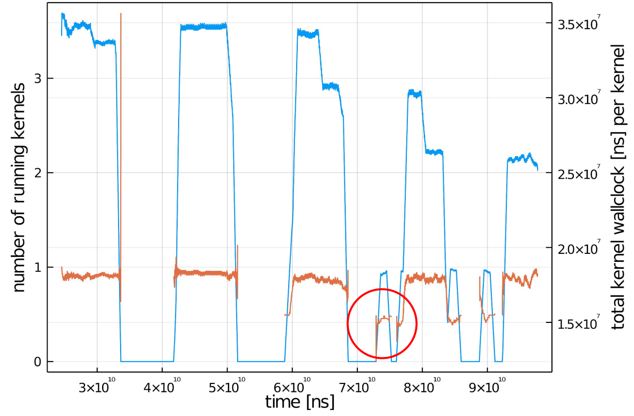


Host to device memcpy

Kernel execution

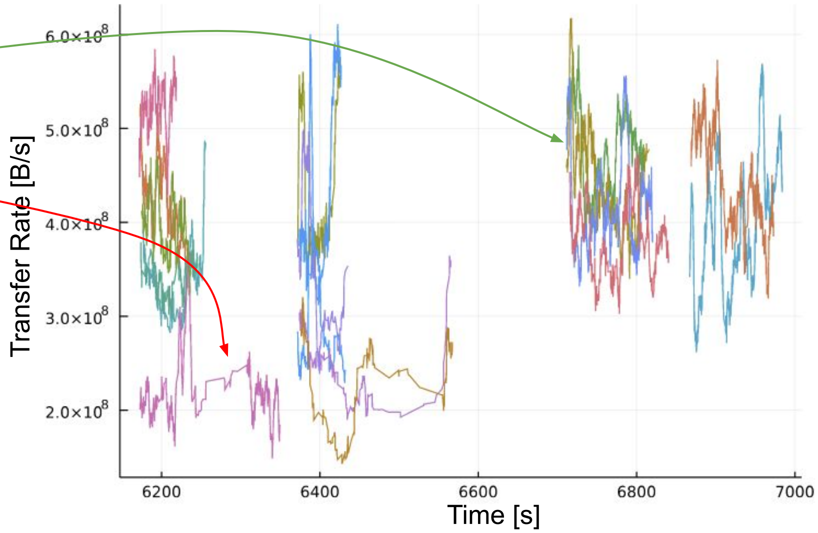
100 ms





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

workflow.py

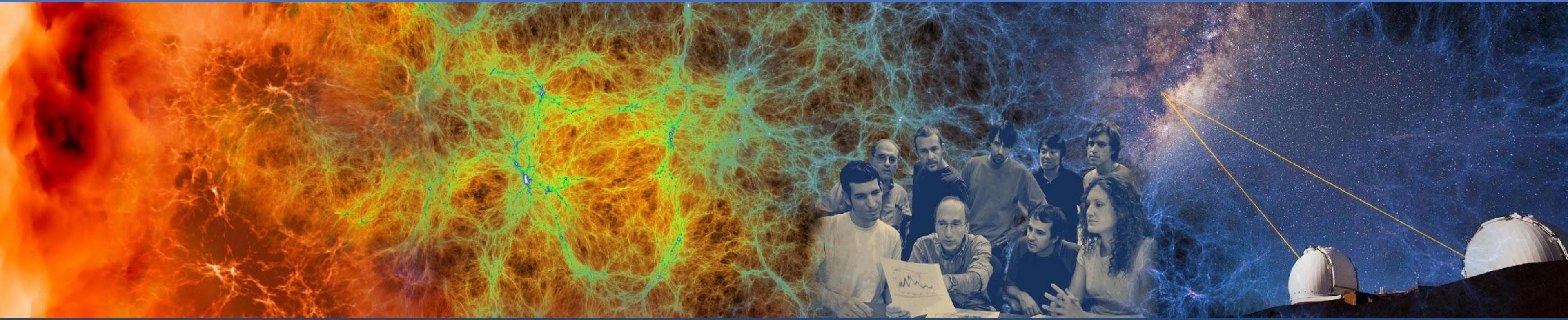
```
qadapter_1 = CommonAdapter(
  q_type = "SLURM",
  rocket_launch=f"rlaunch --json -l '{lp_str}' -w '{fw_str_1}' singleshot",
  constraint="gpu",
  account="nstaff",
  walltime="'00:02:00'",
  qos="regular",
  nodes="2"
)

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)
rapidfire(launchpad, fworker_1, qadapter_1, sleep_time=60, reserve=True)
```

This is really the end



NERSC

BERKELEY LAB

U.S. DEPARTMENT OF ENERGY | Office of Science