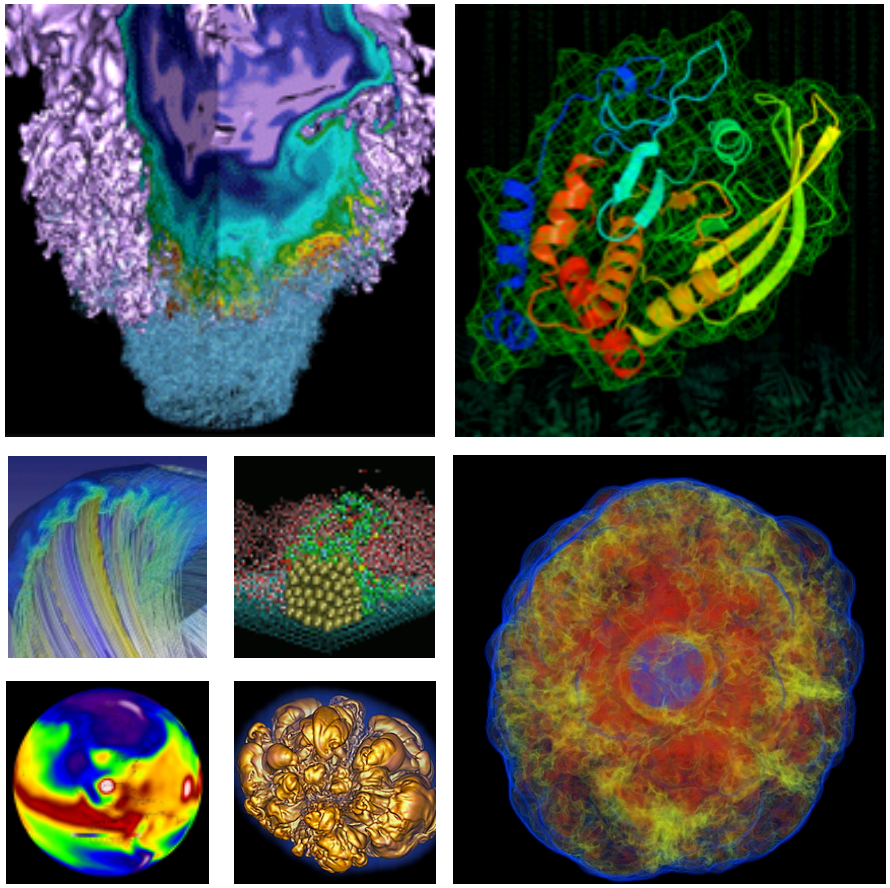
NERSC-8 Update for HiPACC



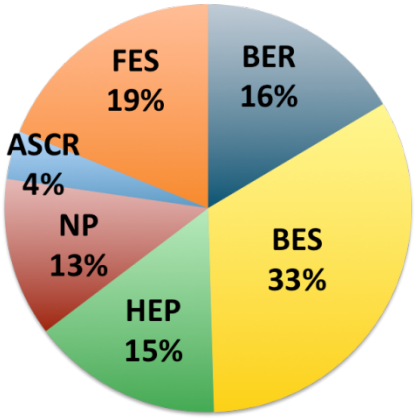
Katie Antypas

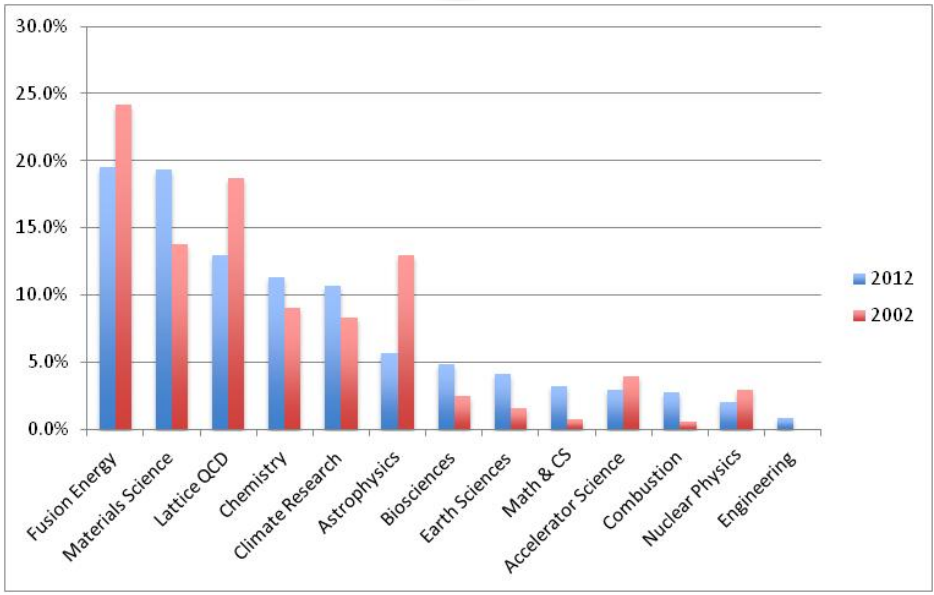
NERSC-8 Project Manager
NERSC Services Dept. Head

HiPACC Meeting
March 22, 2014

We are the primary computing facility for the DOE Office of Science

- **DOE SC allocates the vast majority of the computing and storage resources at NERSC**
  - Six program offices allocate their base allocations and they submit proposals for overtargets
  - Deputy Director of Science prioritizes overtarget requests
- **Usage shifts as DOE priorities change**
- **Over 5000 users and 700 projects run at NERSC**





Systems Strategy

- **We have two large systems on the floor to provide stability and continuity for users**
- **Our strategy is:**
  - Open competition for best solutions
  - Focus on the performance of a broad range of applications, not synthetic benchmarks
  - General-purpose architectures are needed in order to support a wide range of applications, both large-scale simulations and high volumes of smaller simulations
  - Earlier procurements to influence designs
  - Leverage Fast Forward and Design Forward
  - Engage co-design efforts
  - Transition users to a new manycore architectures

NERSC-8 Mission Need approved in November 2012

Acquire and deploy HPC resources for the rapidly increasing computational demands of DOE SC research community.

- Provide a significant increase in computational capabilities over the Hopper system, at least 10x on a set of representative DOE benchmarks
- Platform needs to begin to transition users to more energy-efficient many-core architectures.
- Provide high bandwidth access to existing data stored by continuing research projects.
- System delivery in the 2015/2016 timeframe

Edison, a Cray XC-30 plays a key role in NERSC's strategy

- NERSC assessed that our broad workload was not ready for GPUs and procured Edison, with Ivy Bridge Intel CPUs
- Workloads that have difficulty moving to NERSC-8 can still work productively on Edison while the code is adapted
- In 2016 Edison will likely provide ~20% of NERSC's cycles

Programming Models Strategy

- **The necessary characteristics for broad adoption of a new pmodel is**
  - Performance: At least 10x-50x performance improvement
  - Portability: Code performs well on multiple platforms
  - Durability: Solution must be good for a decade or more
  - Availability/Ubiquity: Cannot be a proprietary solution
- **Our near-term strategy is**
  - Smooth progression to exascale from a user's point of view
  - Support for legacy code, albeit at less than optimal performance
  - Support for a variety of programming models
  - Support optimized libraries

Although architecture for NERSC-8 is not yet known, trend for all 2015/2016 systems is manycore

- **Regardless of processor architecture, users will need to modify applications to achieve performance**
  - Expose more on-node parallelism in applications
  - Increase application vectorization capabilities
  - For co-processor architectures, locality directives must be added
  - Hierarchical memory

NERSC workload has hundreds of codes.
How will application teams make the transition?

NERSC App Readiness Team

Katerina Antypas
(Co-Lead)

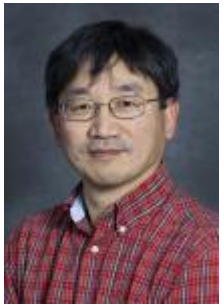
Nick Wright (Co-Lead)
Amber (Proxy for NAMD, LAMMPS)

Harvey Wasserman
SNAP (S_N transport proxy)

Brian Austin
Zori (Proxy for QWalk etc.)

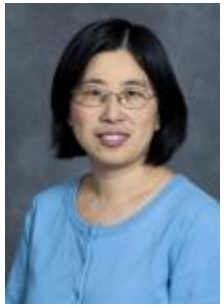
Hongzhang Shan
NWChem (Proxy for qchem, GAMESS)

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

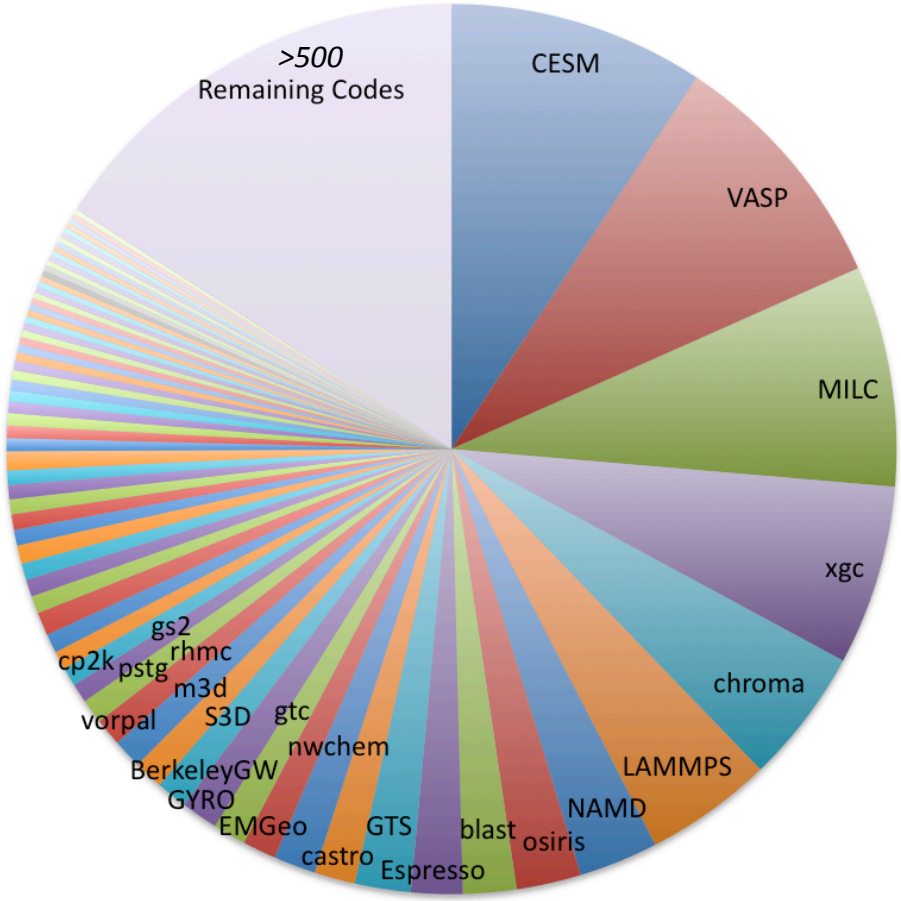
Kirsten Fagnan
Bio-Informatics

Christopher Daley
FLASH

- If you do nothing, your MPI-only code may run poorly on future machines.
- NERSC is here to help

NERSC's highly concentrated workload, but astro codes are not

Breakdown of Application Hours on Hopper 2012



- **10 codes make up 50% of the workload**
- **25 codes make up 66% of the workload**

ART
GADGET
HOT
TreePM
HACC
RAMSES
SPH
MESA

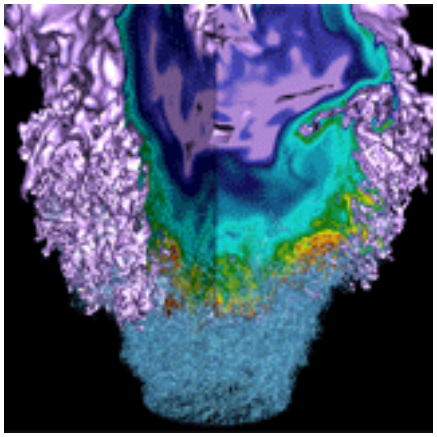
KEPLER
CASTRO
SEDONA
SNSPH
NYX
Madam_toast
CHIMERA
FLASH
PARSEC

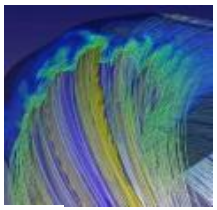
NERSC Application Readiness Effort

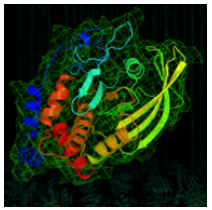
- **NERSC has a robust application transition plan in place which will help transition users to the NERSC-8 architecture**

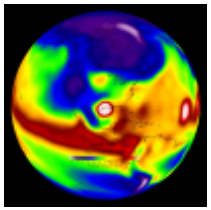
NERSC is committed to helping our users make this transition

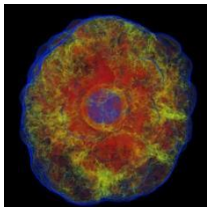
FLASH Case Study
Christopher Daley

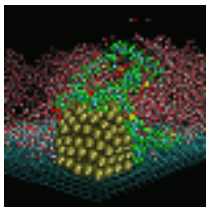












NERSC 40 YEARS at the FOREFRONT

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB
Lawrence Berkeley National Laboratory

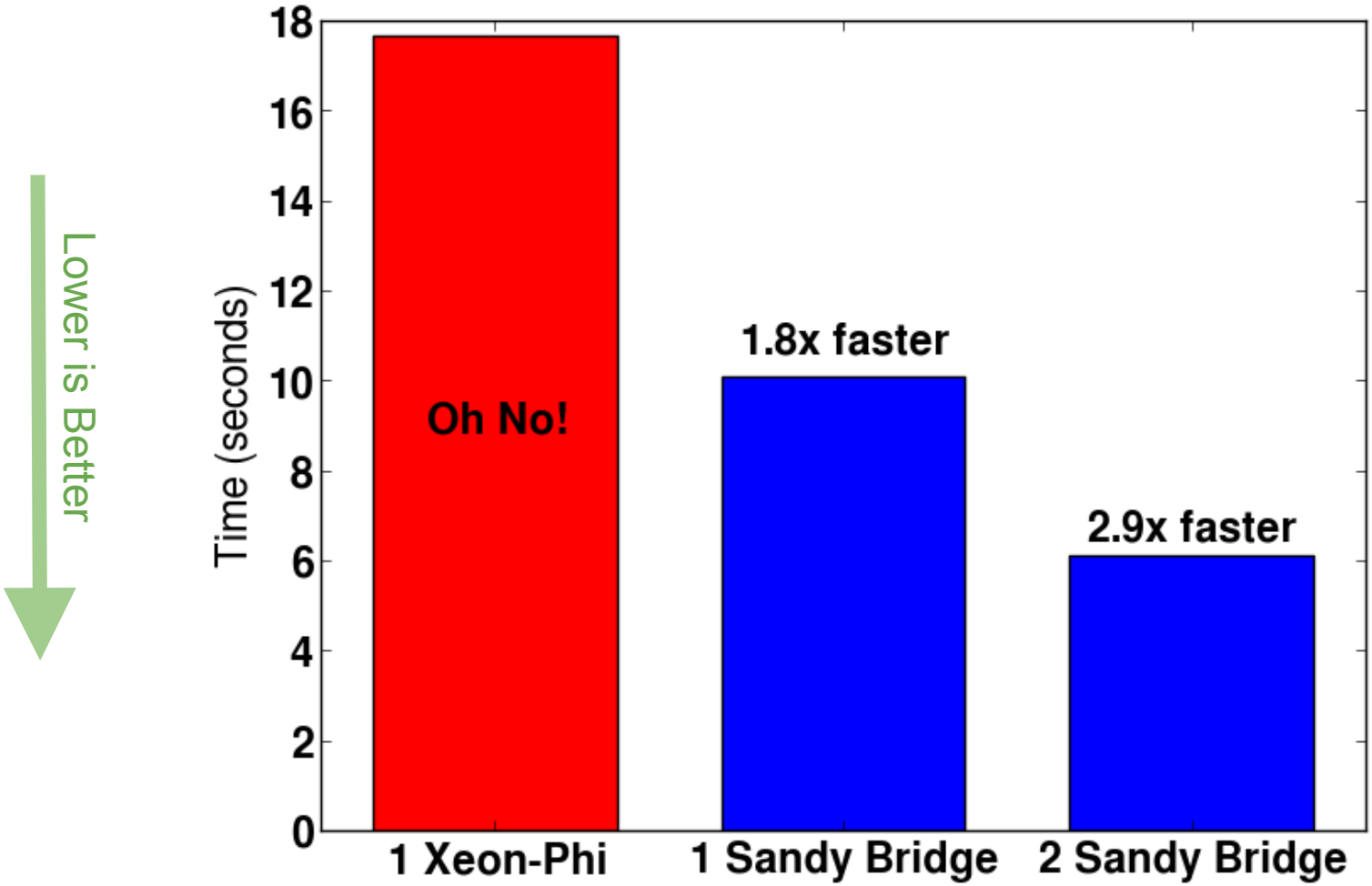
Case Study on the Xeon-Phi (MIC) Architecture

- NERSC Testbed Babbage
  - 45 Sandy-bridge nodes with Xeon-Phi Co-processor
  - Each Xeon-Phi Co-processor has
    - 60 cores
    - 4 HW threads per core
    - 8 GB of memory
  - Multiple ways to program with co-processor
    - As an accelerator
    - As a self-hosted processor (ignore Sandy-bridge)
    - Reverse accelerator
- We chose to test as if the Xeon-Phi was a stand alone processor because Intel has announced next generation will be self-hosted

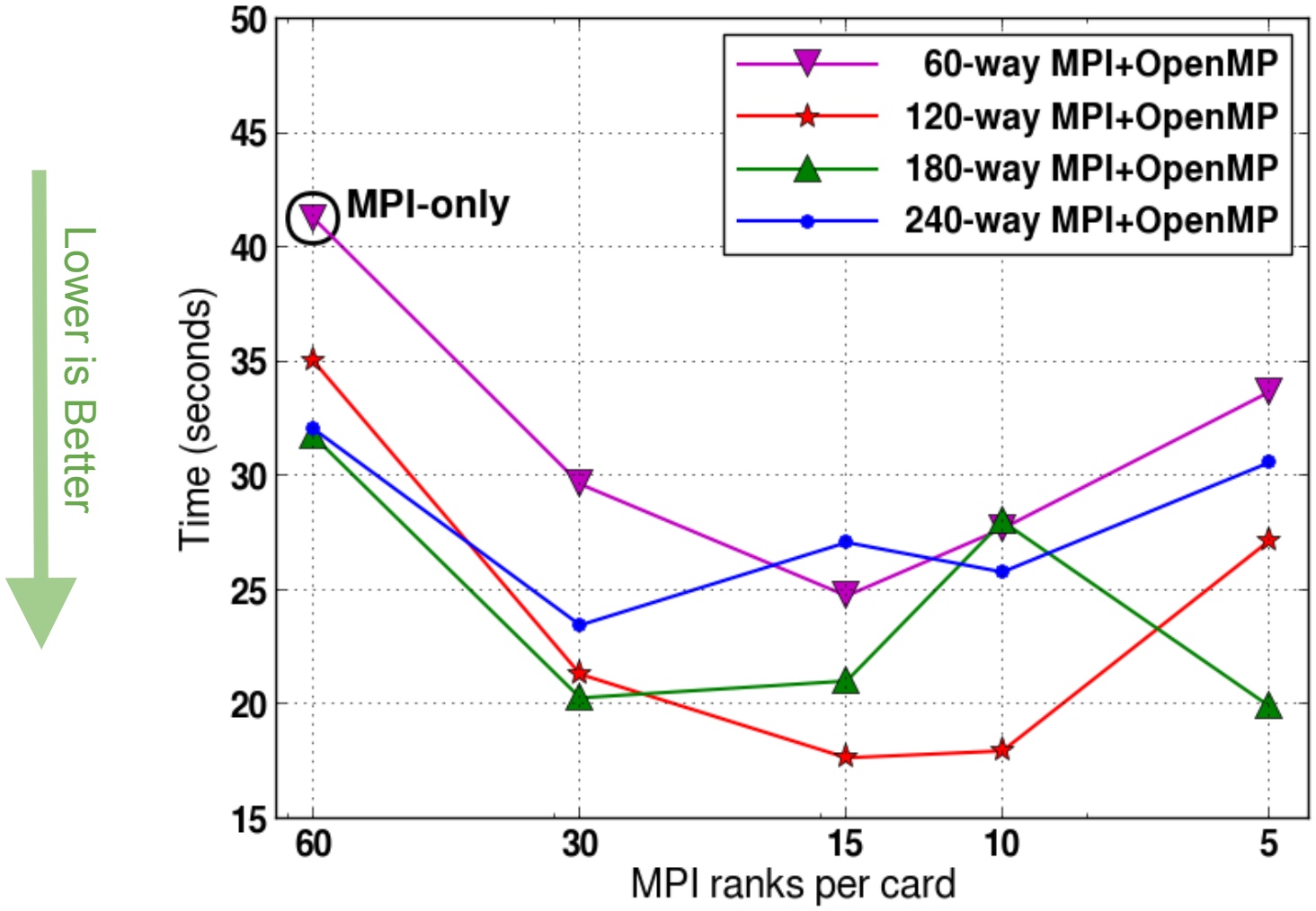
FLASH application readiness

- **FLASH is an Adaptive Mesh Refinement (AMR) code with explicit solvers for hydrodynamics and magneto-hydrodynamics**
- **Parallelized using**
  - MPI domain decomposition AND
  - OpenMP multithreading over either local domains or over cells in each local domain
- **Target application is a 3D Sedov explosion problem**
  - A spherical blast wave is evolved over multiple time steps
  - We test a configuration with a uniform resolution grid (and not AMR) and use 100^3 global cells
- **The hydrodynamics solvers perform large stencil computations.**

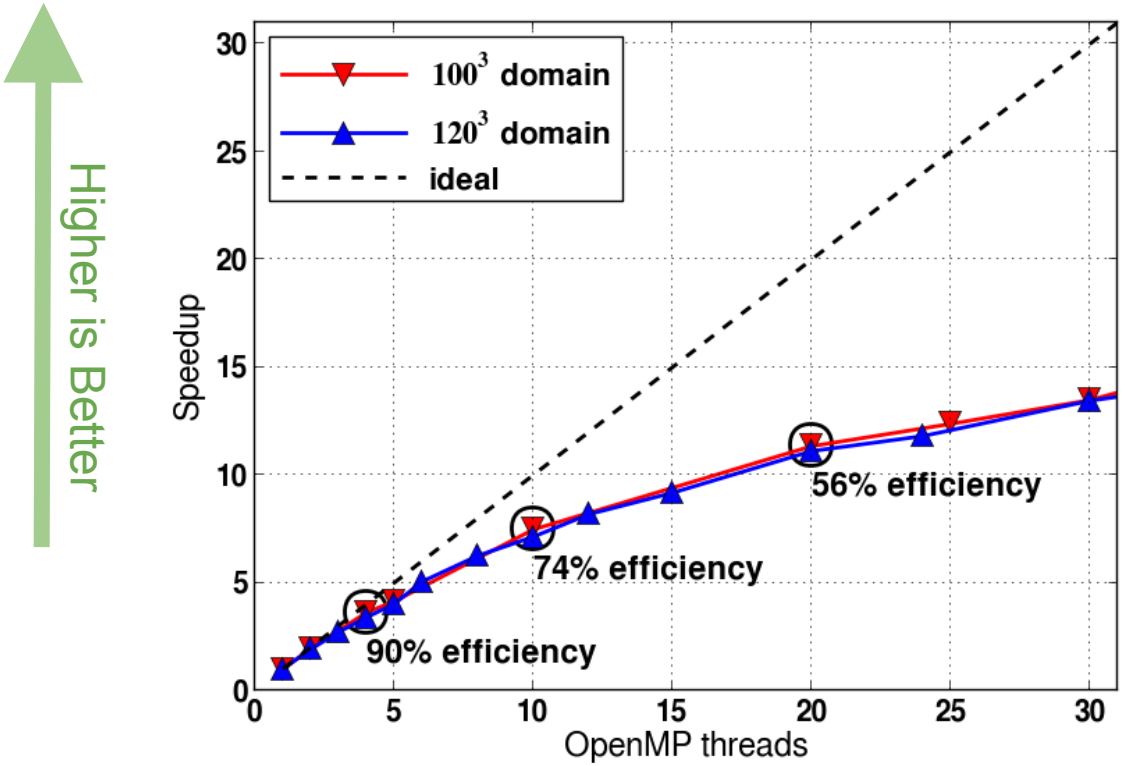
Best MIC performance vs host



Best configuration on 1 MIC card



MIC performance study 1: thread speedup



- 1 MPI rank per MIC card and various numbers of OpenMP threads
- Each OpenMP thread is placed on a separate core
- 10x thread count ideally gives a 10x speedup

- Speedup is not ideal
  - But it is not the main cause of the poor MIC performance
  - ~70% efficiency @ 12 threads (as would be used with 10 MPI ranks per card)

Vectorization is another form of on-node parallelism

```
do i = 1, n
     a(i) = b(i) + c(i)
enddo
```

$$\begin{pmatrix} a_1 \\ \dots \\ a_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \dots \\ b_n \end{pmatrix} + \begin{pmatrix} c_1 \\ \dots \\ c_n \end{pmatrix}$$

Intel Xeon Sandy-Bridge/Ivy-Bridge: 4 Double Precision Ops Concurrently

Intel Xeon Phi: 8 Double Precision Ops Concurrently

NVIDIA Kepler GPUs: 32 SIMT threads

Things that Kill Vectorization

Compilers want to "vectorize" your loops whenever possible. But sometimes they get stumped. Here are a few things that prevent your code from vectorizing:

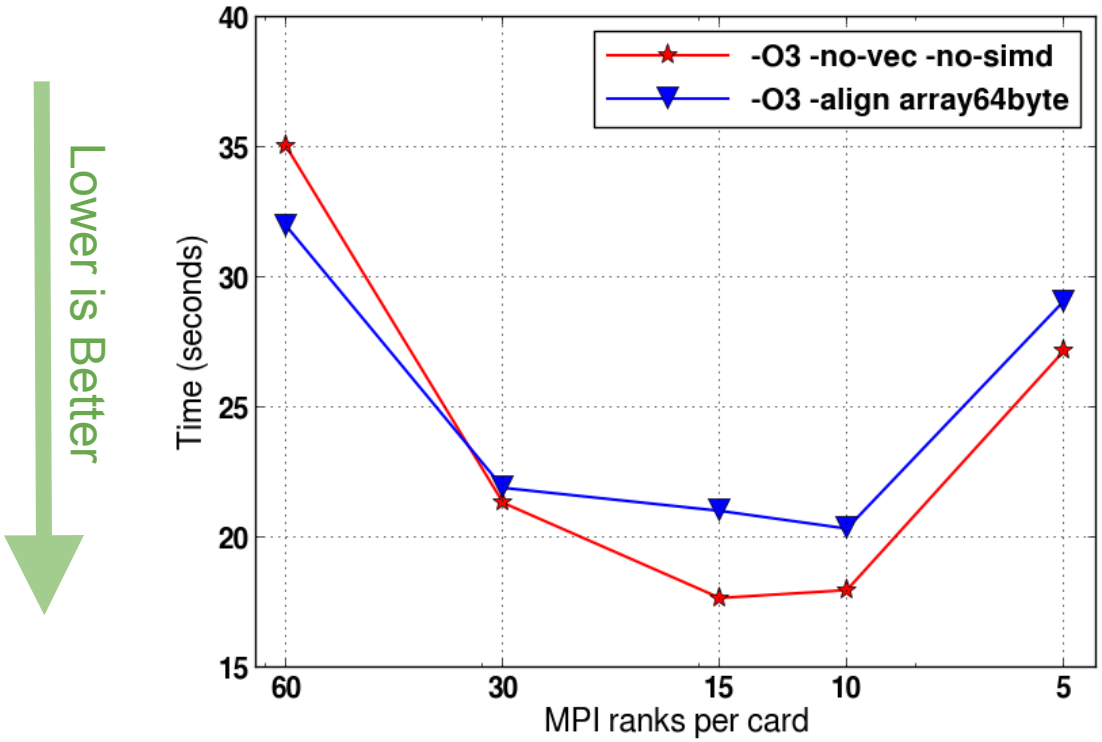
Loop dependency:

```
do i = 1, n
    a(i) = a(i-1) + b(i)
enddo
```

Task forking:

```
do i = 1, n
    if (a(i) < x) cycle
    ...
enddo
```

MIC performance study 2: vectorization



No vectorization gain!

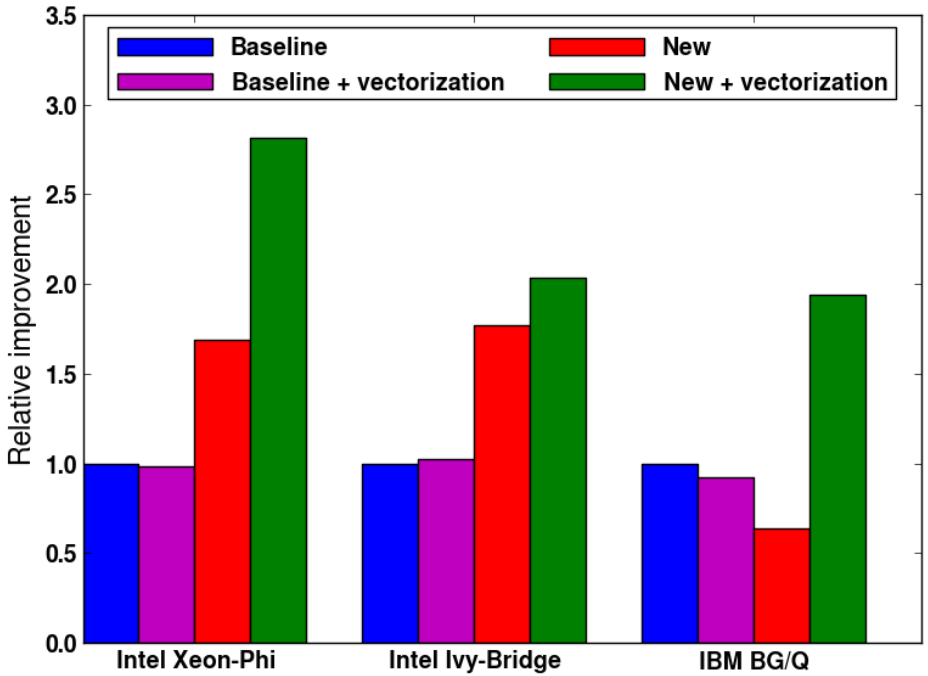
- We find that most time is spent in subroutines which update fluid state 1 grid point at a time

- The data for 1 grid point is laid out as a structure of fluid fields, e.g. density, pressure, …, temperature next to each other: A(HY_DENS:HY_TEMP)
- Vectorization can only happen when the same operation is performed on multiple fluid fields of 1 grid point!

Enabling vectorization

- **Must restructure the code**
  - The fluid fields should no longer be next to each other in memory
  - A(HY_DENS:HY_TEMP) should become A_dens(1:N), …, A_temp(1:N)
    - The 1:N indicates the kernels now operate on N grid points at a time
- **We tested these changes on part of a data reconstruction kernel**

Higher is Better



- **The new code compiled with vectorization options gives the best performance on 3 different platforms**

Summary

Good Parallel Efficiency AND Vectorization = Good MIC Performance

- **FLASH on MIC**
  - MPI+OpenMP parallel efficiency – OK
  - Vectorization – **zero / negative gain** …must restructure!
    - Compiler auto-vectorization / vectorization directives do not help the current code
- **Changes needed to enable vectorization**
  - Make the kernel subroutines operate on multiple grid points at a time
  - Change the data layout by using a separate array for each fluid field
    - Effectively a change from array of structures (AofS) to structure of arrays (SofA)
- **Tested these proof-of-concept changes on a reduced hydro kernel**
  - Demonstrated improved performance on Ivy-Bridge, BG/Q and Xeon-Phi platforms

Summary

➔ **Change is Coming!**

➔ **NERSC is Here to Help Our Users**

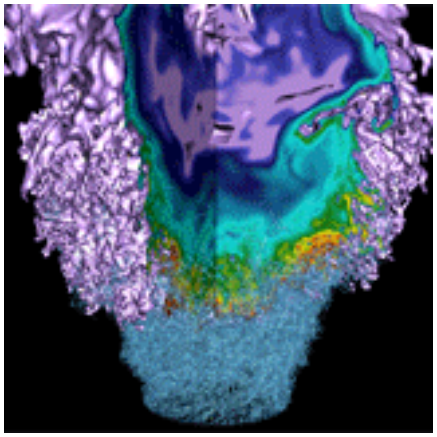
➔ **Good performance will require code changes**

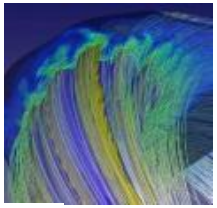
- Identify more on-node parallelism
- Ensure vectorization for critical loops

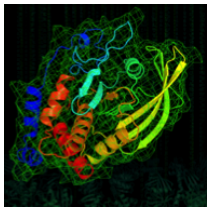
➔ **Need to leverage community. Other centers, NERSC users, 3rd Party Developers**

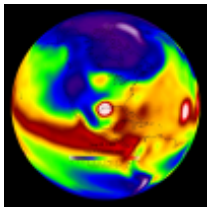
➔ **The code changes you make for many-core architectures will improve performance on all architectures.**

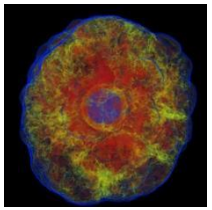
Burst Buffer

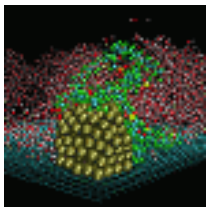












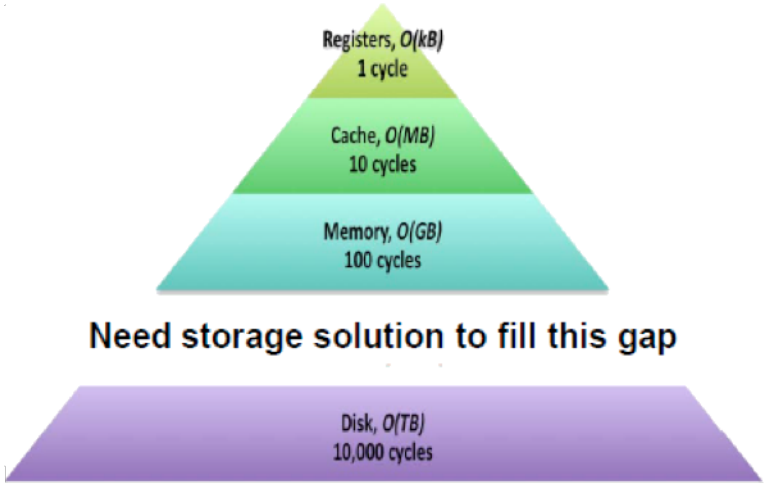
NERSC 40 YEARS at the FOREFRONT

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB
Lawrence Berkeley National Laboratory

Burst Buffer

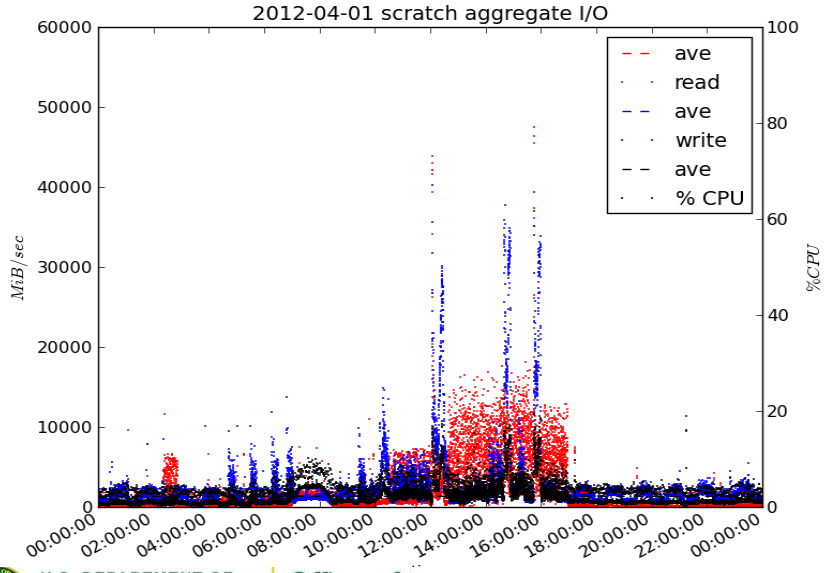
- **Flash storage which would act as a cache to improve peak performance of the PFS.**

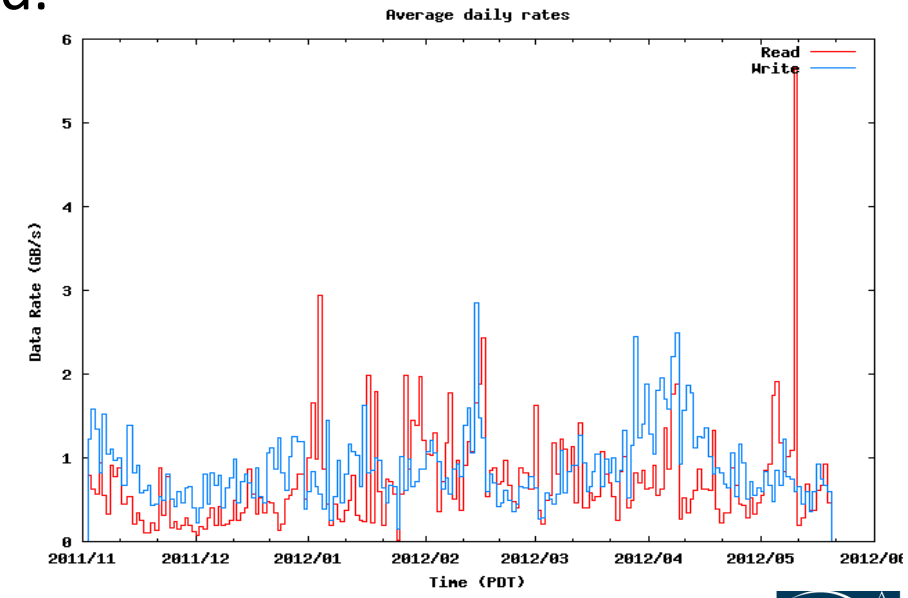


- **Flash is currently as little as 1/6 the cost of disk per GB/s bandwidth and has better random access characteristics(no seek penalty).**

Burst Buffer and why we're interested

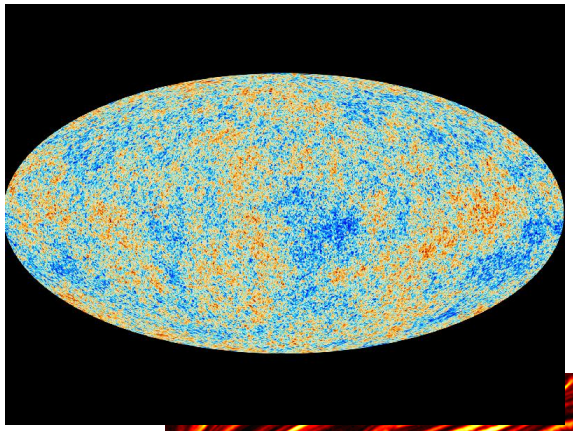
- **NERSC I/O is bursty**
  - While the Filesystem is regularly pushed to near it's peak bandwidth, most of the time it's utilization is well below it's full capability. If peak BW requirements can be adequately met by a flash cache, the file system can be more modestly provisioned.

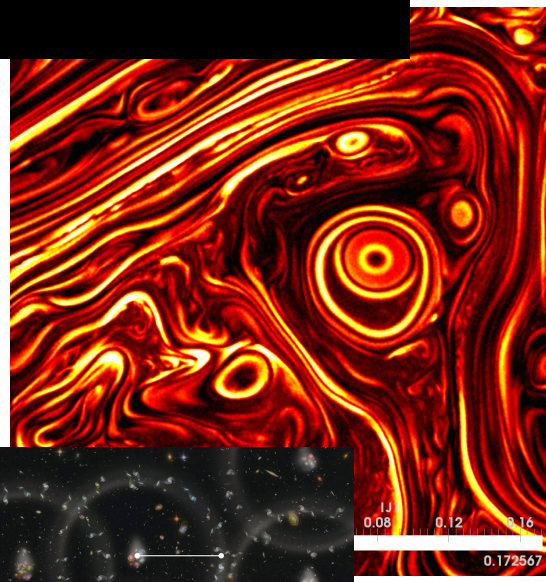


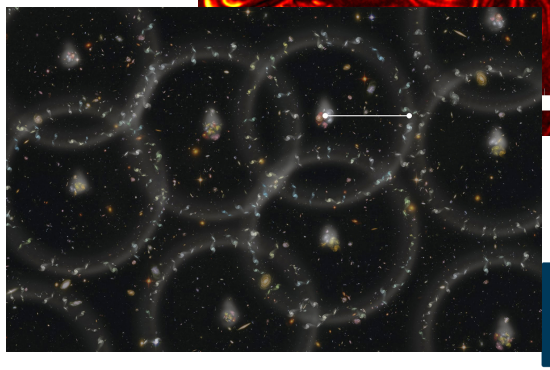


We are proud to support outstanding Computational Astrophysics at NERSC

- 1977 journal publications in 2013
- Simulations at NERSC were key to 3 Nobel Prizes (2007, 2011, 2013)
- Supernova 2011fe was caught within hours of its explosion in 2011 and telescopes from around the world were redirected to it the same night
- Discovering if 'rare earths' are really rare? PNAS. Erik Petigura (UC Berkeley)
- BOSS measures accuracy of universe to 1%
- Stellar Burst before Supernova (Palomar Transient Factory) Ofek Nature.
- CMB mapping from Planck satellite mission computing



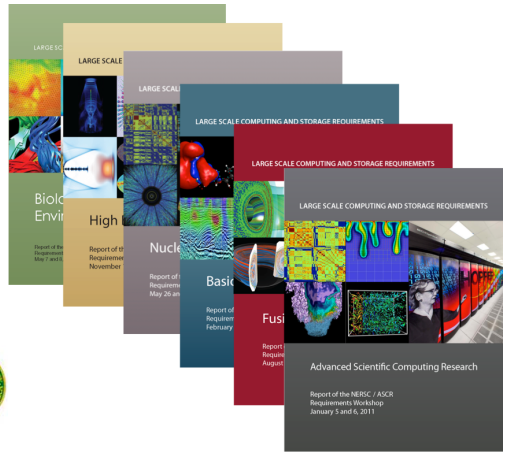


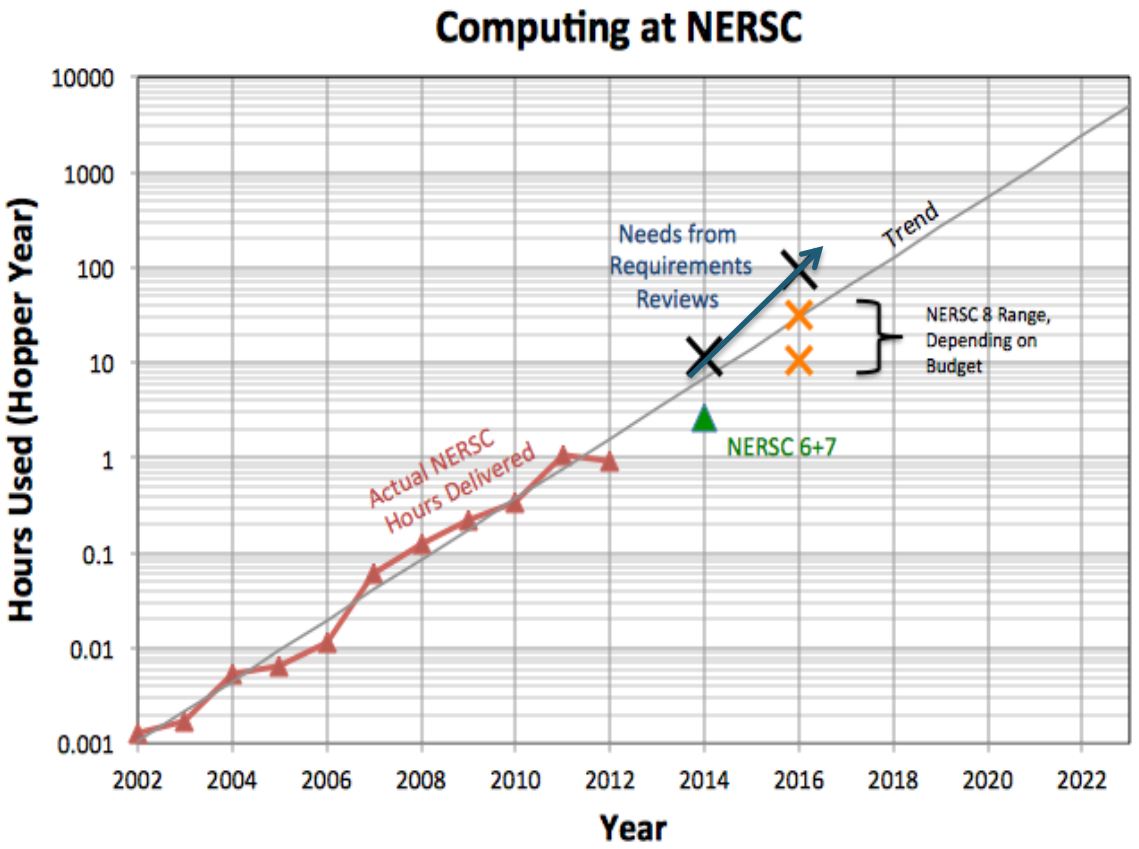


Thank you

Requirements Workshops describe a critical need for a significant increase in HPC resources

- One workshop and report per office, organized by ASCR and NERSC
- Case studies on science needs
- Estimated 2016 need over 47 times **Hopper** capability
- **NERSC-7 will not fulfill need**

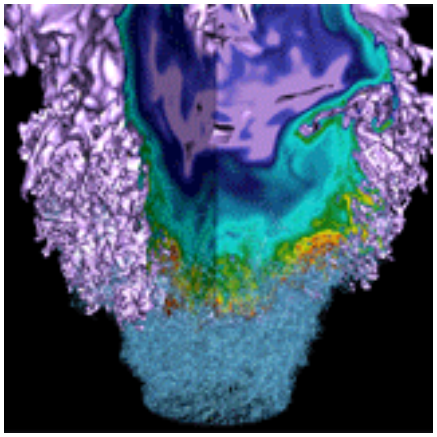


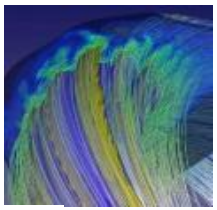


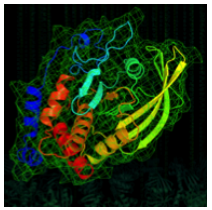
Lead by Richard Gerber and Harvey Wasserman

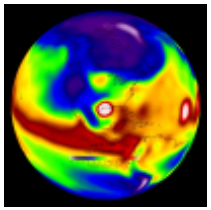
http://www.nersc.gov/science/requirements-workshops/final-reports/

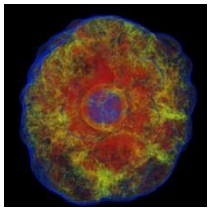
BerkeleyGW Case Study
Jack Deslippe

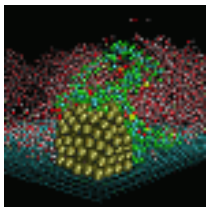












NERSC 40 YEARS at the FOREFRONT

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB
Lawrence Berkeley National Laboratory

Case Study: BerkeleyGW

BerkeleyGW

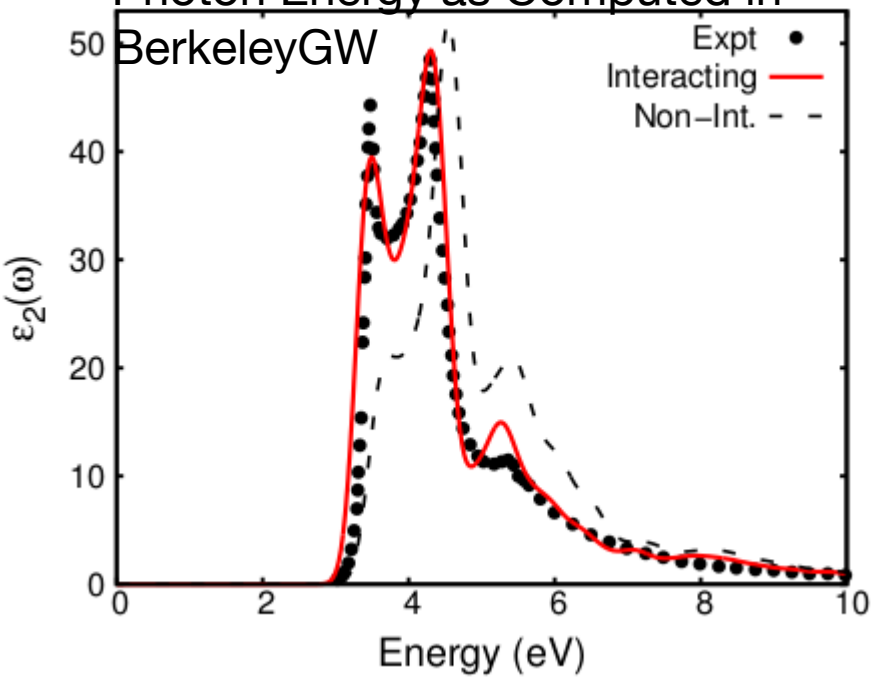
Description:

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

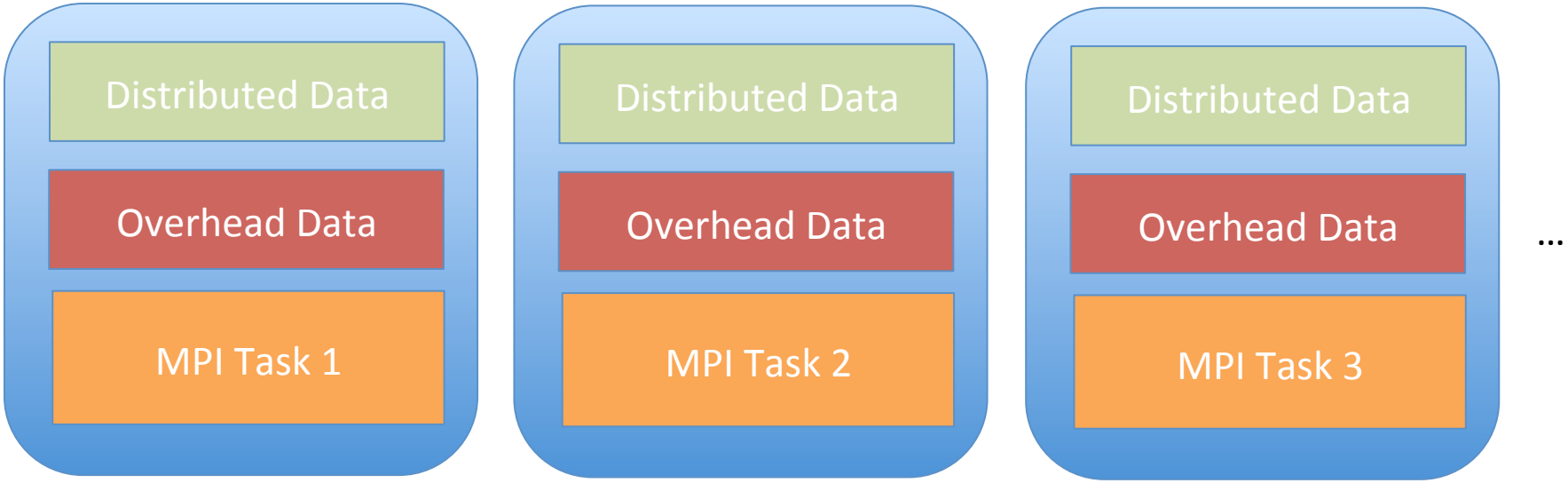


Failure of the MPI-Only Programming Model in BerkeleyGW

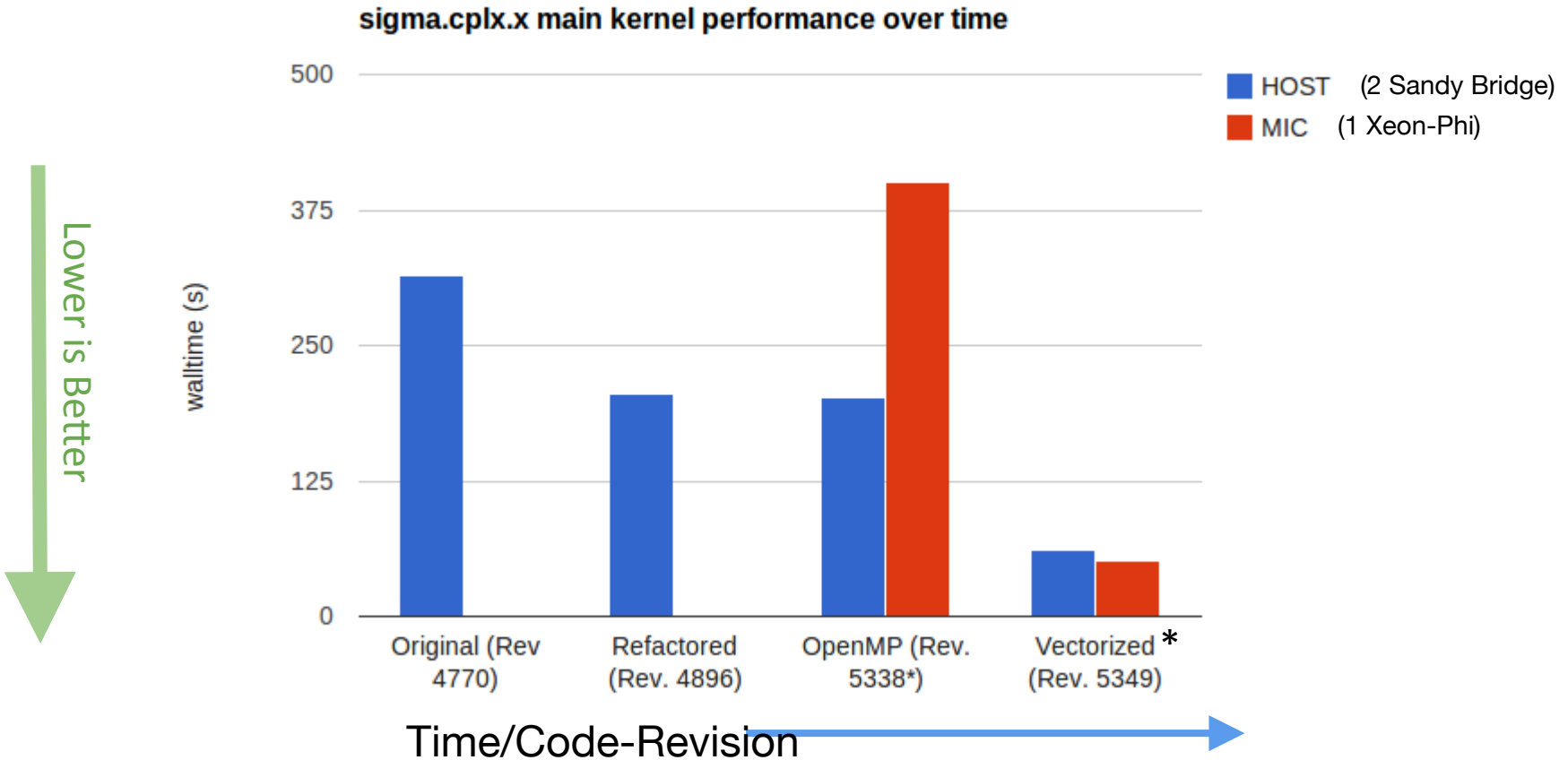
★ Big systems require more memory. Cost scales as N_{atm}^2 to store the data.

★ 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**.



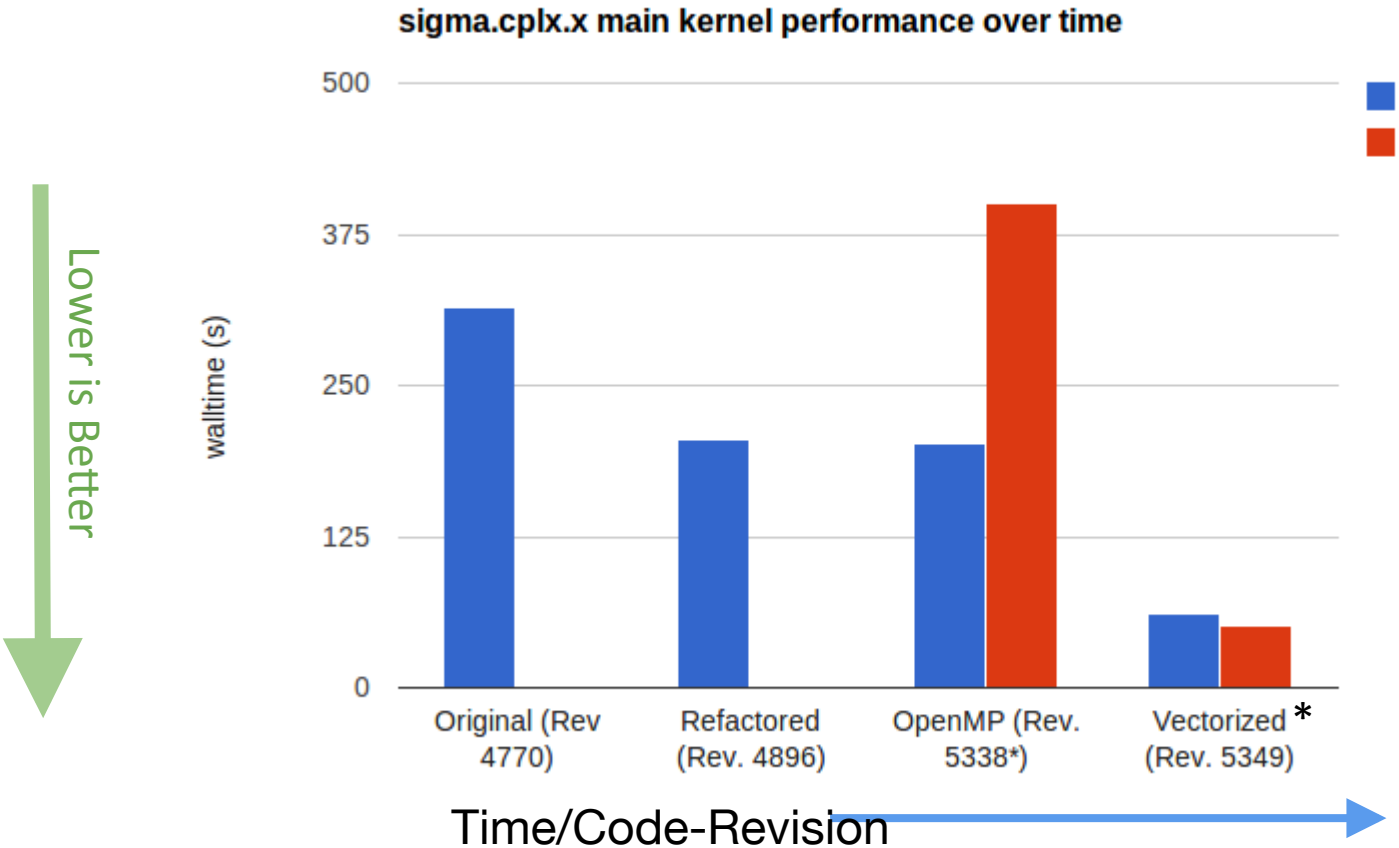
Steps to Optimize BerkeleyGW on Xeon-Phi Testbed



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

Steps to Optimize BerkeleyGW on Xeon-Phi Testbed



After optimization, 4 early Intel Xeon-Phi cards with MPI/OpenMP is ~1.5X faster than 32 cores of Intel Sandy Bridge on test problem.

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
```

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
```

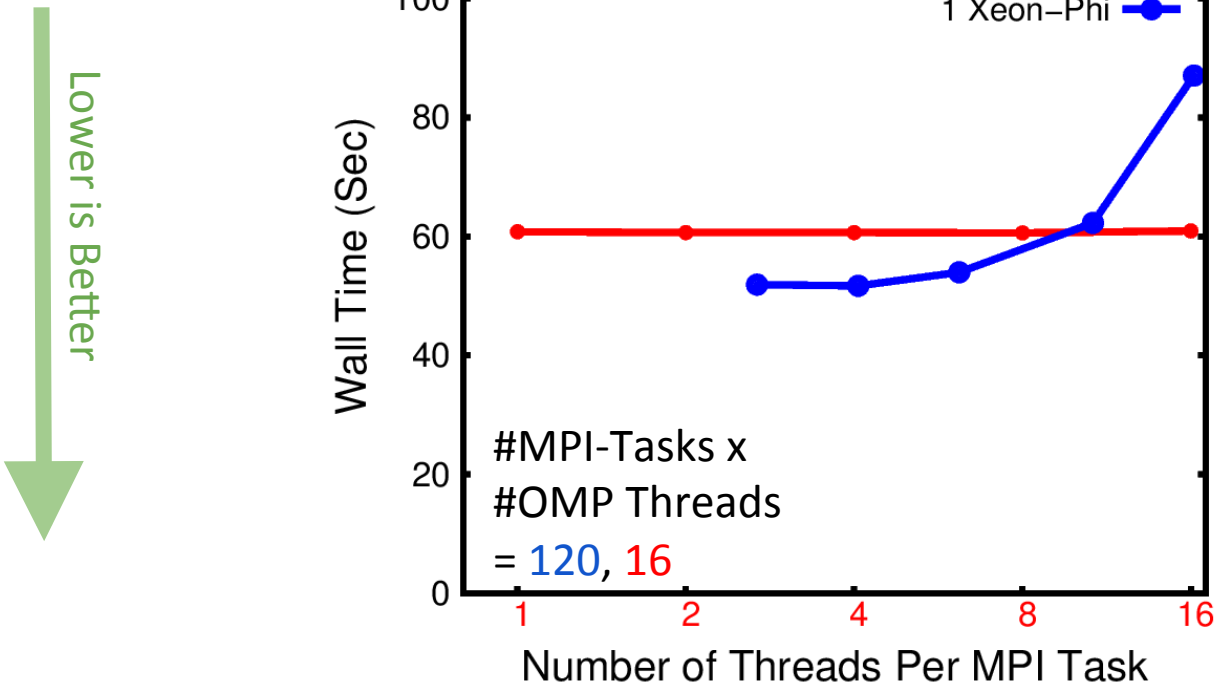
ngpown typically in 100's to 1000s. Good for many

Original inner loop. Too small to vectorize!

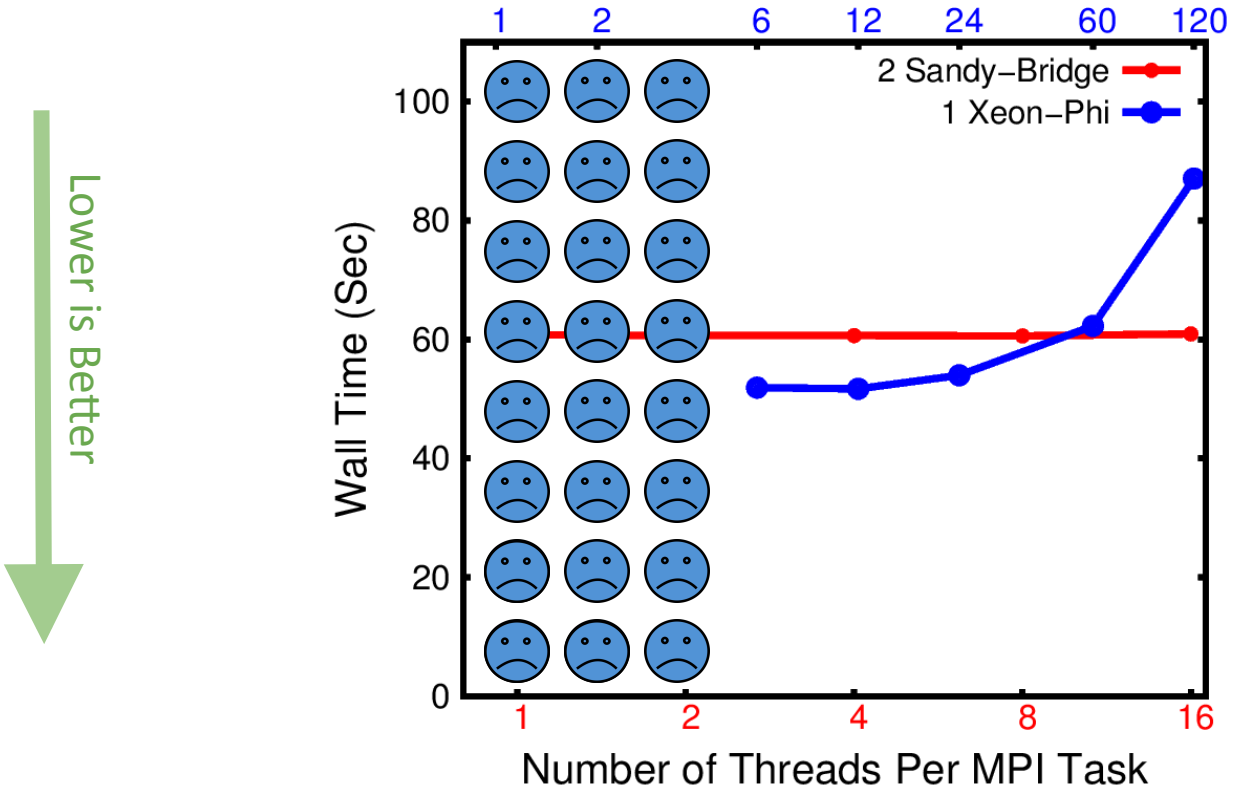
ncouls typically in 1000s - 10,000s. Good for vectorization. Don't have to worry much about memory.

Attempt to save work breaks vectorization and makes code slower.

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



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

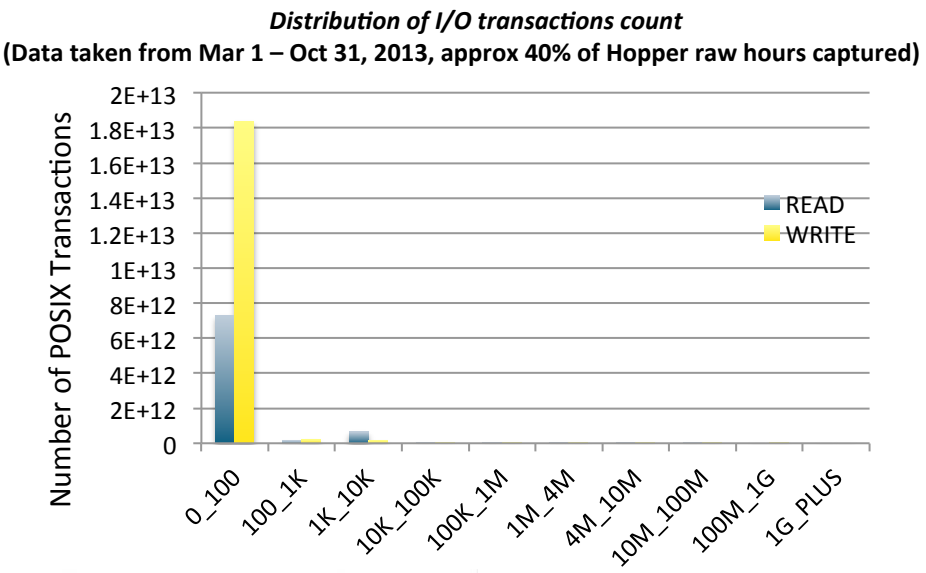


- Example problem cannot fit into memory when using less than 5 OpenMP threads per MPI task.
- **Conclusion: you need OpenMP to perform well on Xeon-Phi in practice**

Burst Buffer cont.

- **NERSC I/O is not well structured**
  - Even after all file system attempts to optimize I/O operations, unaligned, or small I/O patterns result in 50% of all I/Os logged by LMT at NERSC being less than the filesystem block size.
  - Given that Lustre is combining sequential small I/O ops into larger ones, this implies that these are not in the middle of a larger sequential I/O stream, and require head movement in order to be serviced, and would benefit from flash's better small block, and random I/O characteristics vs. disk.

Application(client) I/O request sizes



Lustre(server) I/O sizes