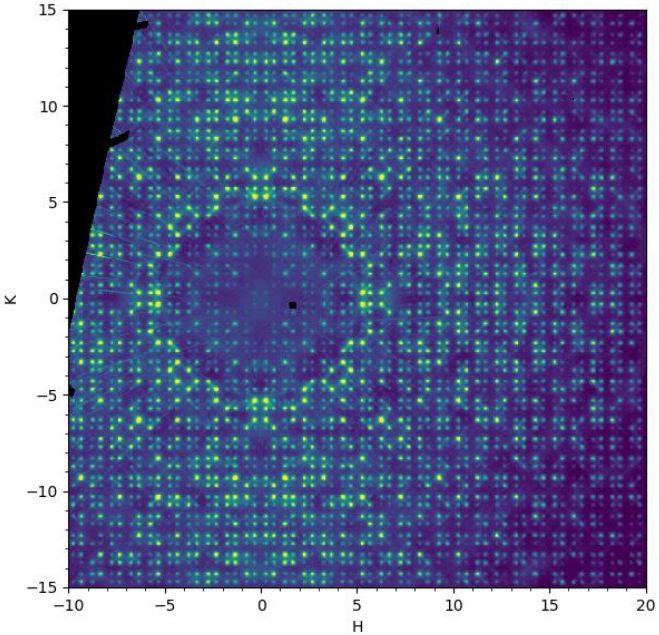
SCALING DIFFUSE SCATTERING WORKFLOWS WITH HYBRID HPC WORKFLOWS



(Sr,Ba)Nb2O6 short-range order

PRESENTER:

JUSTIN M. WOZNIAK – DATA SCIENCE & LEARNING

ADDITIONAL AUTHORS: RAY OSBORN, MATTHEW KROGSTAD, GUY JENNINGS, STEPHAN ROSENKRANZ – MATERIALS AND X-RAY SCIENCE

September 19, 2022

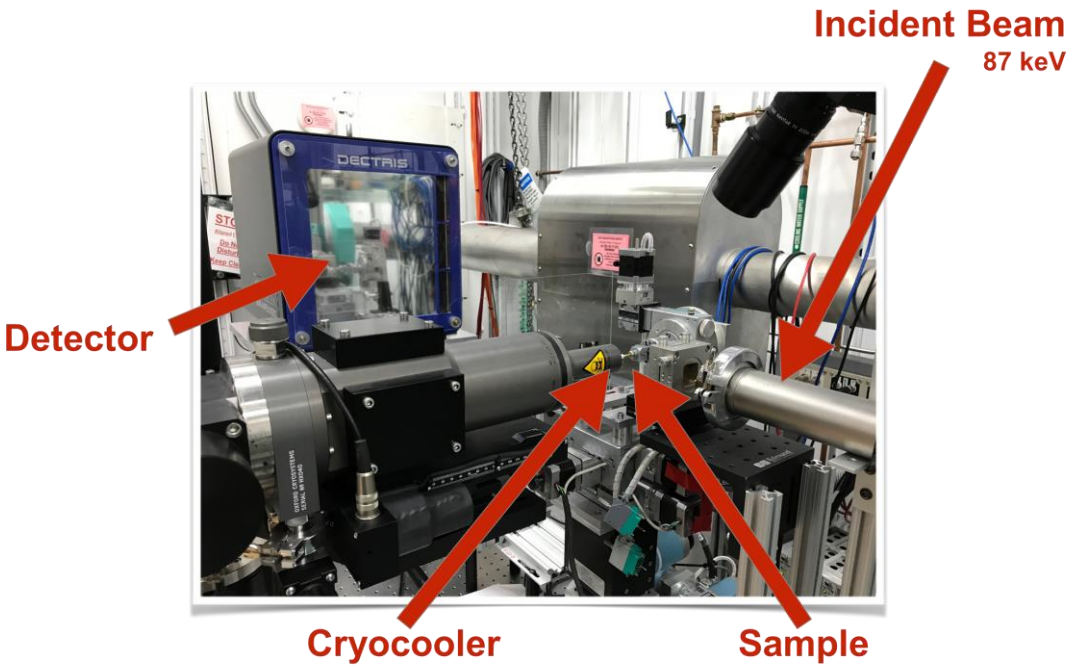
NOBUGS – Paul Scherrer Institute

OVERVIEW

- **Data collection**
  - Experimental setup
  - Example datasets
- **Data analysis and transformation**
  - GUI overview
  - Crystal coordinate transformation
- **HPC workflow solution**
  - GUI-controlled workflows
  - Distributed computing on Theta

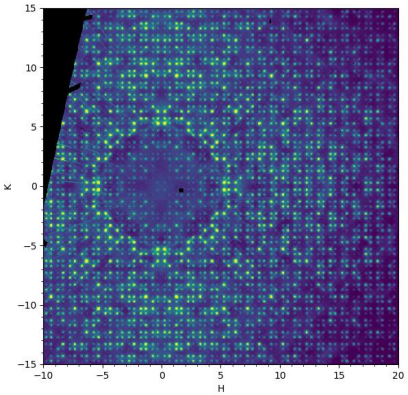
DATA COLLECTION

MEASURING X-RAY DIFFUSE SCATTERING WITH CONTINUOUS ROTATION METHOD

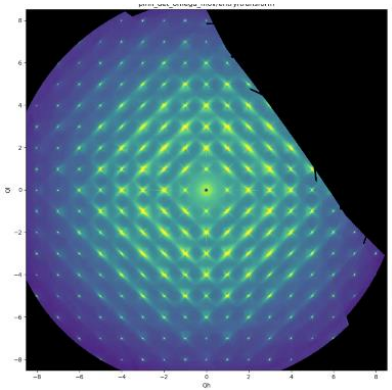


DIFFUSE GALLERY

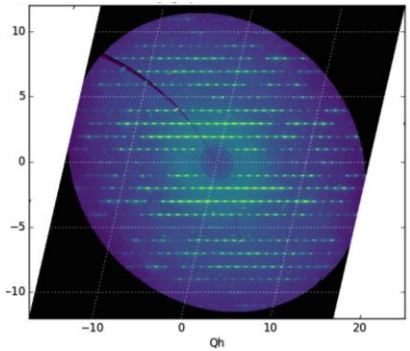
~50TB collected in
April and May, 2018



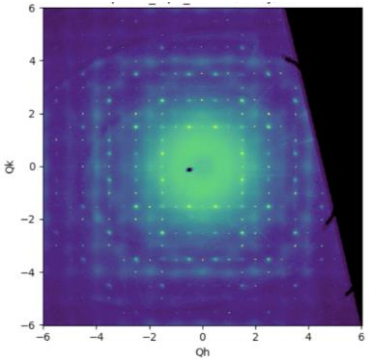
Tungsten bronze relaxor



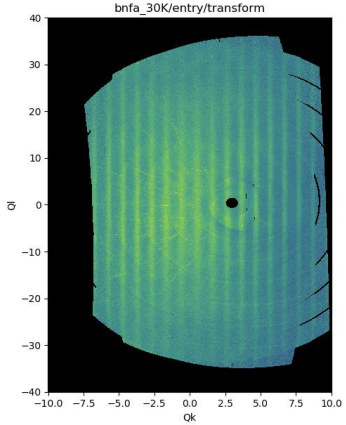
Lead Niobate relaxor



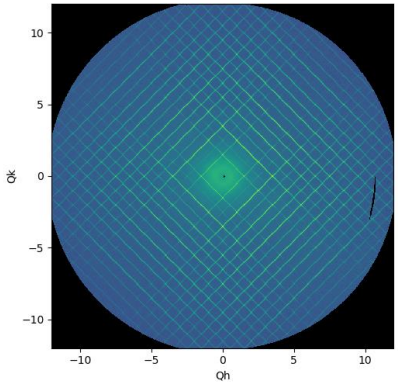
Sodium-intercalated
Vanadium Oxide



Cesium Lead Bromide



Iron Arsenide superconductor



(MoxVO2) short-range order

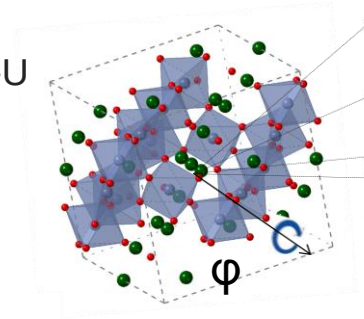
Argonne NATIONAL LABORATORY

X-RAY SCATTERING GEOMETRY

Continuous rotation method

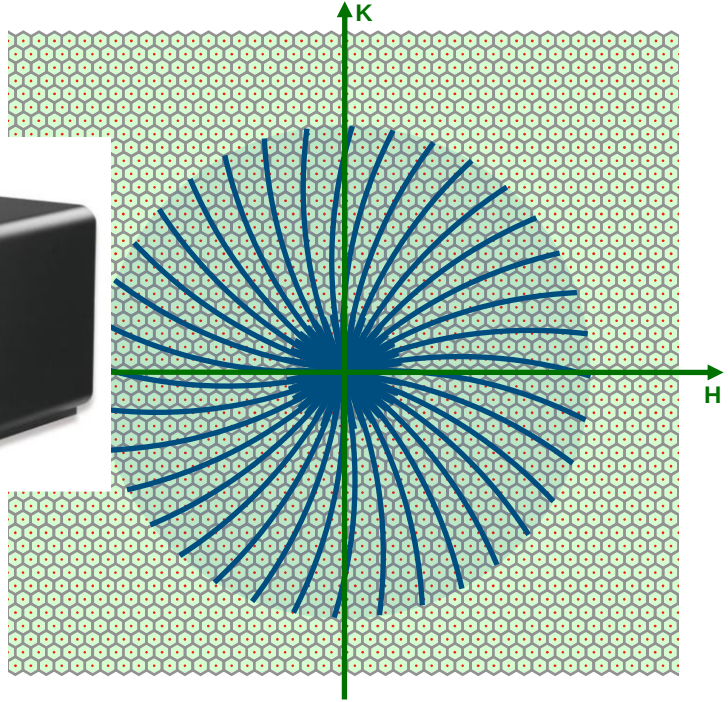
- The sample is continuously rotated at 1°s^{-1}
- Frames are collected at 10Hz
  - 3600 x 8MB frames
  - 30GB every 6 minutes
  - 3TB per day
  - > 10x after APS-U

Incident Beam



Sample

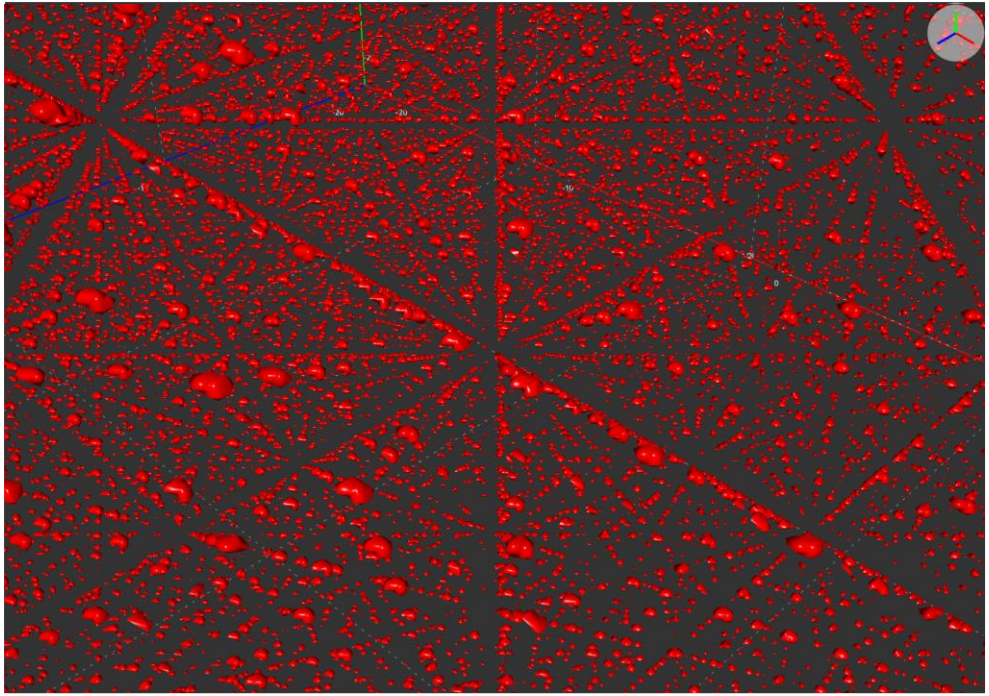
Pilatus 2M CdTe Detector



Reciprocal Space

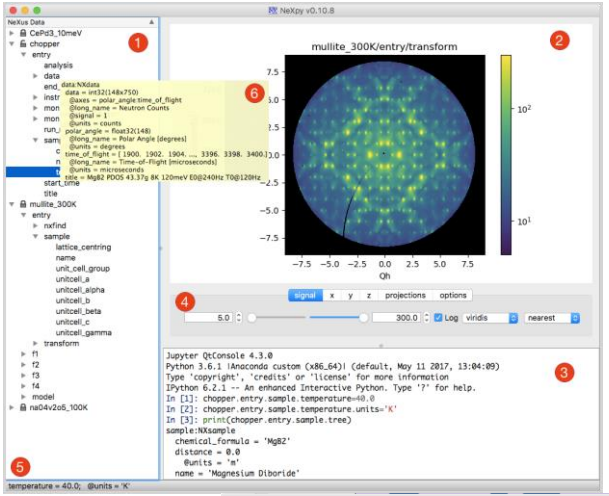
Q-RANGE IN ROTATION METHOD IN SECTOR 6

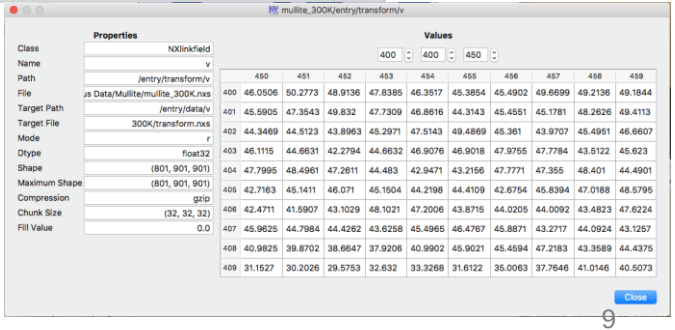
- With the following parameters, we cover $-15Å^{-1}<\mathbf{Q}<15Å^{-1}$
  - E_i ~ 87 keV
  - λ ~ 0.14 Å
  - Detector distance ~650 mm
  - Pilatus 2M CdTe: 1679x1475 pixels
  - Pixel size ~170 µm
- This Q-range includes thousands of Brillouin zones.
  - *e.g.*, for *a* ~ 10 Å, ~60,000 Bragg peaks



TRANSORMATION AND ANALYSIS

NEXPY: PYTHON TOOLBOX FOR NEXUS FILES





	450	451	452	453	454	455	456	457	458	459
400	46.0506	50.2773	48.9136	47.8385	46.3517	45.3854	45.4902	49.6699	49.2136	49.1844
401	45.5905	47.3543	49.832	47.7309	46.8616	44.3143	45.4551	45.1781	48.2626	49.4113
402	44.3469	44.5123	43.8963	45.2971	47.5143	49.4869	45.361	43.9707	45.4951	46.6607
403	46.1115	44.6631	42.2794	44.6632	46.9076	46.9018	47.9755	47.7784	43.5122	45.623
404	47.7995	48.4961	47.2611	44.483	42.9471	43.2156	47.7771	47.355	48.401	44.4901
405	42.7163	45.1411	46.071	45.1504	44.2198	44.4109	42.6754	45.8394	47.0188	48.5795
406	42.4711	41.5907	43.1029	48.1021	47.2006	43.8715	44.0205	44.0092	43.4823	47.6224
407	45.9625	44.7984	44.4262	43.6258	45.4965	46.4767	45.8871	43.2717	44.0924	43.1257
408	40.9825	39.8702	38.6647	37.9206	40.9902	45.9021	45.4594	47.2183	43.3589	44.4375
409	31.1527	30.2026	29.5753	32.632	33.3268	31.6122	35.0063	37.7646	41.0146	40.5073

1. Tree View
2. Plot View
3. IPython Shell
4. Axis Panels
5. Status Bar
6. Tooltips

http://nexpy.github.io/nexpy/

$ pip install nexpy

$ conda install -c nexpy nexpy

pyFAI
Fabio

localhost

Forecast Maps Apple Argonne Neutrons Journals Computing BBC News Local

Searchi... Python... Haitink... NOBUG... https://... Home nexusfo...

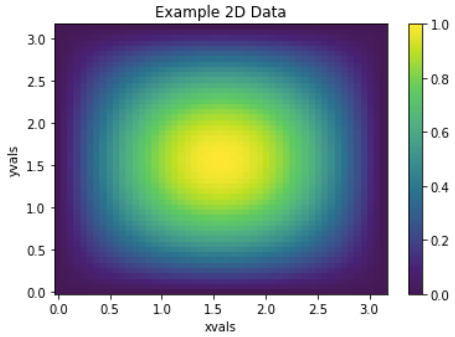
jupyter **nexusformat** (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Code

In [24]:
```
print(data2D[0:np.pi,0:np.pi].tree)
data2D[0:np.pi,0:np.pi].plot()
```

```
data:NXdata
  @axes = ['yvals', 'xvals']
  @signal = 'zvals'
  title = 'Example 2D Data'
  xvals = float64(51)
  yvals = float64(51)
  zvals = float64(51x51)
```



Manipulating NXdata Groups

The NXdata groups can be used in arithmetic expressions involving addition, subtraction, multiplication, and division, using the same broadcast rules as Numpy arrays.

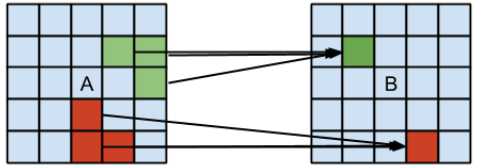
In [25]:
```
bkgd=100.*data1D.energy_transfer
(2*data1D+bkgd).plot()
```

Example 1D Data

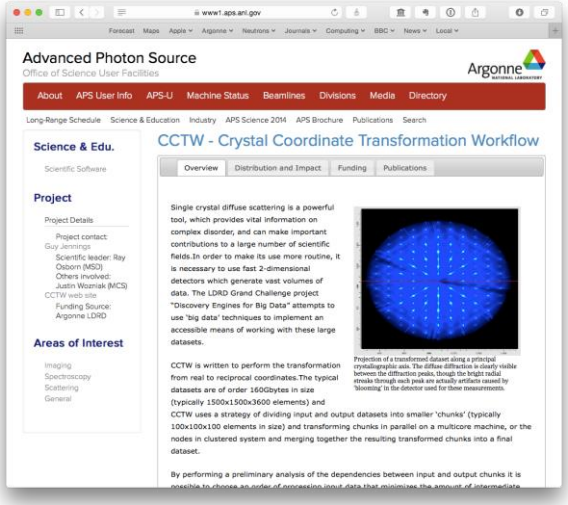
Argonne NATIONAL LABORATORY

CCTW - CRYSTAL COORDINATE TRANSFORMATION WORKFLOW

Written by Guy Jennings



- Performs transformations from detector coordinates to reciprocal space
- Full volume transformations using a strategy of dividing input and output datasets into smaller ‘chunks’
  - typically 100x100x100 elements in size
  - chunks transformed in parallel on a multicore machine
  - chunks merged together in the final dataset.
- Operates on up to 50 GB data
- Core numerics in C++
- Considerably faster than prior code
- Previous programs were too slow to transform whole volumes
- CCTW operates in only 10-20 minutes for a Pilatus 2M detector

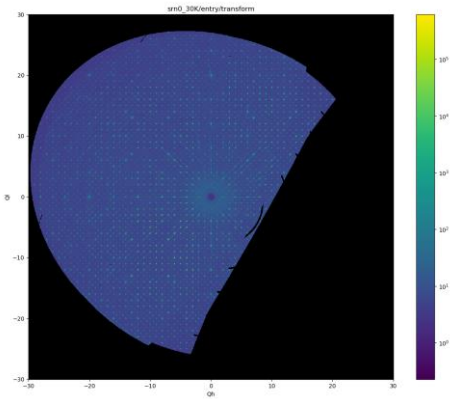


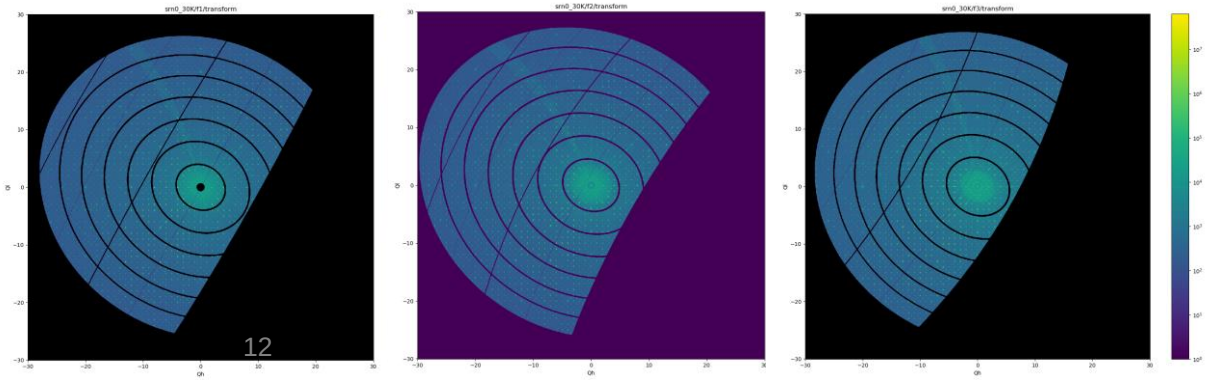
Argonne NATIONAL LABORATORY

INITIAL WORKFLOW

The parameters of the workflow are determined interactively after an automatic peak search.

- Refine detector geometry
- Determine orientation matrix
- Perform coordinate transformation Using CCTW
- Merge transforms from three positions
- Once the sample is oriented, the workflow is automated for other temperatures; the orientation matrix is refined automatically





HPC WORKFLOWS

HPC SYSTEMS AT ALCF

Theta and related systems

- **Argonne Leadership Computing Facility**

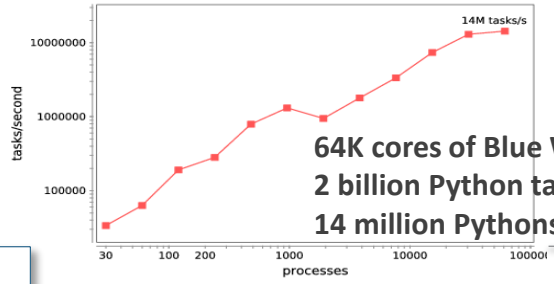
- **Theta (this work)**
  - Based on the Intel KNL 7230 – no GPUs
  - 4,392 compute nodes
  - 64 cores per node x 4 threads per core
  - 208 GB RAM per node
  - Near 12 Petaflops

- **Aurora**
  - Based on 60,000 Ponte Vecchio GPUs
  - Will exceed 2 Exaflops

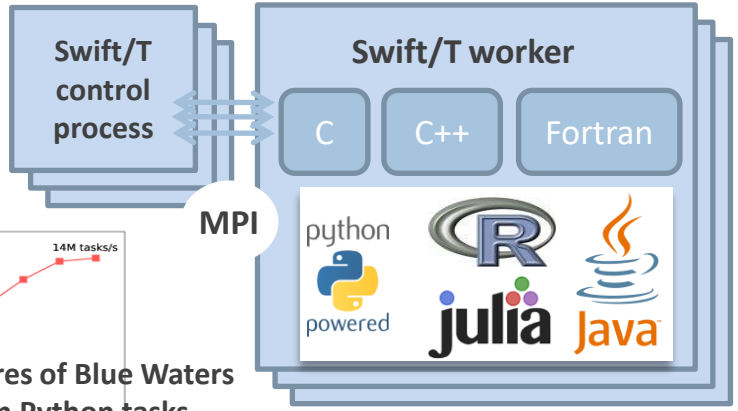
SWIFT/T: ENABLING HIGH-PERFORMANCE SCRIPTED WORKFLOWS

Supports many task types for HPC

- Write site-independent scripts, translates to MPI
- Automatic task parallelization and data movement
- Invoke native code, script fragments in Python and R
- Rapidly subdivide large partitions for MPI jobs in multiple ways (MPI 3.0)



64K cores of Blue Waters
2 billion Python tasks
14 million Pythons/s

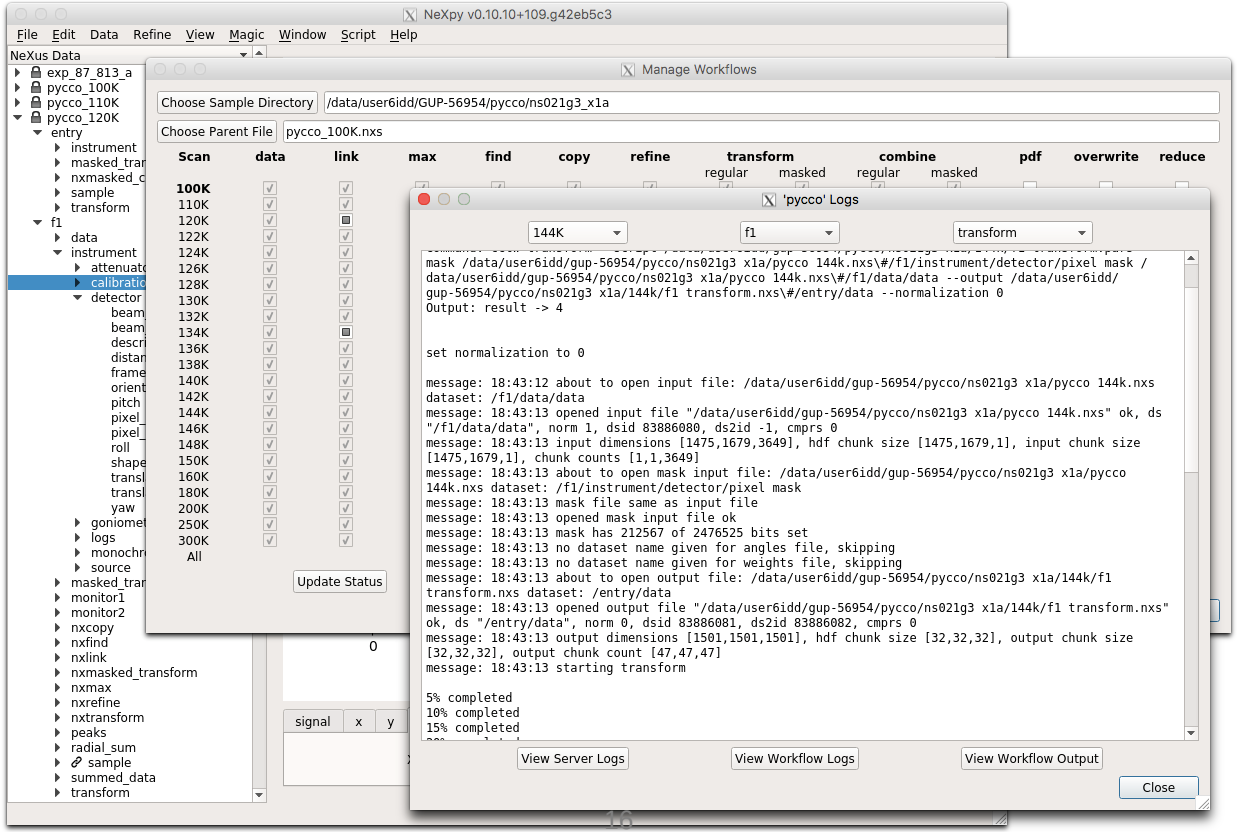



```
$ spack install stc
```

```
$ conda install -c lightsource2-tag swift-t
```

Swift/T: Scalable data flow programming for distributed-memory task-parallel applications
Proc. CCGrid 2013.

WORKFLOW MANAGER



Argonne NATIONAL LABORATORY

SUPERCOMPUTING USE CASE

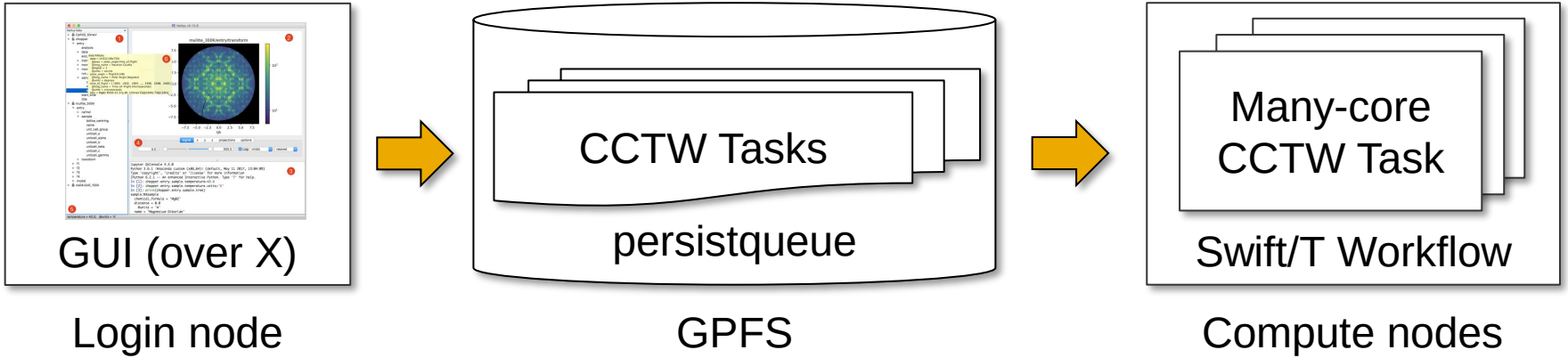
Managing near-real-time data collection and analysis

- Use case: As data streams onto shared storage, it is informally visualized for clarity, then automated data collection is started at various temperatures and other settings
- Data sets must be oriented, a semi-manual process
- Thus, a backlog of transformations (for CCTW) is queued up (tens to hundreds)
- Data is automatically synced from beamline storage to the ALCF via Globus
- Need to distribute work from the GUI-based workflow manager to the Theta compute nodes
- Need to integrate Python GUI with Swift/T workflow

CONNECTING TO ALCF THETA

Combine Python-based GUI with MPI-based Swift/T workflow

- The GUI workflow fills in a small SQLite database with the work to be done and inserts requisite command lines into a Python persist-queue
- This queue is stored in a generic filesystem and is portable to Theta, where it is accessible by both the login and compute nodes
- Entries in the queue are consumed by a Python snippet in the Swift/T workflow and distributed to free workers



CODE SNIPPETS

```python
class NXFileQueue(FileQueue):
    """A file-based queue with locked access"""

    def __init__(self, directory, autosave=True):
        self.directory = directory
        tempdir = os.path.join(directory, 'tempdir')
        self.lockfile = os.path.join(directory, 'filequeue')
        if not os.path.exists(tempdir):
            os.makedirs(tempdir)
        with NXLock(self.lockfile):
            super().__init__(directory, serializer=json,
                             autosave=autosave,
                             tempdir=tempdir)
            self.fix_access()

    def put(self, item, block=True, timeout=None):
        with NXLock(self.lockfile):
            super().put(item, block=block, timeout=timeout)
            self.fix_access()

    def get(self, block=True, timeout=None):
        with NXLock(self.lockfile):
            item = super().get(block=block, timeout=timeout)
            self.fix_access()
        return item
```

GUI workflow

Swift/T workflow – queue access

```
string code = """
QUEUE = os.getenv("HOME") + "/NX-Q/nxserver/task_list"
Q = NXFileQueue(QUEUE)
try:
  TASK = Q.get(block=False, timeout=1)
except Empty:
  TASK = "PERSISTQUEUE.EMPTY"
""";
```

Swift/T workflow – task execution

```python
def do_work(msg):
    print("do_work: " + msg + " ...")
    sys.stdout.flush()
    subprocess.call(msg.split())
    sys.stdout.flush()
```

CONCLUSION

- Illustrated end-to-end automated data transformation and analysis workflow with HPC capacity to handle realistic data backlogs
- Demonstrated integration of existing Python GUI and MPI-based workflow technologies
- Based on established codes tested over many data collection cycles

ACKNOWLEDGMENTS

- This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. This research was completed with resources provided by the Laboratory Computing Resource Center at Argonne National Laboratory.
- This material is based upon work supported by the U.S. Department of Energy, Office of Science, under contract number DE-AC02-06CH11357.
- This work was supported by the U.S. Department of Energy, Office of Basic Energy Sciences, Division of Materials Sciences and Engineering and used resources of the Advanced Photon Source, a U.S. Department of Energy Office of Science User Facility at Argonne National Laboratory.

QUESTIONS?

WOZ, ROSBORN
@ANL.GOV