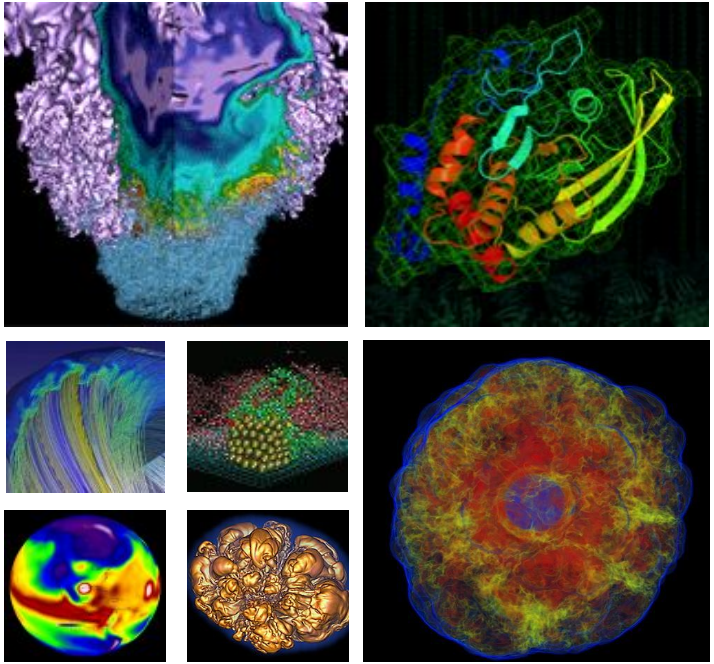
Python at NERSC



Rollin Thomas

NERSC Data and Analytics Services

IXPUG

2018-05-10

Outline

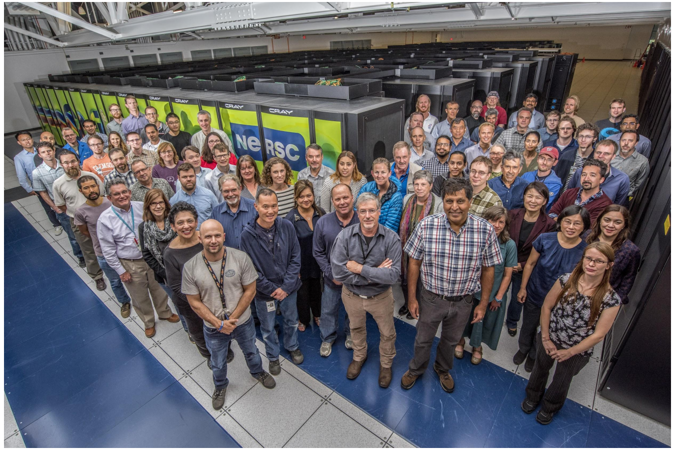
1. **Python enables HPC science at NERSC**
   Orchestration • Workflows • Analytics • HPC Apps

2. **How we help Python users at NERSC**
   Productivity • Performance

3. **Experimental/Observational Science Engagements**
   Python in NESAP for Data Projects w/Intel

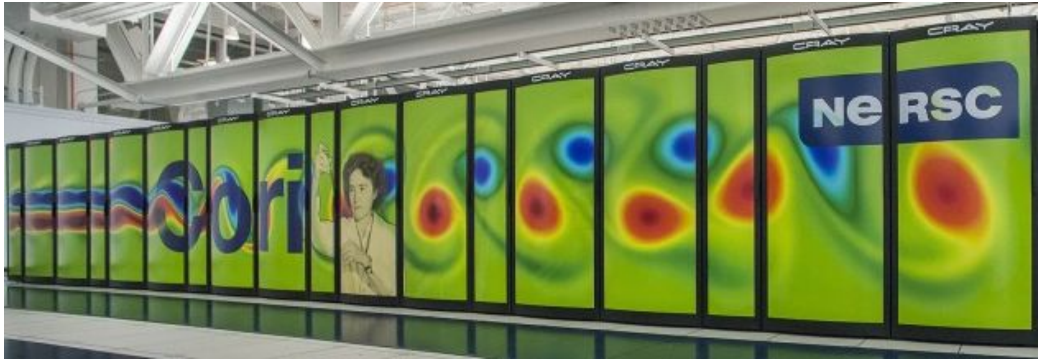
What is NERSC?

National Energy Research Scientific Computing Center



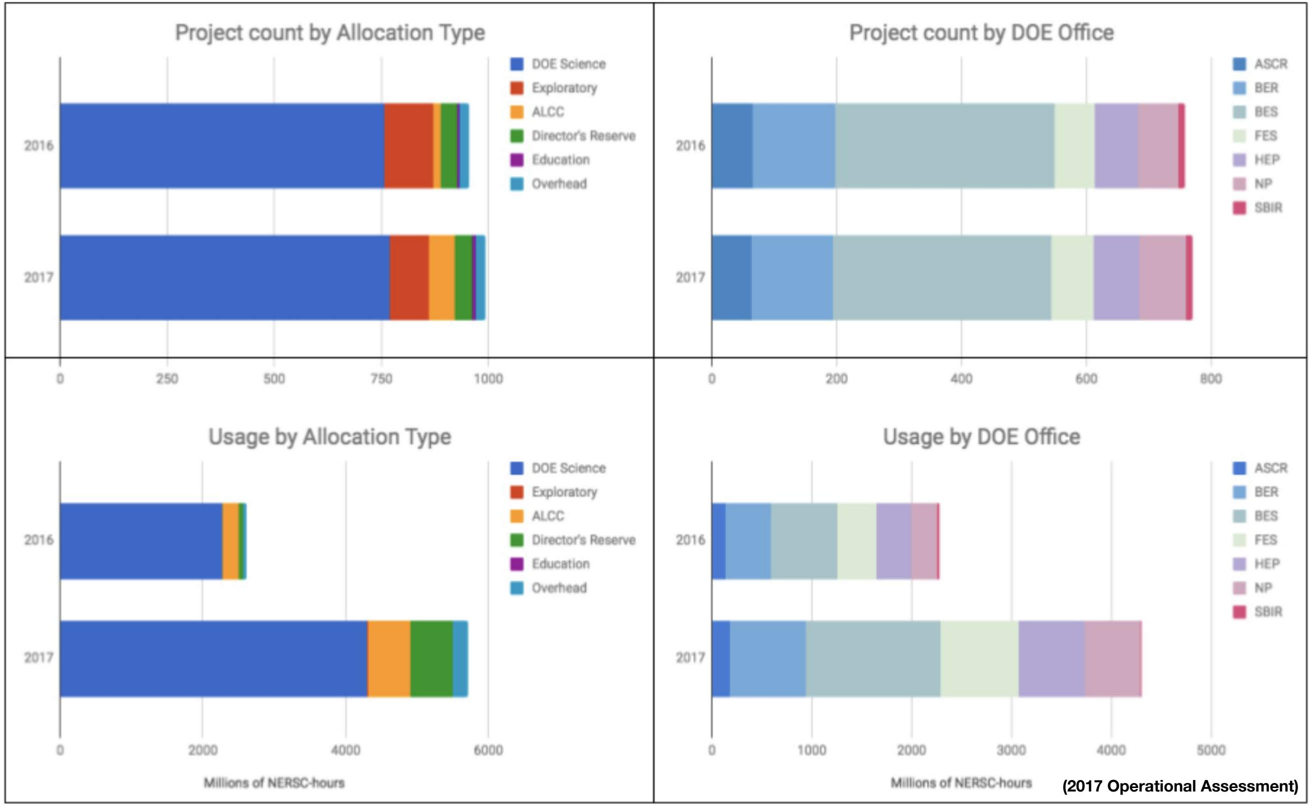
The production user facility for high performance computing and data for the Department of Energy's Office of Science.

NERSC Systems



Name	System Type	Processor Type	Speed/Cores per Node	Peak Performance	# Nodes	Aggregate Memory	Memory per Node
Edison	**Cray XC30**	**Ivy Bridge**	**2.4/24**	**2.57 PF/s**	**5586**	**357 TB**	**64 GB**
Cori	**Cray XC40**	**Haswell**	**2.3/32**	**1.92 PF/s**	**2388**	**305 TB**	**128 GB**
		KNL	**1.4/68**	**28 PF/s**	**9688**	**1.1 PB**	**96+16 GB**

7K users, 800 projects, 2K papers



(2017 Operational Assessment)

Advanced Scientific Computing Research

Biological & Environmental Research

Basic Energy Sciences

Fusion Energy Sciences

High Energy Physics

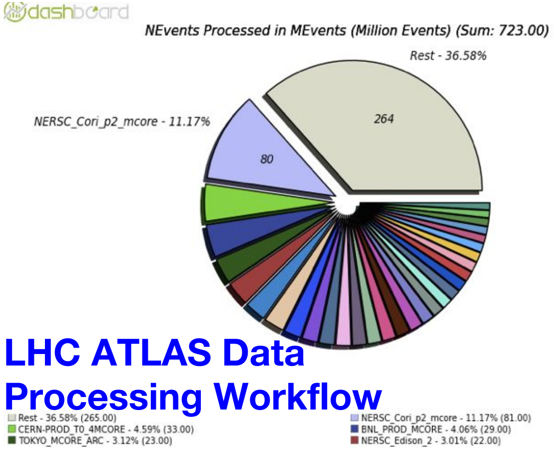
Nuclear Physics

Small Business Innovation Research

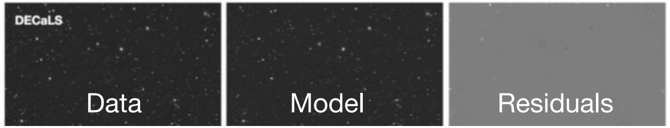
~10 billion "NERSC hours" provided to users in 2017.

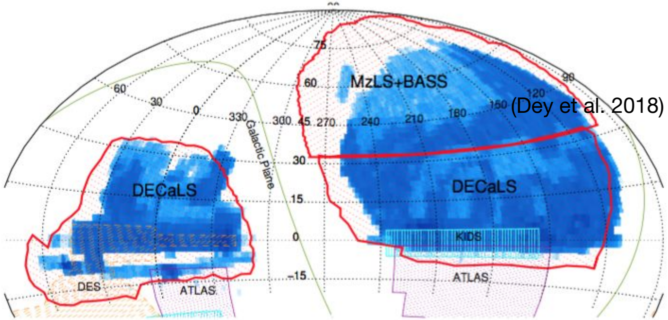
Science via Python@NERSC

Powering Workflows to Understand Properties of Materials



LHC ATLAS Data Processing Workflow

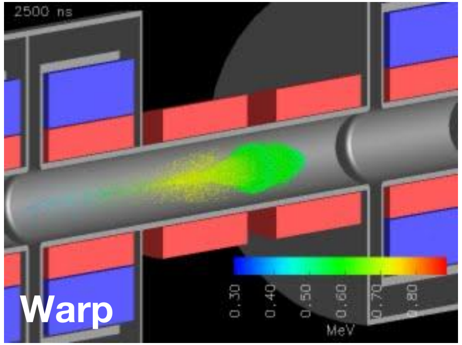




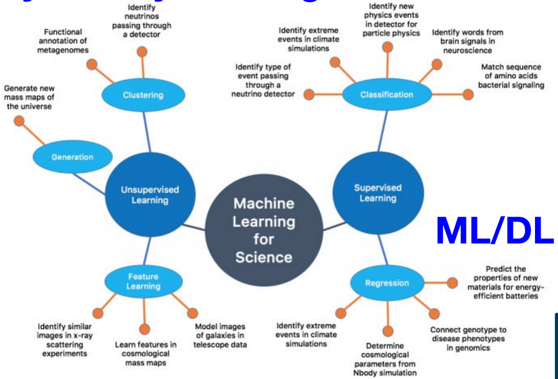
Sky Survey Catalogs for Cosmology

NBODYKIT

Modeling Dark Matter and Dark Energy



PIC Code for Plasmas and High Current Particle Beams



ML/DL

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Python in HPC Jobs at NERSC

Around 3% of NERSC hours on Cori in the past year easily detected as Python jobs*:

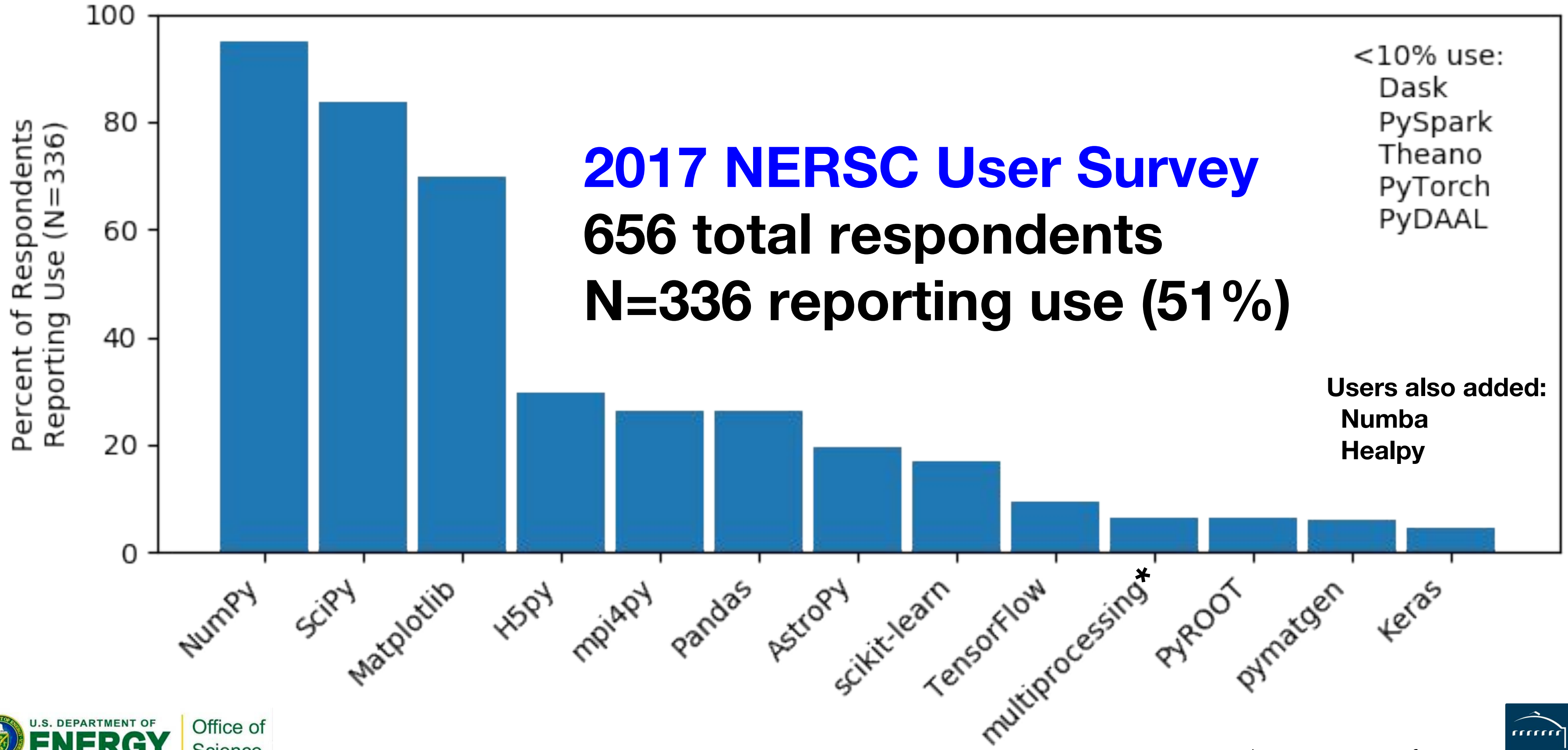
```
srun -n … python whatever.py …
```

***This is a lower limit*, as users:**

- **Often make main programs executable**
- **Use Python in containers to scale up**

* Production batch *jobs*, not use on shared login nodes.

Packages Users Say They Use



2017 NERSC User Survey

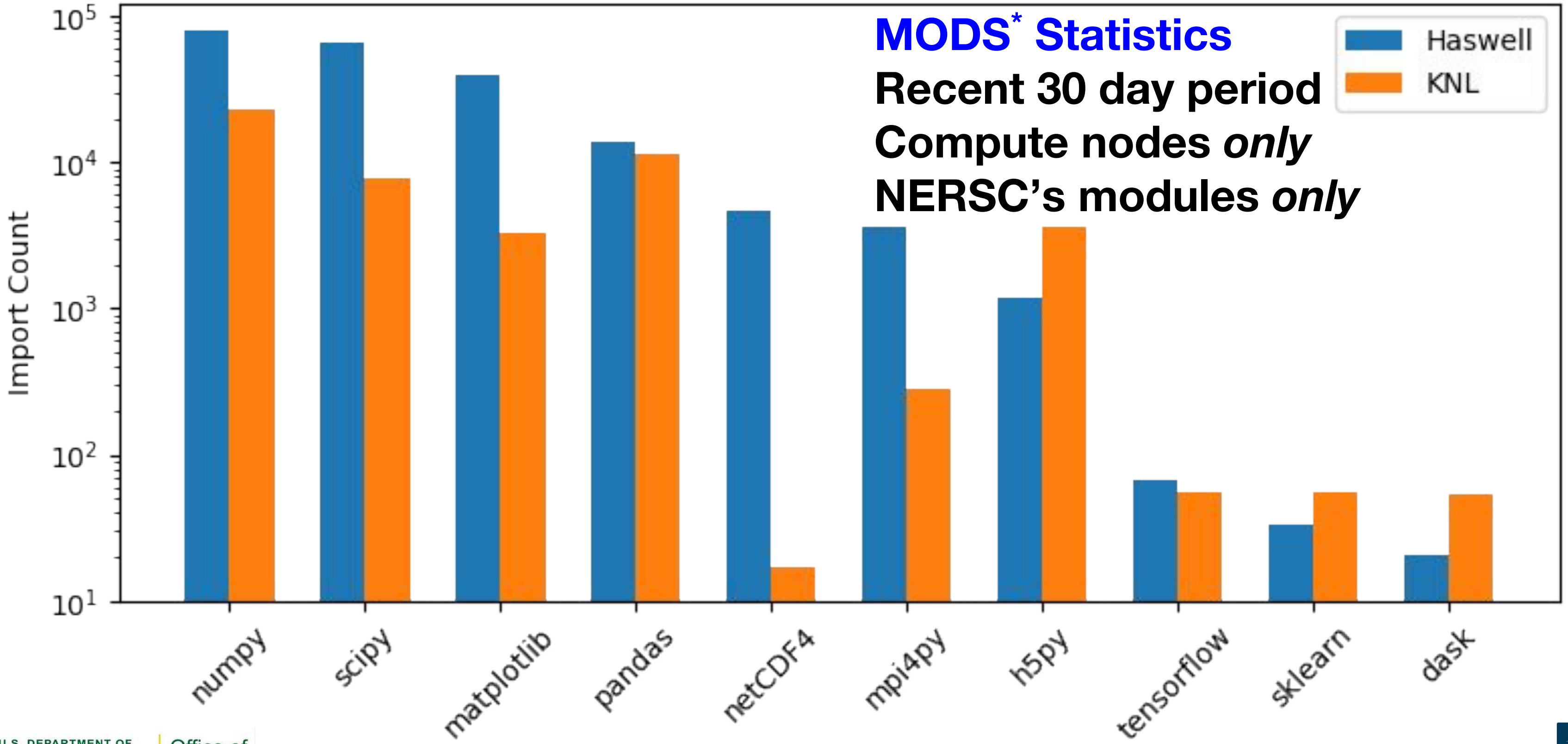
656 total respondents

N=336 reporting use (51%)

Users also added:
Numba
Healpy

* or concurrent.futures

Monitored Imports (Cori)



MODS* Statistics

Recent 30 day period

Compute nodes *only*

NERSC's modules *only*

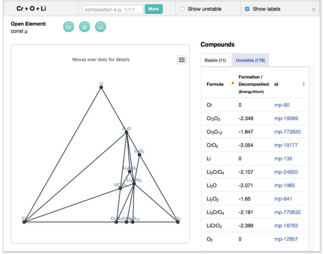
* MODS = Monitoring of Data Services at NERSC = BI Project in DAS

Python in Edge Services

enables science through . . .

Data Sharing Across Facilities

Interfaces to HPC resources & workflows

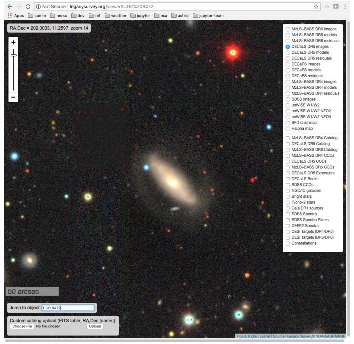


Interactive Tools

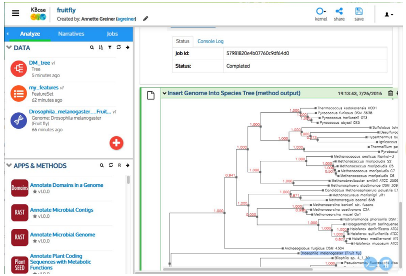
The Legacy Surveys

The Legacy Surveys are producing an inference model catalog of the sky from a set of optical and infrared imaging data, comprising 14,000 deg² of extragalactic sky visible from the northern hemisphere in three optical bands (g, r, z) and four infrared bands. The sky coverage is approximately bounded by -18° < δ < +84° in celestial coordinates and |b| > 18° in Galactic coordinates. To achieve this goal, the Legacy Surveys are conducting 3 imaging projects on different telescopes, described in more depth at the following links:

The Beijing-Arizona Sky Survey (BASS)	*The DECam Legacy Survey* (DECaLS)	*The Mayall z-band Legacy Survey* (MzLS)



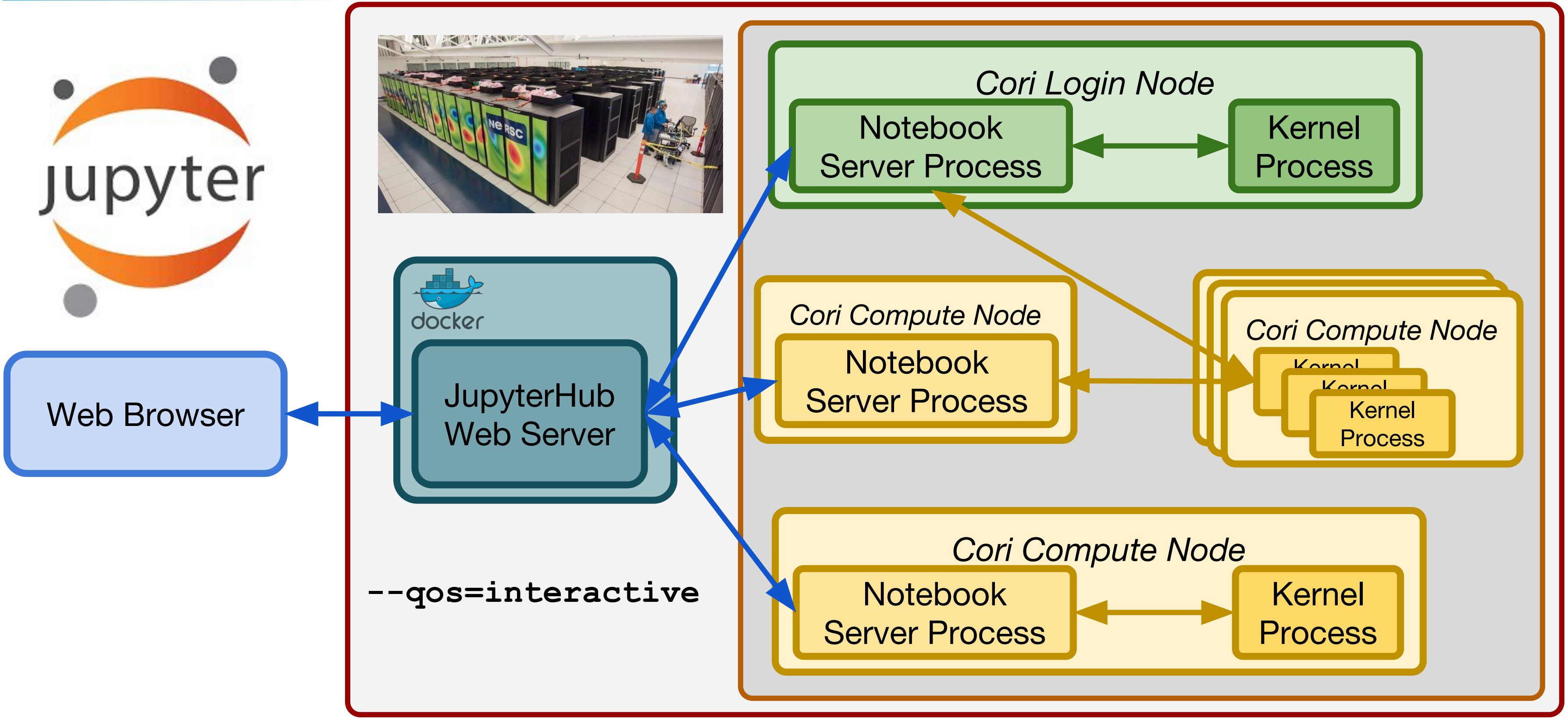
Rich Visualizations and UIs



U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Interactive Supercomputing



NERSC's Python Strategy

Focus on user productivity.
Support familiar, trusted, up-to-date libraries.
Find ways to put performance in user reach.

Examples:

Threaded libraries:	Intel MKL
Support cluster scaling:	Cray+mpi4py
Close architecture gaps:	Containers

NERSC Python: Anaconda

Most well-known and widely used distribution.
Designed around analytics, statistics, ML/DL.
"Personalized" environments and package manager.
Environments are rapid setup, shareable, & reusable.

2016: MKL added, and Intel upstreams optimizations:
NERSC drops its builds of Python on Cray the same year.

Other options for HPC:

Source builds, Spack, etc.

Handling MPI with mpi4py

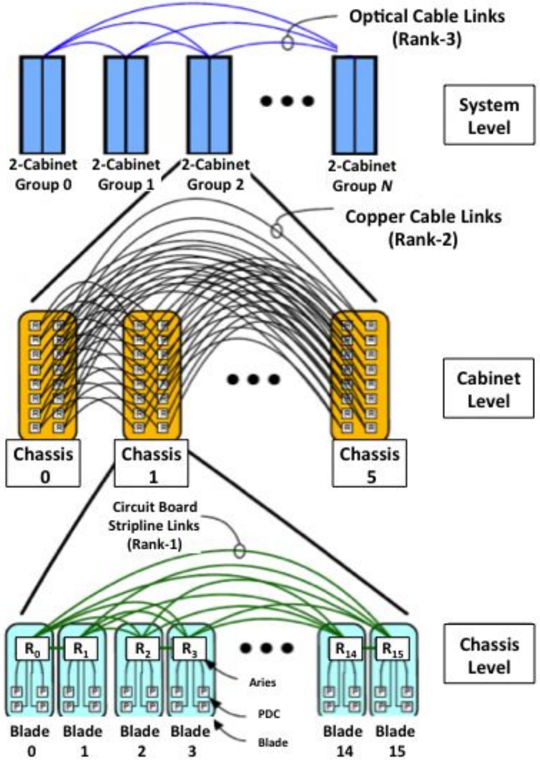
Cluster parallelism with MPI via mpi4py:

- MPI-1/2/3 specification support
- OO interface ~ MPI-2 C++ bindings
- Point-to-point and collectives
- Picklable Python objects & buffers

Build mpi4py & dependents with Cray MPICH:

```
python setup.py build --mpicc=cc
python setup.py install
```

Cray-provided Compiler wrapper



Cori Aries Interconnect

Containers

and Python go well together at NERSC

Motivations, esp. for data science:

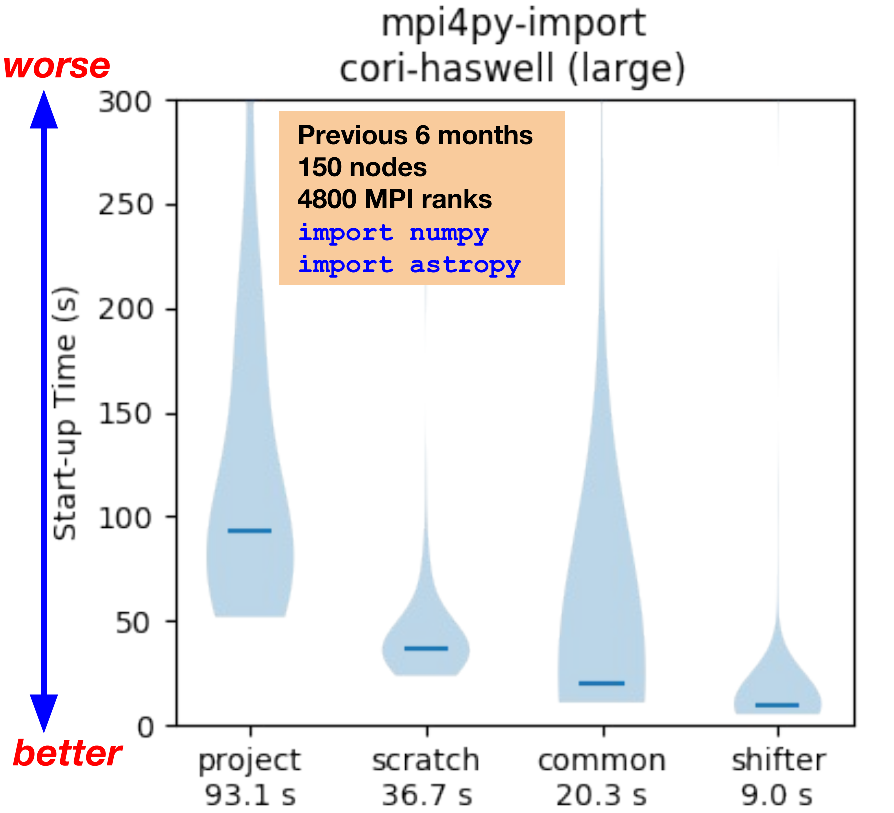
- Flexibility
- Convenience
- Consistency
- Reproducibility

Some Options:

- Docker
- *Shifter (~Docker on Cray)*
- Singularity
- CharlieCloud

Nice recent blog summary of the state of HPC containers:
https://www.stackhpc.com/the-state-of-hpc-containers.html

“Slow Launch” at Scale



[Median launch time incl. MPI_Init()]

Python’s import is metadata intensive,
⇒ catastrophic contention at scale
⇒ it matters where you put your env

Project (GPFS):
For sharing large data files

Scratch (Lustre):
OK, but gets purged periodically!

Common (GPFS):
RO w/Cray DVS client-side caching
Open to users now, was only staff

Shifter (Docker Containers):
Metadata lookup only on compute
Storage on compute is RAM disk
ldconfig when you build image

Python on Knights Landing

General advice from to users regarding KNL:

- **Your code will run without modification**
- **Expect some refactor if you want best performance**
- **Fine-grained parallelism to exploit 68 cores/node**
- **Make aggressive use of the 512-bit vector units**
- **Structure data to stay in KNL's 16 GB of MCDRAM**

Python on Knights Landing

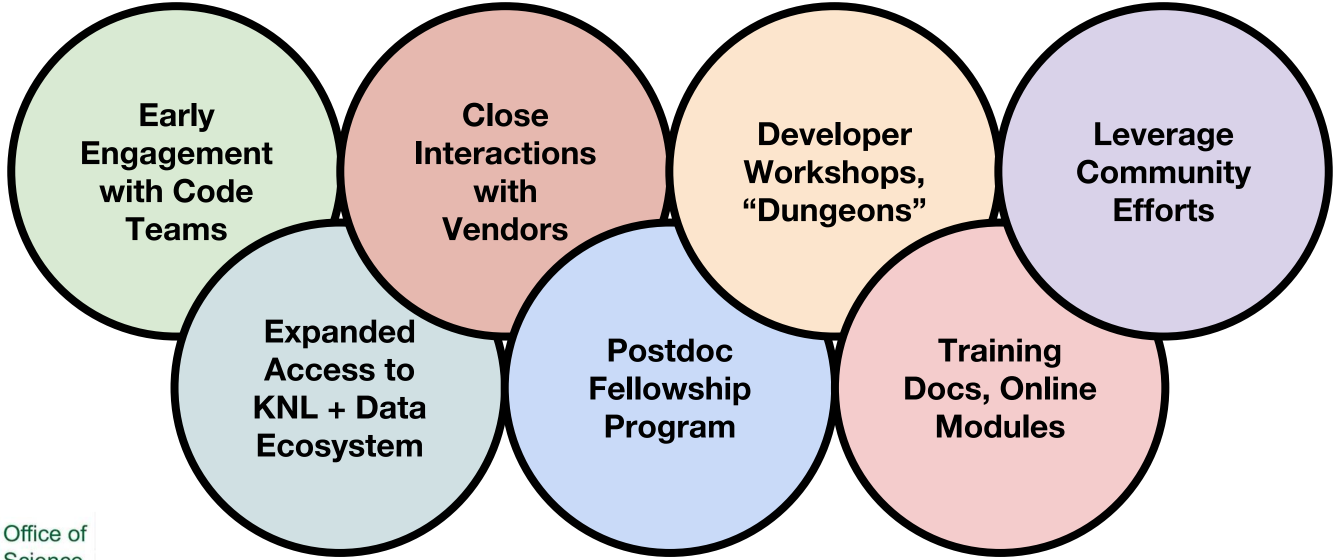
Translation for Python users. At least,

- **Understand and use numpy array syntax, broadcast rules, and scalar/“vector” interfaces to functions.**
- **Use threaded+vectorized libraries and compiled extensions, minimize time outside of using them.**
- **There may, in fact, be more than one way to do it; Prepare to rethink algorithms, memory usage, etc.**
- **Layer use of profiling tools to identify/assess hotspots.**

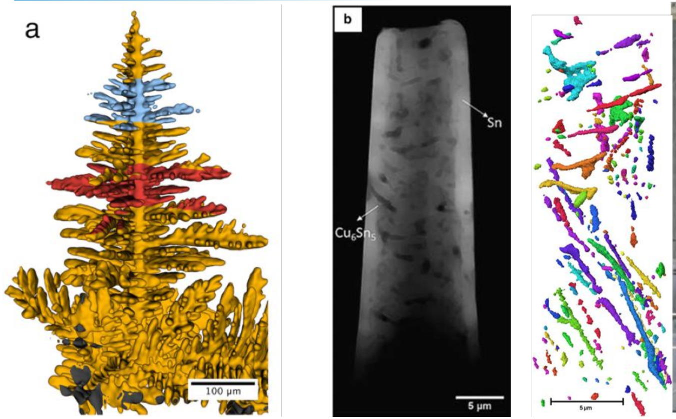
NESAP for Data

NERSC Exascale Science Applications Program for Data:

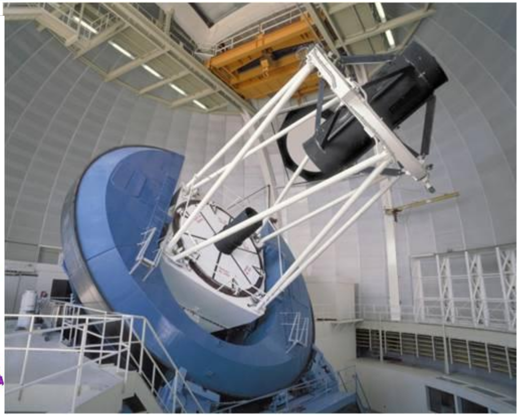
Users whose applications process, analyze, and/or simulate data sets or data streams from experiments and instrumentation supported by DOE need help preparing for extreme scale and exascale computing.



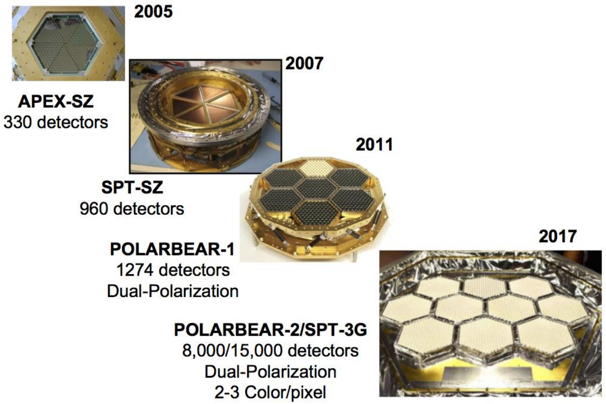
Python NESAP for Data Projects



TomoPy



DESI



TOAST

TomoPy (Python & C):

Tomographic data processing and image reconstruction

PI: Doga Gursoy, Argonne National Laboratory

DESI Pipeline (As Pure Python as Reasonably Possible):

Baryon acoustic oscillations (DESI Project)

PI: Stephen Bailey, Lawrence Berkeley Laboratory

TOAST (Time Ordered Astrophysics Scalable Tools, Python & C++):

Cosmic microwave background data analysis and simulation (CMB S4)

PI: Julian Borrill, Lawrence Berkeley Laboratory

TomoPy

(D. Gursoy, Z. Ronaghi; O. Pavlyk)

Framework for analysis of synchrotron tomography data

75% Python, ~25% C/C++

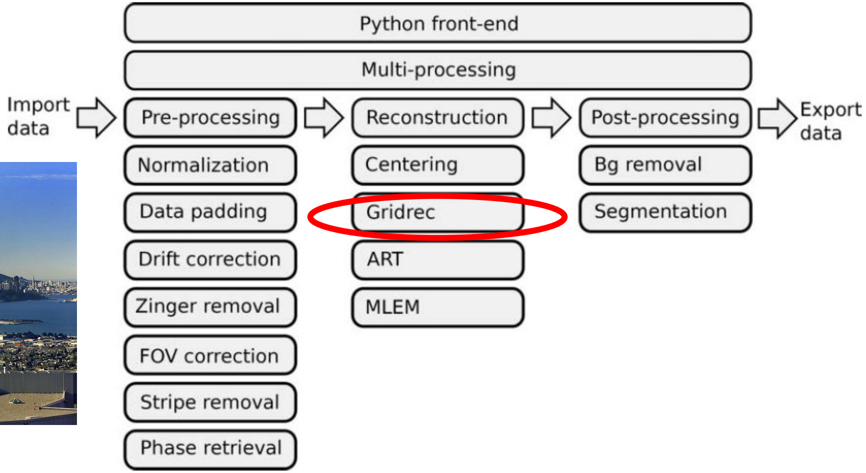
I/O handled via dxchange, HDF5

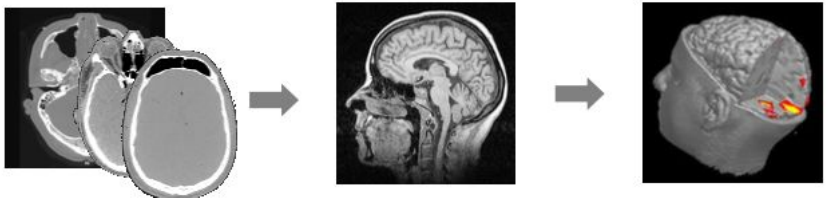
Modular design

- Pre-processing
- Image Reconstruction
- Post-processing

Parallelism within node:

- multiprocessing
- concurrent.futures





TomoPy Optimizations So Far

(mostly gridrec algorithm)

Library dependencies:

- Intel Python: NumPy, SciPy, scikit-image
- Local builds: fftw, pyfftw, dxchange, dxfile, olefile

Compilation:

- Build TomoPy C extensions with icc
- Target common-avx512 architecture enabling vectorization

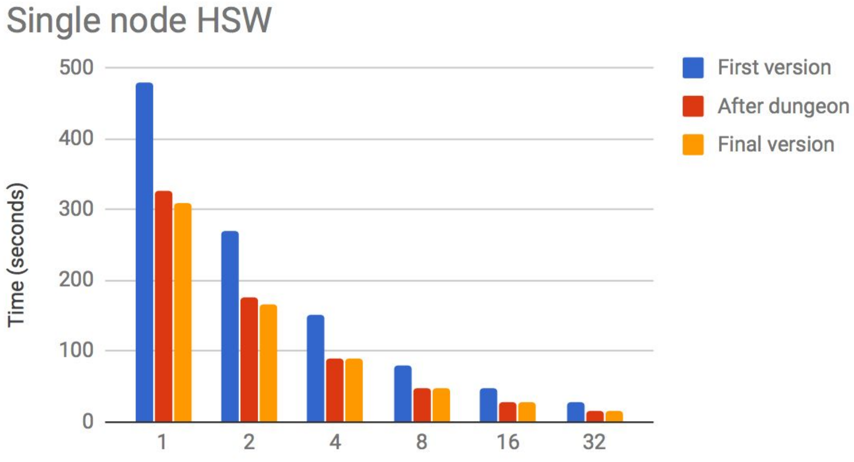
Runtime:

- To use multiprocessing on KNL needed to set KMP_AFFINITY=disabled
- ***Using huge memory pages: gridrec algorithm 30% faster on Haswell, 45% faster on KNL***

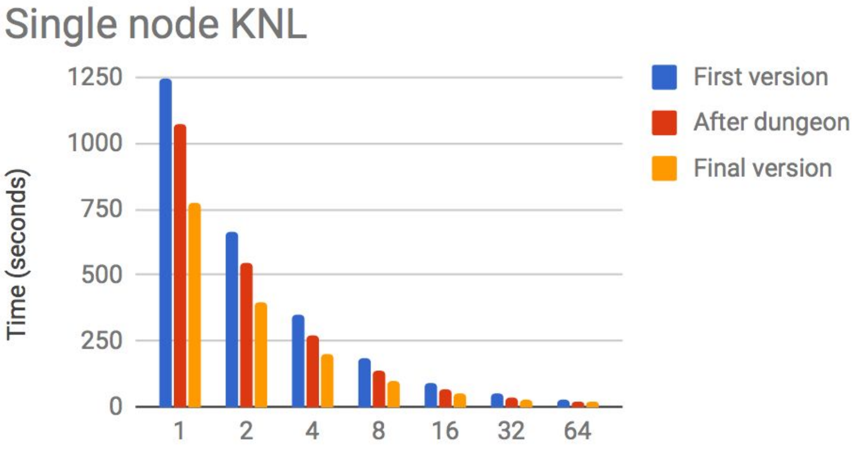
Code changes, mostly in C layer gridrec algorithm, some preprocessing:

- Appropriate precision (**`ceilf, sinf, cosf`**), avoid upcast/downcast if we can
- Replace **`lroundf(x)`** with **`(int)roundf(x)`** to enable vectorization
- Apply icc-specific vectorization pragmas (Intel Compiler)
- 64-byte aligned memory allocation from X/Open-7 posix standard instead of fftw allocator
- Parallelize FFTs with “many FFT” for more slices at once
- ***Employ cache blocking to lower miss rate for interpolation step (25% to 4% on KNL)***
- Pre-processing: Encapsulated uses of fft, ifft and fft2, ifft2 by replacing direct calls to pyfftw.interfaces.numpy functions with calls to wrapper functions

TomoPy Gridrec Optimizations



Cores (HSW)	First version	After dungeon	Final version
1	478	327	309
2	268	175	166
4	149	88	89
8	79	47	47
16	45	26	26
32	27	14	14

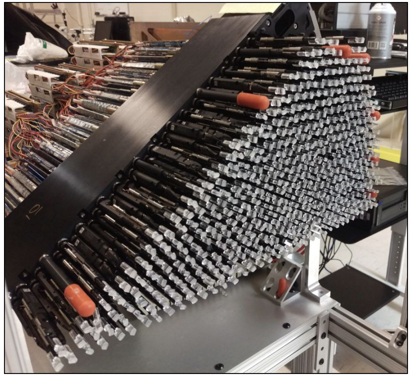


Cores (KNL)	First version	After dungeon	Final version
1	1248	1074	774
2	662	547	397
4	350	275	200
8	182	139	101
16	92	69	52
32	48	36	27
64	26	21	20

DESI [Bailey, Stephey; Pavlyk, Douyeb, Fernandez, Hogan]

Science Purpose: Spectroscopy for Dark Energy science

- 3D map of the Universe over 10 billion years
- Spectra of 10's of millions of galaxies and quasars
- *Create flux-calibrated 1D tables of flux vs wavelength of Galaxies, quasars, etc. from 2D CCD image frames*

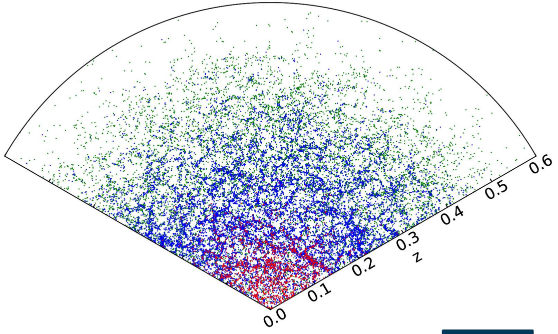


DESI Fiber Positioner Petal

1 Exposure = 30 Frames = 15,000 Traces

Algorithms and Methods

- Scientific Python stack (NumPy, SciPy, etc.; threaded)
- Linear algebra (esp. Hermitian eigen-decomposition)
- Special function evaluations, fitting functions to data
- MPI (mpi4py) data-parallel processing + Shifter to scale up



Production Requirements

- Real-time pressure to do real-time survey planning each day

DESI Optimization & Scaling

Simulation Code (Simulate Spectra on CCDs): 1.5-1.7x on HSW, multi-node scaling w/MPI

- Numba JIT compilation to speed up 2 lines of expensive matrix slicing
- MPI work to scale up the code:
  - Broadcast/reduce to scatter/gather where best use, complete initial I/O faster
  - Multi-level Comm scheme to optimally fill nodes
  - Scale tests up to 60 nodes so far, will be used in production soon
  - Single exposure (30 frames simultaneously) in 8 minutes
  - Roughly equal performance between multiprocessing and MPI on single node

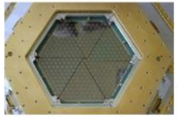
Main Extraction Code (1D traces from CCD images)

- Main bottleneck is `legval` in NumPy (scalar/vector args) observed at first Dungeon.
- ***Precompute `legval` w/large vector input (not scalar): promising but delicate refactor.***
- ***Also `legval` itself: 4x speedup with loop unrolling and Numba.***
- Using some of the code as a testbed for initial experimenting with PyPy.

TOAST

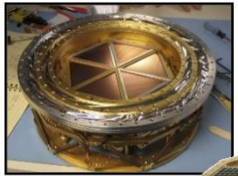
(T. Kisner, J. Madsen; D. Liu, V. Litvinov)

Cosmic Microwave Background: HPC and Data



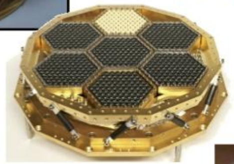
2005

APEX-SZ
330 detectors



2007

SPT-SZ
960 detectors

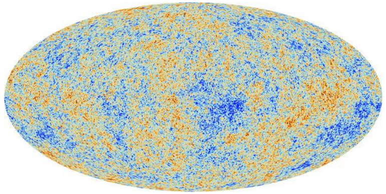


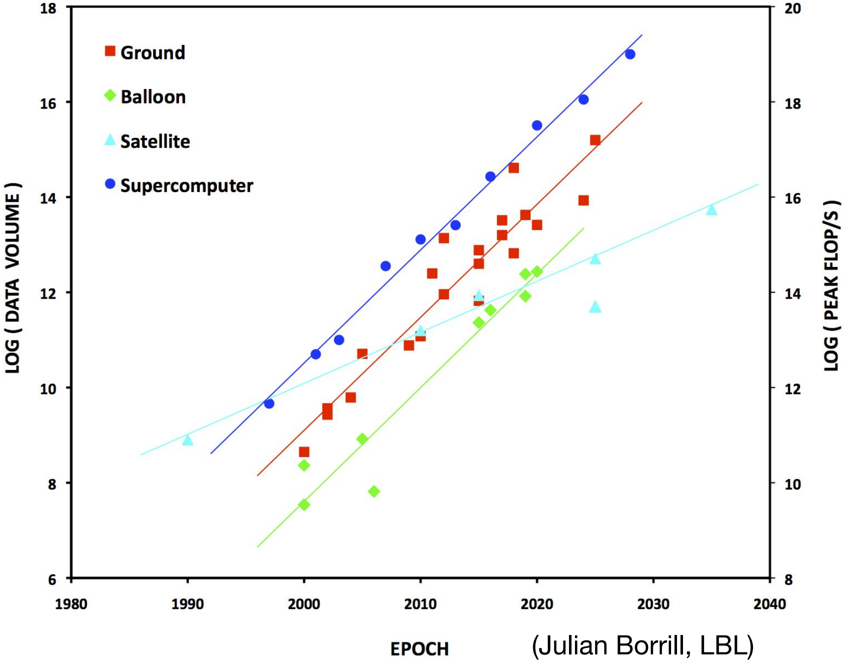
2011

POLARBEAR-1
1274 detectors
Dual-Polarization

2017

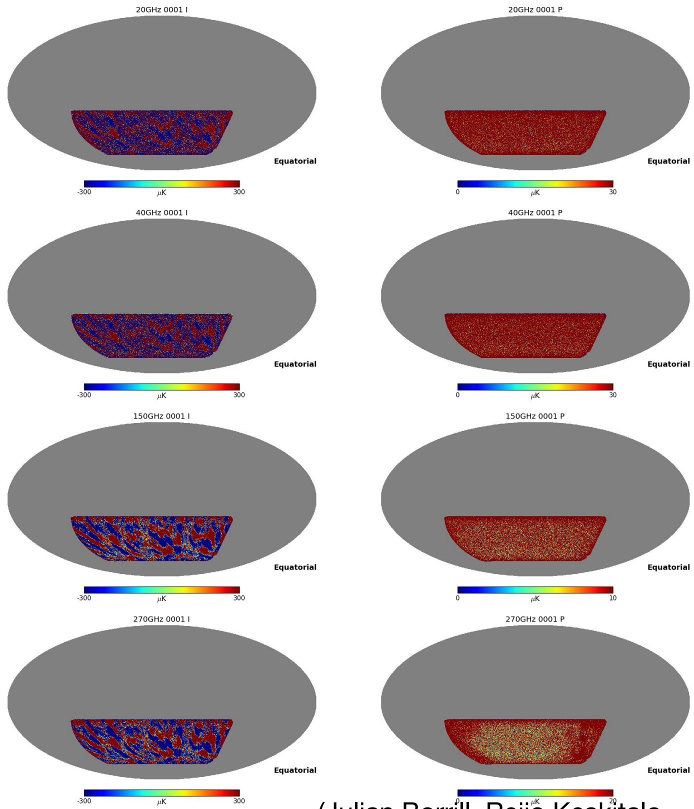
POLARBEAR-2/SPT-3G
8,000/15,000 detectors
Dual-Polarization
2-3 Color/pixel





(Julian Borrill, LBL)

Preparations for CMB-S4



(Julian Borrill, Reijo Keskitalo,
Ted Kisner, LBL)

Future CMB experiments:

- 10x more detectors
- More telescopes
- New systematics (weather at different sites)

"TOAST" App Hero Run in June 2017

Simulation and map-making:

- 50,000 detectors
- 30X Planck mission data
- 1 year of simulated observations from Chile
- Including effects of atmosphere

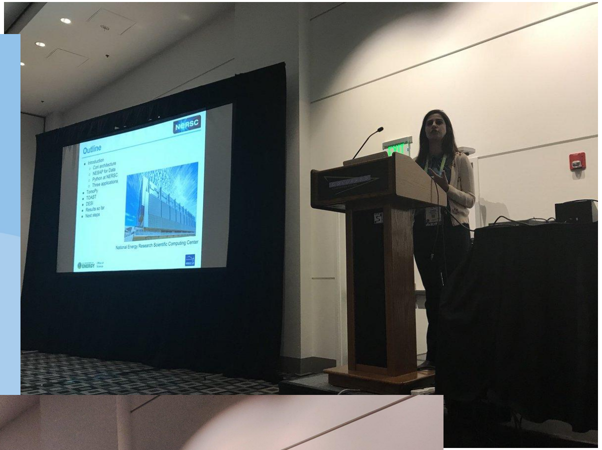
Using all of NERSC's Cori Phase II:

- 658,784 Intel Xeon Phi (Knights Landing) Cores
- Hybrid MPI/OpenMP (150K MPI ranks)
- First full-KNL production Shifter (container) job
- Hybrid Python/C++

PyHPC 2018

At SC18!

SC18
Dallas, TX | hpc inspires.



8th Workshop on Python for High-Performance and Scientific Computing

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Conclusion

Python fills numerous critical roles at HPC scientific computing centers like NERSC.

Especially true in experimental/observational sciences, data processing/analysis more than analytics for now.

Achieving good Python performance is challenging and users (not often HPC-oriented) need to partner with center staff and vendors/developers to get it.