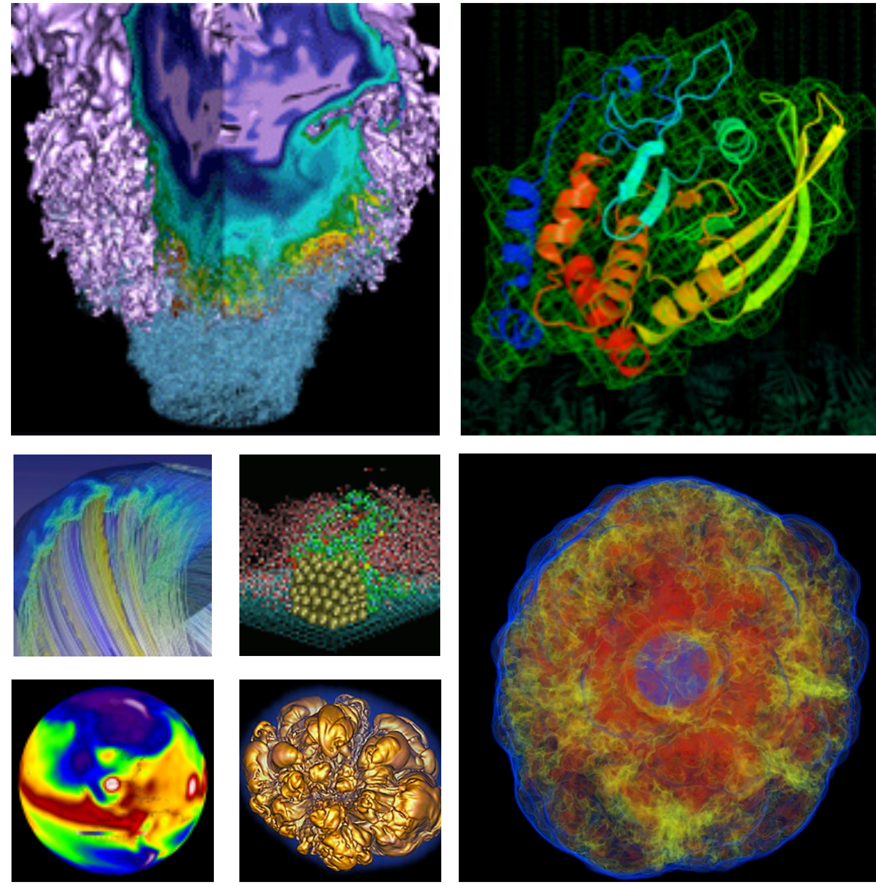
NUG Monthly Meeting



Katie Antypas, Richard Gerber, Jack Deslippe

NUG Monthly Meeting
June 5, 2014

NERSC is moving to our new building in 2015!!

- **Four story, 140,000 GSF**
  - 300 offices on two floors
  - 20K -> 29Ksf HPC floor
  - 12.5MW -> 42 MW to building
- **Located for collaboration**
  - CRD and ESnet
  - UC Berkeley
- **Exceptional energy efficiency**
  - Natural air and water cooling
  - Heat recovery
  - PUE < 1.1
  - LEED gold design

Computational Research and Theory Building

2015 will be a busy year for NERSC

- **Move plans are still being finalized and we will share more information as soon as possible.   Here is our current thinking:**
  - Up to 6 weeks of Edison downtime in Q3 or Q4 2015
  - 2-3 weeks downtime for HPSS
  - Periods of slow I/O on NGF as data is synced between Oakland and new facility
  - Some Intermittent disruptions on other services
- **System retirements**
  - Dirac (GPU testbed) will retire Dec. 12, 2014
  - Carver will retire in our Oakland facility Aug. 31, 2015
  - Hopper will retire in our Oakland facility Sept. 30th 2015
- **We will not leave you without a system!**
  - Looking at options to fill gap between Hopper's retirement and Cori delivery (June 2015)
  - Goal is to have Edison or another system running before Hopper's retirement

We ask for your patience during our move!

Tuesday we launched the NERSC Exascale Science Application Program

- Umbrella program for all NERSC Application Readiness Activities
- Approximately 20 application teams will be accepted into NESAP
- Each application team will be partnered with a member of NERSC's App Readiness team who will assist with code profiling and scaling analyses
- Through this program NERSC will allocate resources from Cray and Intel
- 8 application teams will receive NERSC funded Post-docs

NERSC Exascale Science Applications Program (NESAP)

Application teams in NESAP will have access to the following:

- A partner from NERSC's Application Readiness team who will assist with code profiling and optimization
- Access to Cray and Intel resources to help with code optimization.
- Up to 1M MPP hours in 2014 and 2M MPP hours in 2015 for code testing, optimization, scaling and debugging on Edison
- Early access to prototype Knights Landing processor hardware (expected in late 2015)
- Early access and significant hours on the full Cori system (expected delivery mid-2016)
- Opportunity for a Post-doctoral researcher to be placed within your application team. (NERSC will fund 8 Post-doctoral researchers and place each one within one of the 20 NESAP teams meaning that approximately 40% of NESAP applications teams will include a NERSC sponsored Post-doc.)

NERSC Exascale Science Applications Program (NESAP)

Application teams in NESAP are responsible for:

- Working with your NERSC Application Readiness partner to produce profiling and scaling plots as well as vectorization and memory bandwidth analyses.
- Assigning someone in your group to work on optimizing, refactoring, testing, and further profiling your code to transition to the Cori node architecture.
- Producing an intermediate and final report detailing the application's science and performance improvement as a result of the collaboration

NERSC Exascale Science Applications Program (NESAP)

NERSC will use the following criteria to evaluate submissions:

- An application's computing usage within the DOE Office of Science
- Representation among all 6 Offices of Science
- Ability for application to produce scientific advancements
- Ability for code development and optimizations to be transferred to the broader community through libraries, algorithms, kernels or community codes
- Resources available from the application team to match NERSC/Vendor resources

NERSC is committed to helping our users

Help transition the NERSC workload to future architectures by exploring and improving application performance on manycore architectures.

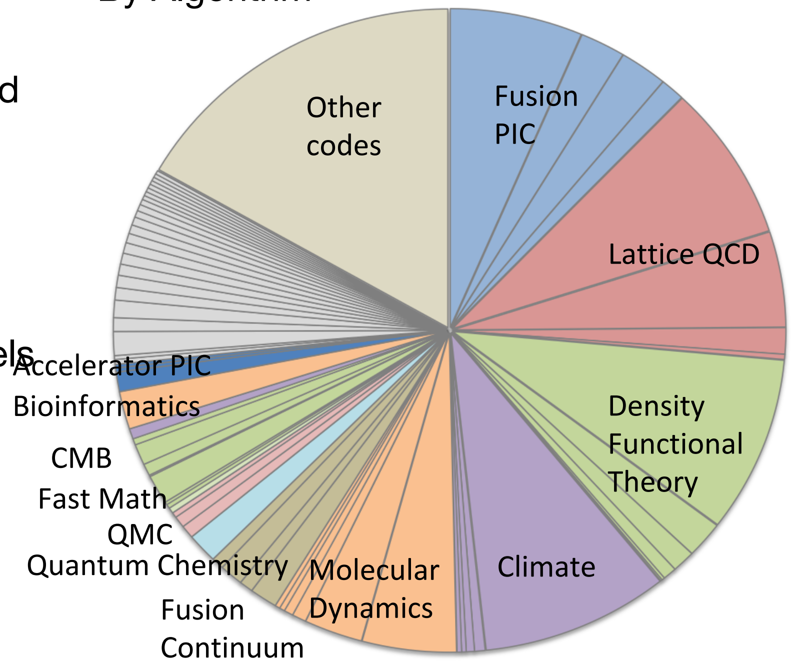
Phase 1:

- ➔ Identify major algorithms in the NERSC workload. Assigned 14 codes to represent class.
  - 1 team member per code
- ➔ Code status discovery
  - What has been done at other centers
  - How are various code teams preparing
- ➔ Profile OpenMP/MPI scaling and vectorization in key kernels on GPU testbed (dirac) and Xeon-Phi testbed (babbage).

Phase 2:

- ➔ NESAP - Exascale Application Program (+PostDocs)
- ➔ Organize user training around node-parallelism, vectorization and other KNL strategies.
- ➔ Application deep dives with Cray and Intel.
- ➔ Meet with key application developers / workshops at NERSC and leverage 3rd party efforts.

NERSC Workload
By Algorithm



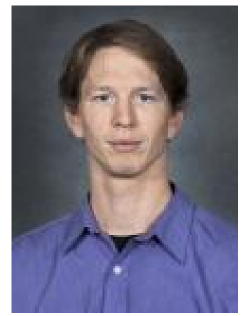
NERSC App Readiness Team in Phase 1

NERSC is kicking off an "Application Readiness" effort. Devoting significant staff effort to help users and developers port their codes to many-core architectures

Katerina Antypas
(Co-Lead)

Nick Wright (Co-Lead)
Amber (Proxy for NAMD, LAMMPS)

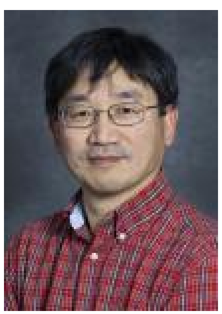
Harvey Wasserman
SNAP (S_N transport proxy)



Brian Austin
Zori (Proxy for QWalk etc.)

Hongzhang Shan
NWChem (Proxy for qchem, GAMESS)

Aaron Collier
Madam-Toast / Gyro



Woo-Sun Yang
CAM (Proxy for CESM)

Jack Deslippe
Quantum ESPRESSO / BerkeleyGW (Proxy for VASP, Abinit)

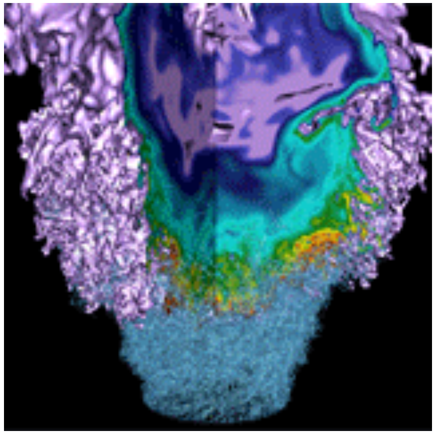
Helen He
WRF

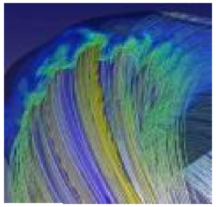
Matt Cordery
MPAS

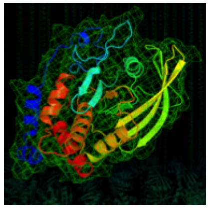
Kirsten Fagnan
Bio-Informatics

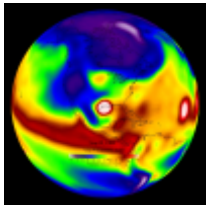
Christopher Daley
FLASH

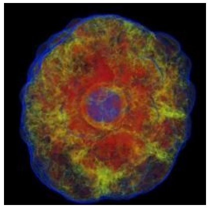
BerkeleyGW Case Study

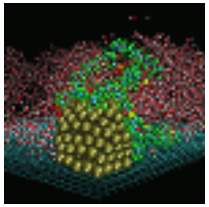












NERSC 40 YEARS at the FOREFRONT

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Case Study: BerkeleyGW

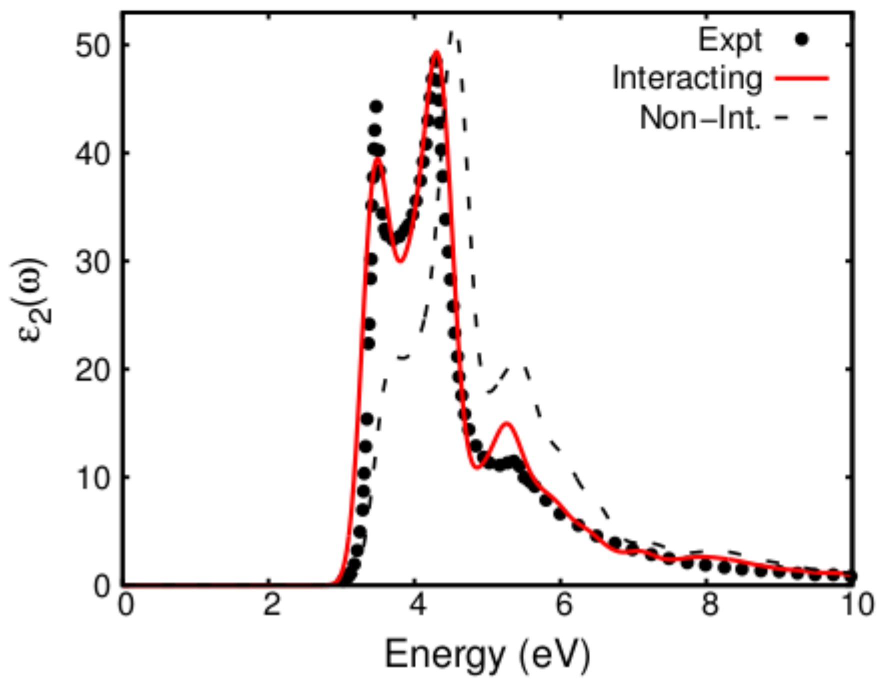
Description:

A material science code to compute excited state properties of materials. Works with many common DFT packages.

Algorithms:

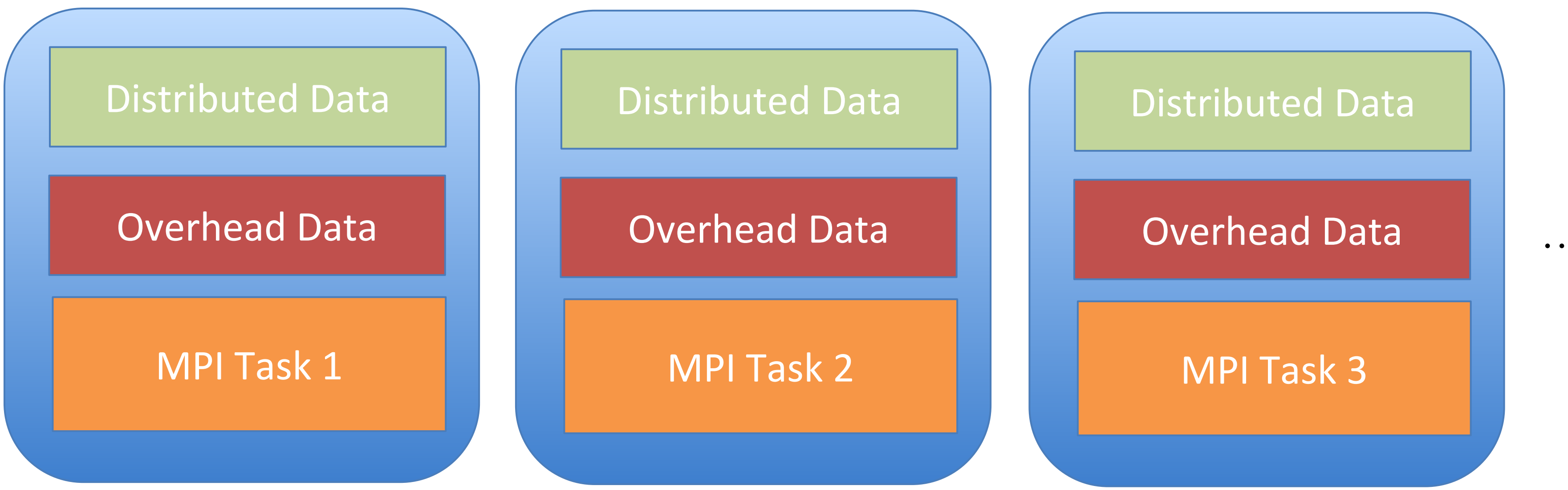
- FFTs (FFTW)
- Dense Linear Algebra (BLAS / LAPACK / SCALAPACK / ELPA)
- Large Reduction Loops.

Silicon Light Absorption vs. Photon Energy as Computed in BerkeleyGW

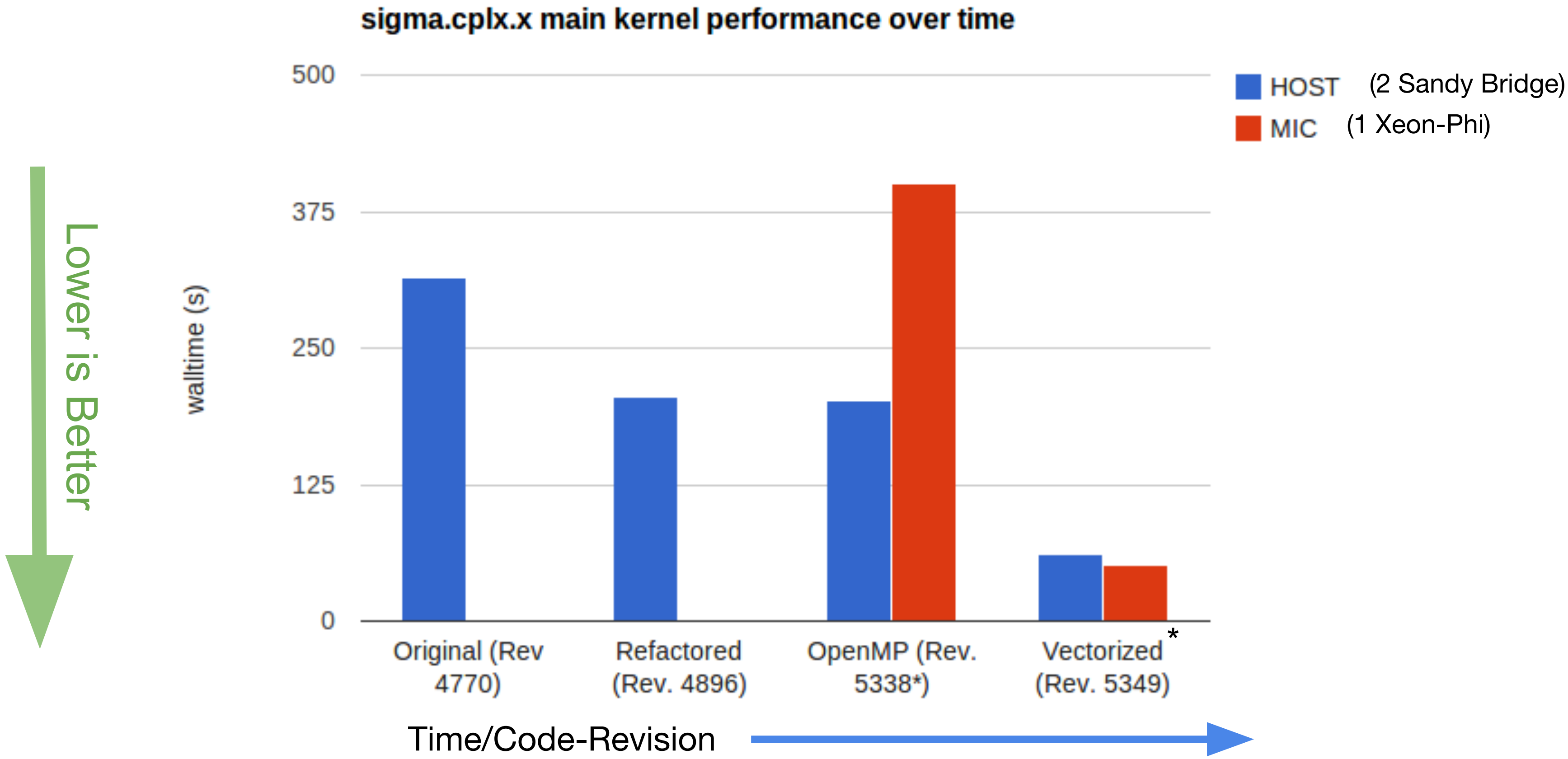


Failure of the MPI-Only Programming Model in BerkeleyGW

- ★ Big systems require more memory. Cost scales as N_{atm}^2 to store the data.
- ★ In an MPI GW implementation, in practice, to avoid communication, data is duplicated and **each MPI task has a memory overhead**.
- ★ On Hopper, users often forced to use 1 of 24 available cores, in order to provide MPI tasks with enough memory. **90% of the computing capability is lost**.



Steps to Optimize BerkeleyGW on Xeon-Phi Testbed



1. Refactor to create hierarchical set of loops to be parallelized via MPI, OpenMP and Vectorization and to improve memory locality.
2. Add OpenMP at as high a level as possible.
3. Make sure large innermost, flop intensive, loops are vectorized

* - eliminate spurious logic, some code restructuring simplification and other optimization

Simplified Final Loop Structure

```
!$OMP DO reduction(+:achtemp)
  do my_igp = 1, ngpown

    ...

    do iw=1,3

      scht=0D0
      wxt = wx_array(iw)

      do ig = 1, ncouls

        !if (abs(wtilde_array(ig,my_igp) * eps(ig,my_igp)) .lt. TOL) cycle

        wdiff = wxt - wtilde_array(ig,my_igp)
        delw = wtilde_array(ig,my_igp) / wdiff

        ...

        scha(ig) = mygpvar1 * aqsntemp(ig) * delw * eps(ig,my_igp)

        scht = scht + scha(ig)

      enddo ! loop over g

      sch_array(iw) = sch_array(iw) + 0.5D0*scht

    enddo

    achtemp(:) = achtemp(:) + sch_array(:) * vcoul(my_igp)

  enddo
```

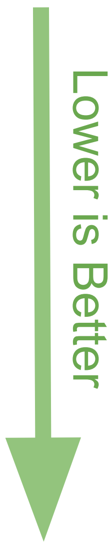
ngpown typically in 100's to 1000s. Good for many threads.

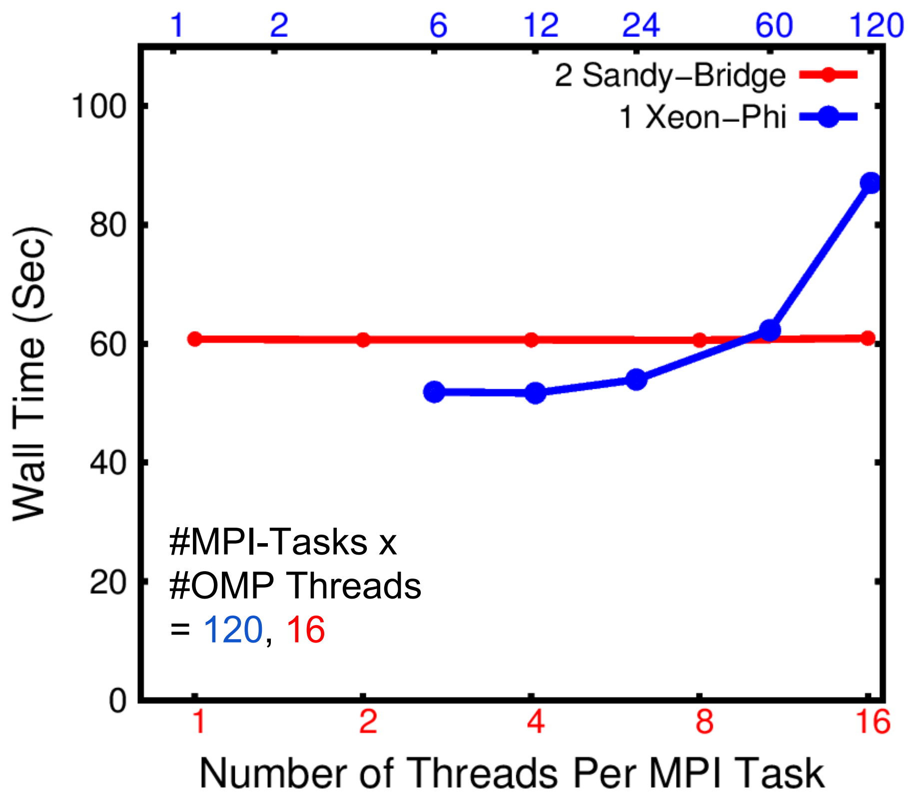
Original inner loop. Too small to vectorize!

ncouls typically in 1000s - 10,000s. Good for vectorization. Don't have to worry much about memory. alignment.

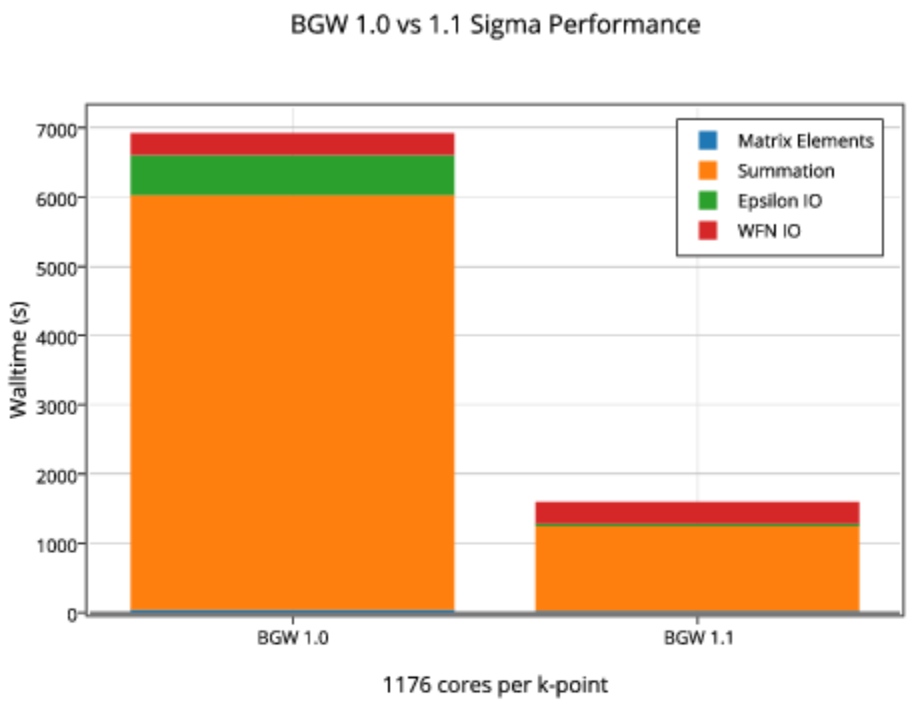
Attempt to save work breaks vectorization and makes code slower.

Running on Many-Core Xeon-Phi Requires OpenMP Simply To Fit Problem in Memory

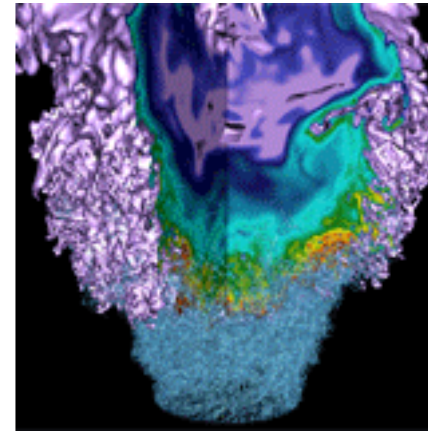


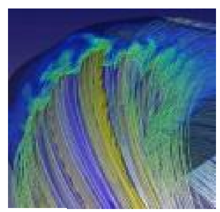


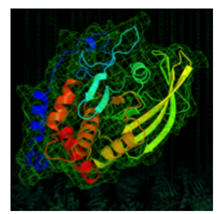
Meanwhile, Back on Edison…

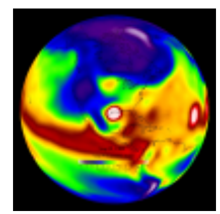


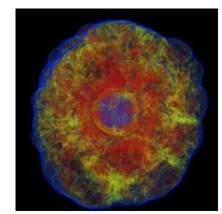
Conclusions and Lessons Learned

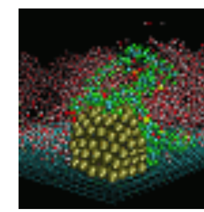












Summary

- ➔ Change is Coming!
- ➔ NERSC is Here to Help Our Users
- ➔ Good performance will require code changes
  - ◆ Identify more on-node parallelism
  - ◆ Ensure vectorization for critical loops
- ➔ Need to leverage community. Other centers, NERSC users, 3rd Party Developers
- ➔ The code changes you make for many-core architectures will improve performance on all architectures.