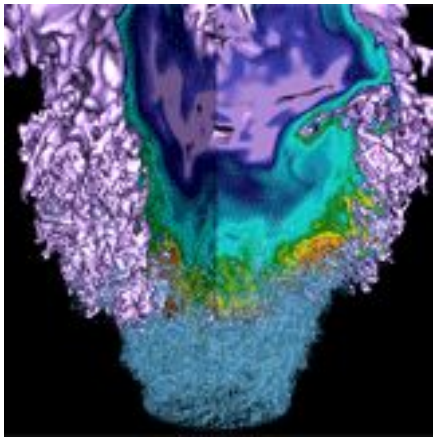
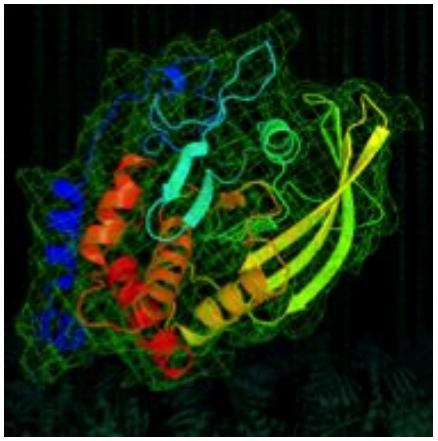
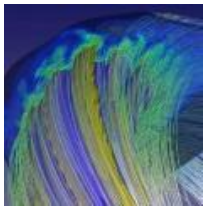
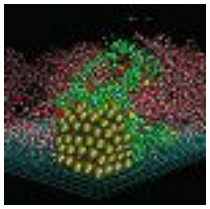
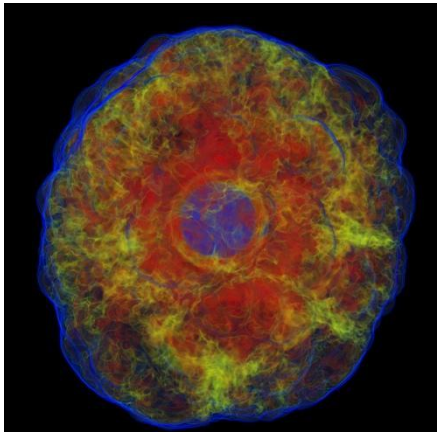
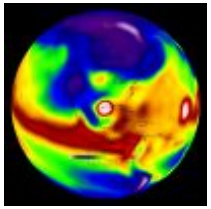
NUG Monthly Telecon


NERSC

November 6, 2015

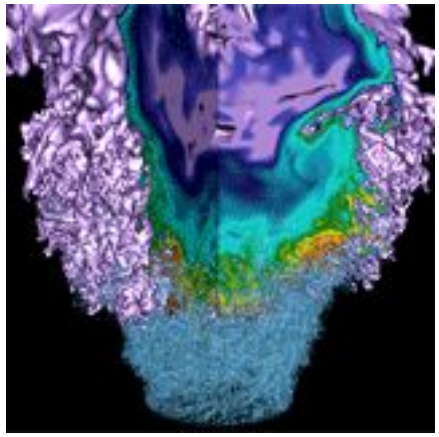
U.S. DEPARTMENT OF ENERGY | Office of Science

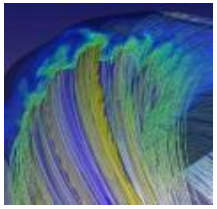
BERKELEY LAB
Lawrence Berkeley National Laboratory

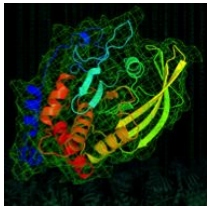
Agenda

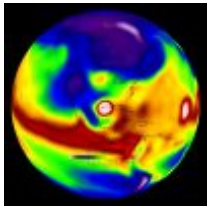
1. Cori Phase 1 Update
2. NERSC Move Outage Schedule and User Impact
3. Edison Update
4. Security Update
5. Shifter Update
6. NESAP Update - Tales from the Dungeon
7. Highlight Story - On-the-fly data post-processing in simulations using “sidecars” in Nyx/BoxLib
8. SLURM Mini-Tutorial

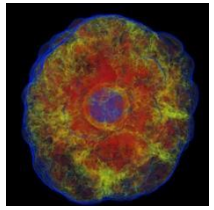
Cori Phase 1 Update

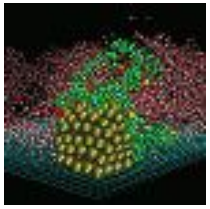












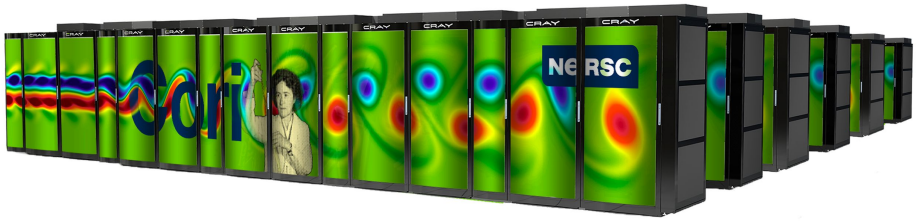
Wahid Bhimji

NERSC Users Group
November 6th 2015

Cori Overview (reminder)

- **Phase 2 (coming mid-2016) - over 9,300 '[Knights Landing](#)' nodes**

Phase 1 (installed now):



- **1630 Compute Nodes**
- Two x 16-core (2.3 GHz) Haswell
  - 128 GB DDR4 2133 Mhz memory/ node(some 512 /768 GB)
  - Cray Aries high-speed "dragonfly" topology interconnect
  - 12+ login nodes for advanced workflows and analytics
  - SLURM batch system
- **Lustre File system (also installed now)**
  - 28 PB capacity, >700 GB/sec peak performance
  - Ultimately will be mounted across other NERSC systems
- **Burst Buffer (NVRAM based)**
  - 750 TB now. Striped / Private / Persistent use-cases

Status

- Final configuration and acceptance testing
- Some user access now in intervals between acceptance tests
- All users should be enabled in the next 2 weeks by mid-Nov
- Cori Phase 1 should be in full production before Edison move (end-Nov) and Hopper retirement (mid-Dec)

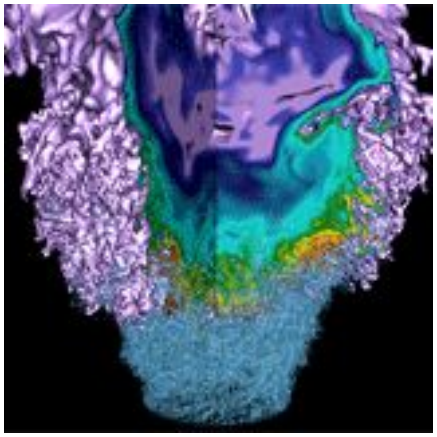
Burst Buffer:

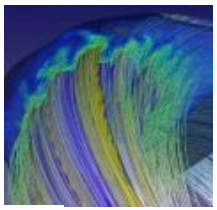
- **Not yet available for users**
- Hardware and software installed. Undergoing configuration and testing. Ready for 'early users' in coming weeks

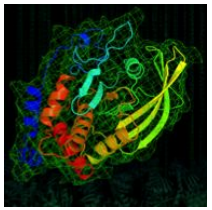
Full configuration details and user guides (including SLURM transition available at:

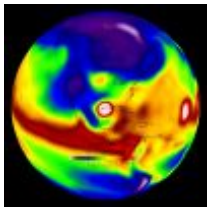
- https://www.nersc.gov/users/computational-systems/cori/

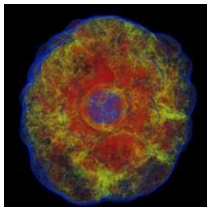
NERSC Move: Outage Schedule and User Impact

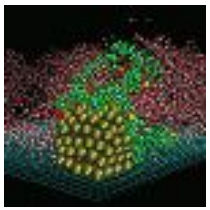












NERSC 40 YEARS at the FOREFRONT

Helen He

NUG Meeting, 11/06/2015

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB
Lawrence Berkeley National Laboratory

Main Timeline

Event	Date
Cori Phase 1 available to all users	By mid Nov 2015 (estimate)
Edison offline to move to CRT	Nov 30, 2015 (estimate to last 6 weeks), back online in Jan 2016 with SLURM batch scheduler
Hopper retires	Dec 15, 2015 (after Cori Phase 1 is stable)
Global project and global homes file systems migration to CRT	Nov - Dec 2015
JGI and PDSF file systems move to CRT	Jan - Feb 2016
Mendel moves to CRT	Feb 2016, affects JGI, PDSF, Materials Project and Babbage.

More details: https://www.nersc.gov/users/move-to-crt/

Global File Systems Move to CRT

- **Migrate various file systems in phases over high speed network connection, expect NO file system downtime (not as previously mentioned).**
- **During the migration, you may observe a reduction of file system performance.**
- **After the migration completes, some compute systems located at OSF will have decreased file system bandwidth.**

File System	Date
/global/projecta	Nov 9 - 11, 2015
/global/project	Nov 24 - Dec 2, 2015
/global/homes, /global/common, /global/syscom	Dec 14 - 16, 2015
JGI retired file systems	Proposed start date Jan 11, 2016

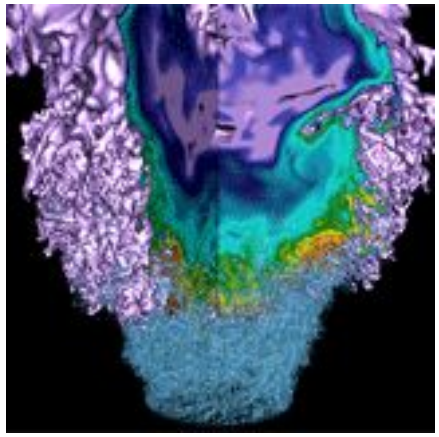
Significantly Reduced System Availabilities

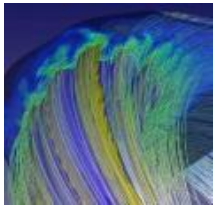
- After Edison is powered off on Nov 30, and Hopper is retired on Dec 15, Cori Phase 1 will be the only MPP system for 4 to 6 weeks.
- Cori phase 1 has only 1,628 total nodes (32 cores per node) as compared to 5,000+ nodes each (24 cores/node) on Hopper and Edison.
  - Expect much longer queue wait time
- Some software may not be immediately available on Cori phase 1.

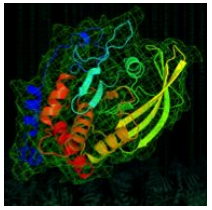
Back up your data on Edison and Hopper

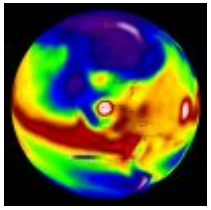
- During the move, Edison's all 3 scratch file systems (/scratch1, /scratch2, and /scratch3) will be reformatted
- Hopper's scratch file systems (/scratch, /scratch2) will retire when Hopper retires.
- Please save your important files to HPSS, your /project directory, or elsewhere. Starting from NOW.
- Use "htar" utility to concatenate small files instead of saving them individually with "hsi" to HPSS.

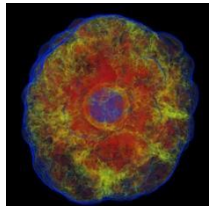
Edison Updates

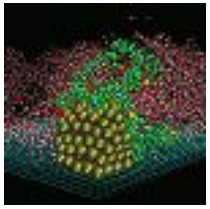












Zhengji Zhao
November 6, 2015

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB
Lawrence Berkeley National Laboratory

Edison Move Plan

- **Edison will be powered off on 11/30/2015 7:00 PST.**
  - Edison queues will be turned off at 00:01 PST (midnight) on November 30, 2015.
  - Login nodes will be still open to users until 7:00 PST on 11/30
- **All files on the /scratch1, /scratch2, and /scratch3 on Edison will be deleted.**
  - Please back up your files to HPSS or other permanent file systems.
  - Please use the htar command (NOT hsi) to backup files to HPSS for optimal file storage/retrieve.
- **Edison will be down for up to 6 weeks.**

Edison will migrate to Slurm in January 2015

- **Edison batch system will be Slurm when it is back to production in mid January.**
  - https://www.nersc.gov/users/computational-systems/edison/running-jobs-for-slurm/example-batch-scripts/
  - If you need help with your job scripts, please let us know at consult@nersc.gov

CDT 15.09 is available as non-default version

atp/1.8.3
chapel/1.12.0
cray-ccdb/1.0.7
cray-ga/5.3.0.3
cray-hdf5/1.8.14
cray-hdf5-parallel/1.8.14
cray-lgdb/2.4.5
cray-libsci/13.2.0
cray-mpich/7.2.5
cray-netcdf/4.3.3.1
cray-netcdf-hdf5parallel/4.3.3.1
cray-parallel-netcdf/1.6.1
craype/2.4.2
cray-petsc/3.6.1.0
cray-petsc-64/3.6.1.0
cray-petsc-complex/3.6.1.0
cray-petsc-complex-64/3.6.1.0
craypkg-gen/1.3.2
cray-shmem/7.2.5
cray-tpsl/1.5.2
cray-tpsl-64/1.5.2
cray-trilinos/11.12.1.5
fftw/2.1.5.9
fftw/3.3.4.5
iobuf/2.0.6
papi/5.4.1.2
perftools/6.3.0
perftools-lite/6.3.0
stat/2.2.0.1

intel/2016.0.109
cce/8.4.0
gcc/5.1.0

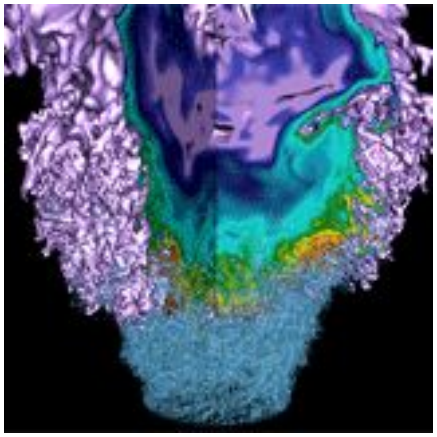
We will set this CDT version to default during next Edison maintenance. The CDT 15.09 release note can be found here,

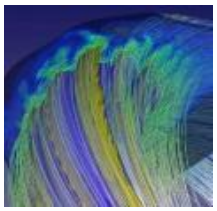
http://docs.cray.com/books/S-9408-1509//S-9408-1509.pdf

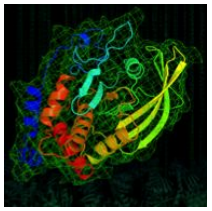
Intel VTune 2016 is available on Edison

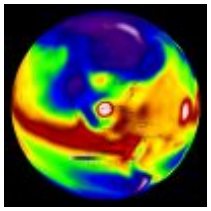
- **Intel Parallel Studio 2016 release is available on Edison**
  - vtune/2016(default)
  - advisor/2016(default)
  - inspector/2016(default)
  - impi/5.1.1.109(default)

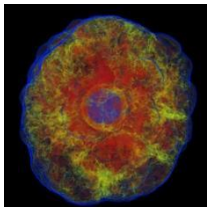
Security Update With Brent

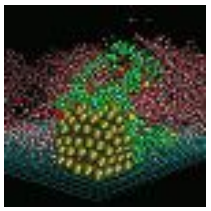












NERSC

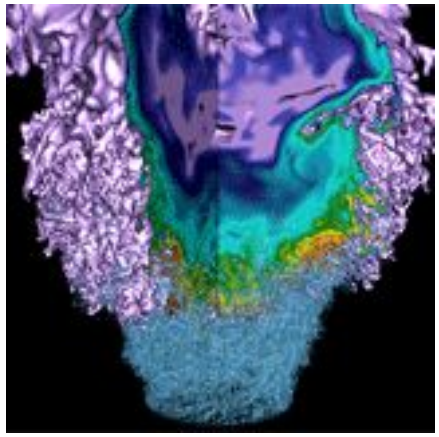
U.S. DEPARTMENT OF ENERGY | Office of Science

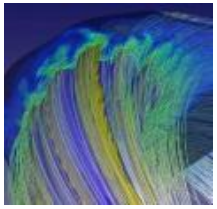
BERKELEY LAB
Lawrence Berkeley National Laboratory

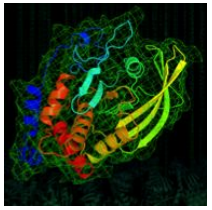
NERSC Cybersecurity Update

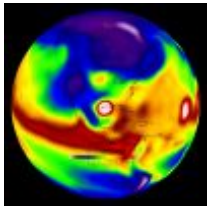
- **Philosophy**
  - Science first, security is a net enabler of science
  - Have avoided OTP tokens so far
  - Allow public/private key authentication
- **Biggest risk**
  - Stolen login credentials
    - Mostly from other scientific clusters
- **Changes for next Allocation Year**
  - shost authentication within supercomputers
  - Public keys moved from filesystem to NIM/ldap
- **NERSC security tutorial web page**
  - http://www.nersc.gov/users/training/online-tutorials/cybersecurity-tutorial/

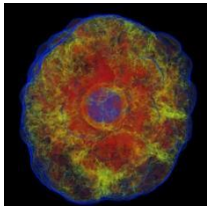
Shifter

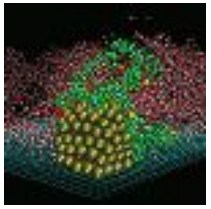












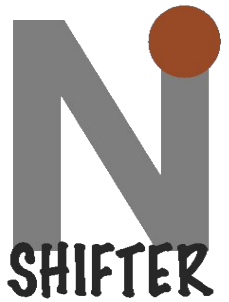
NERSC

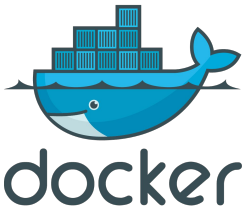
U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB
Lawrence Berkeley National Laboratory

User Defined Images/Containers in HPC

- **Data Intensive computing often require complex software stacks**
- **Efficiently supporting "big software" in HPC environments offers many challenges**
- **Shifter**
  - NERSC R&D effort, in collaboration with Cray, to support User-defined, user-provided Application images
  - "Docker-like" functionality on the Cray and HPC Linux clusters
  - Efficient job-start & Native application performance





ChOS

Why Users will like Shifter

- Develop an application on your desktop and run it on Cori and Edison
- Enables you to solve your dependency problems yourself
- Run the (Linux) OS of your choice and the software versions you need
- Improves application performance in many cases
- Improve reproducibility
- Improve sharing (through sites like Dockerhub)

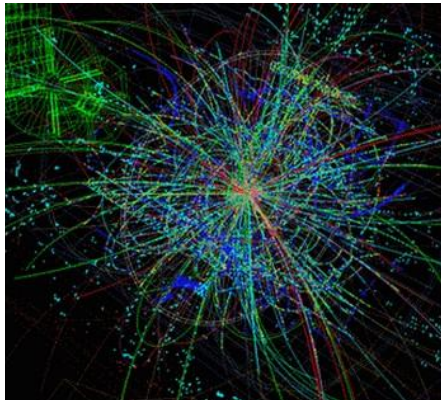
Use Cases

- **Large high energy physics collaborations (e.g., ATLAS and STAR) requiring validated software stacks**
  - Some collaborations will not accept results from non-validated stacks
  - Simultaneously satisfying compatibility constraints for multiple projects is difficult
  - Solution: create images with certified stacks
- **Bioinformatics and cosmology applications with many third-party dependencies**
  - Installing and maintaining these dependencies is difficult
  - Solution: create images with dependencies
- **Seamless transition from desktop to supercomputer**
  - Users desire consistent environments
  - Solution: create an image and transfer it among machines

Where are we now?

- **An early version of Shifter is deployed on Edison. Early users are already reporting successes!**
- **Shifter is fully integrated with batch system, users can load a container on many nodes at job-start time. Full access to parallel FS and High Speed Interconnect (MPI)**
- **Shifter and NERSC were recently featured in HPC Wire and other media. Several other HPC sites have expressed interest.**
- **Our early users:**

Light Sources
Structural Biology
(early production)

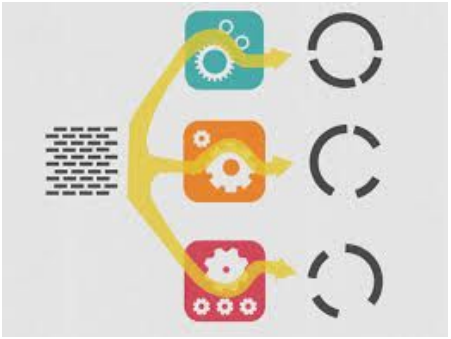


LHC – Nuclear Physics
(testing)

Large Synoptic Survey Telescope

LSST DESC Dark Energy Science Collaboration

Cosmology
(testing)

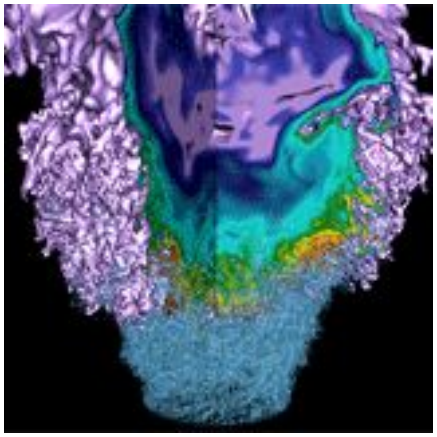


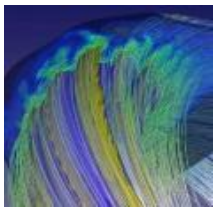
nucleotid.es
Genome Assembly
(proof of concept)

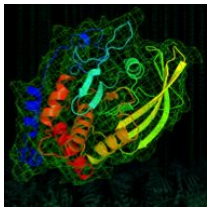
Where are we going?

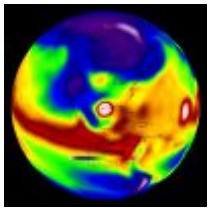
- **Cori phase 1 will debut with a new fully *open-source* version of Shifter**
  - Goal: Publicly release the code (via bitbucket) prior to SC15
  - Users can run multiple containers in a single job (i. e., dynamically select and run Shifter containers)
  - Tighter integration and simple user interface
- **NERSC is working with Cray who has indicated interest in making a supported product out of Shifter**
- **Ways to enable users to create Cray-optimized images**

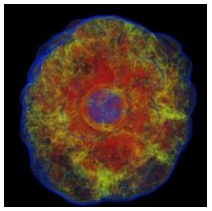
NESAP Update - Tales from the Dungeon

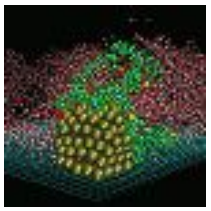






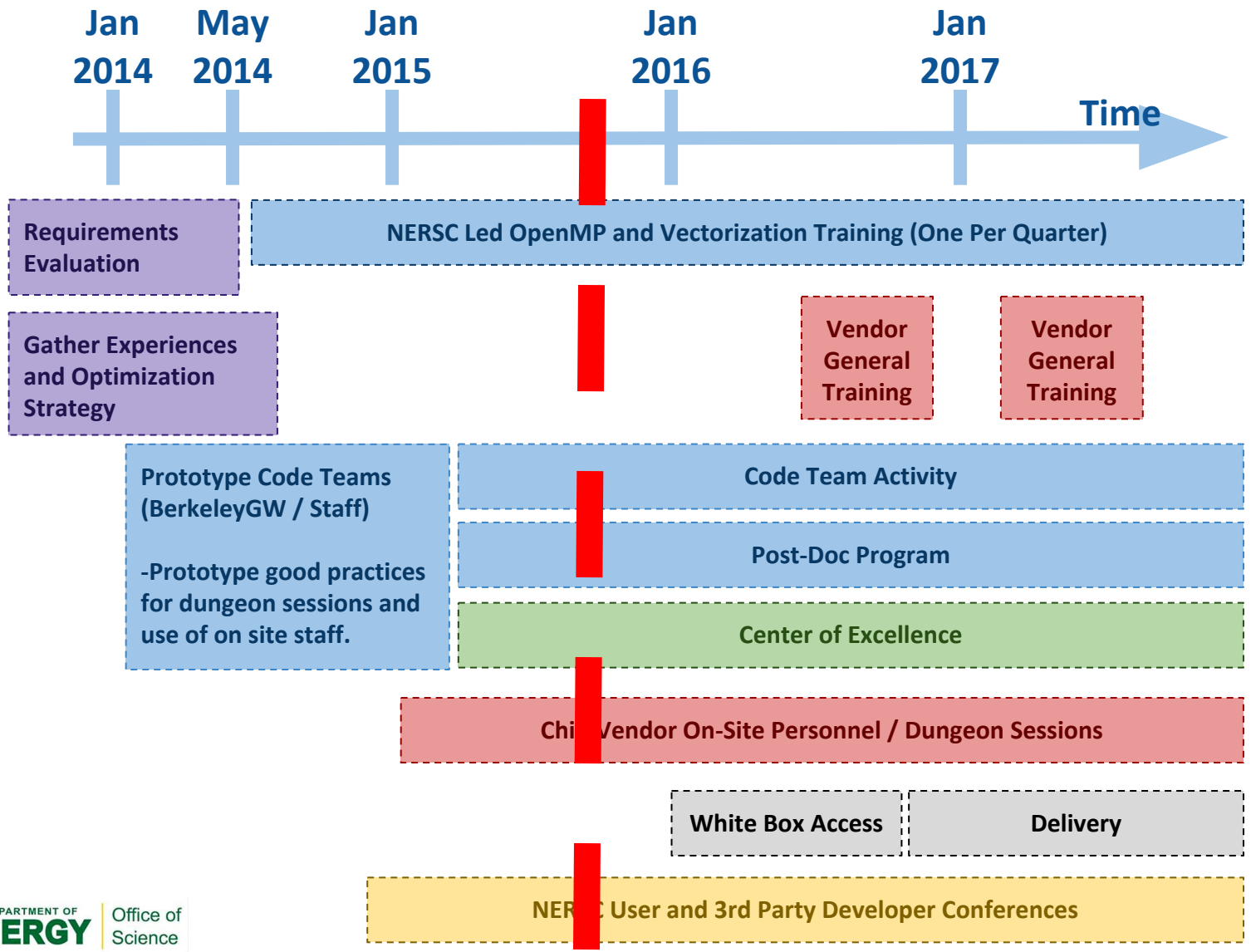






NERSC

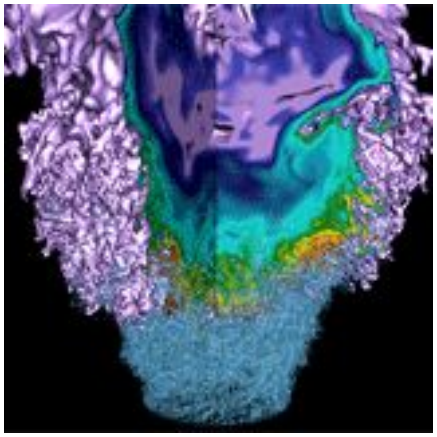
NESAP Timeline

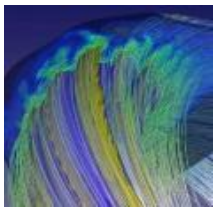


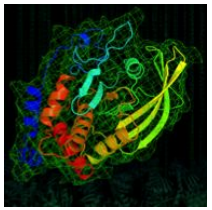
NESAP Postdoc Program Update

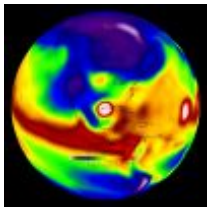
- **Currently: 3 postdocs**
  - Brian Friesen: BoxLib (Almgren)
  - Andrey Ovsyannikov: Chombo Crunch (Trebotich)
  - Taylor Barnes (Hopper Fellow): Quantum Espresso (Kent)
- **Three new postdocs starting in new year**
  - Tareq Malas: EMGeo (Newman)
  - Tuomas Koskela: XGC1 (Chang)
  - Mathieu Lobet: WARP/Synergia (Vay)
- **Two open positions – please encourage qualified applicants**
  - NERSC Exascale Science Applications Postdoctoral Fellowship (NESAP) https://lbl.taleo.net/careersection/jobdetail.ftl?job=81356&lang=en

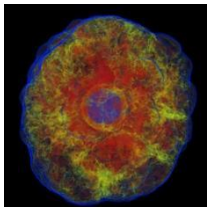
Preparing for a Dungeon Session

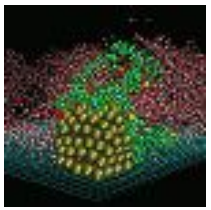












NERSC

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB
Lawrence Berkeley National Laboratory

The Ant Farm!

OpenMP scales only to 4 Threads

large cache miss rate

Communication dominates beyond 100 nodes

Code shows no improvements when turning on vectorization

50% Walltime is IO

Compute intensive doesn't vectorize

MPI/OpenMP Scaling Issue

Memory bandwidth bound kernel

IO bottlenecks

Can you use a library?

Increase Memory Locality

Use Edison to Test/Add OpenMP Improve Scalability. Help from NERSC/Cray COE Available.

Create micro-kernels or examples to examine thread level performance, vectorization, cache use, locality.

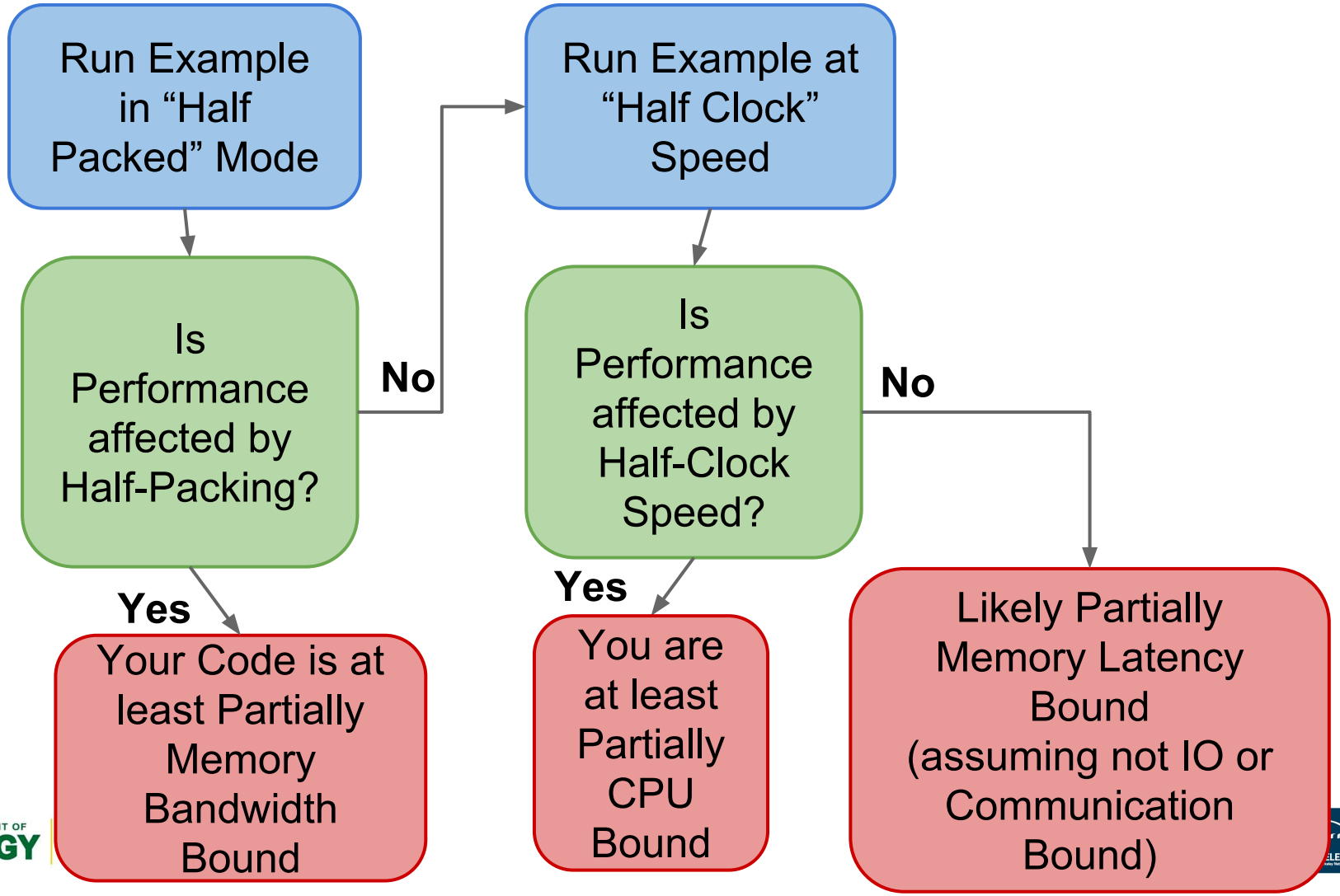
Utilize High-Level IO-Libraries. Consult with NERSC about use of Burst Buffer.

Utilize performant / portable libraries

The Dungeon: Simulate kernels on KNL. Plan use of on package memory, vector instructions.

Optimization Flow Chart

https://www.nersc.gov/users/computational-systems/cori/application-porting-and-performance/getting-started-and-optimization-strategy/



Are you memory or compute bound? Or both?

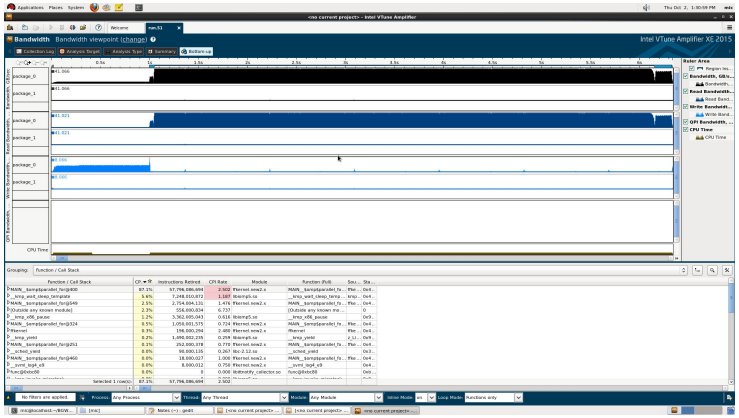
Run Example in "Half Packed" Mode

If you run on only half of the cores on a node, each core you do run has access to more bandwidth

`aprun -n 24 -N 12 - S 6 ...` VS `aprun -n 24 -N 24 -S 12 ...`

If your performance changes, you are at least partially memory bandwidth bound

Or! Measure your actual bandwidth usage with VTune!

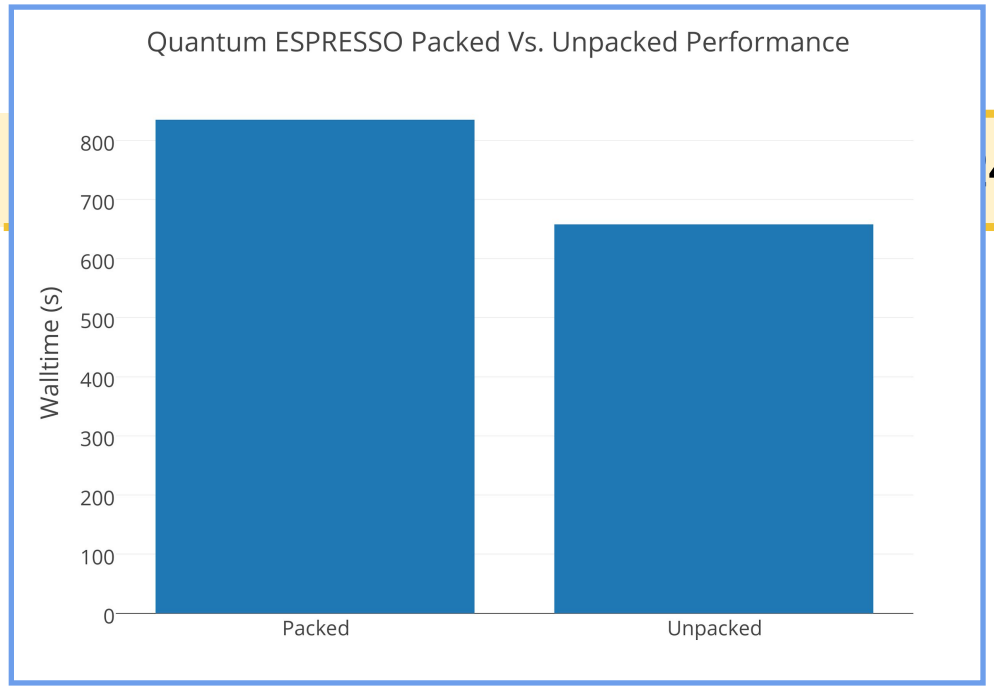


Are you memory or compute bound? Or both?

Run Example in “Half Packed” Mode

If you run on only half of the cores on a node, each core you do run has access to more bandwidth

aprun -n 24 -N 4 -



Are you memory or compute bound? Or both?

Run Example at "Half Clock" Speed

Reducing the CPU speed slows down computation, but doesn't reduce memory bandwidth available.

```
aprun --p-state=2400000 ...
```

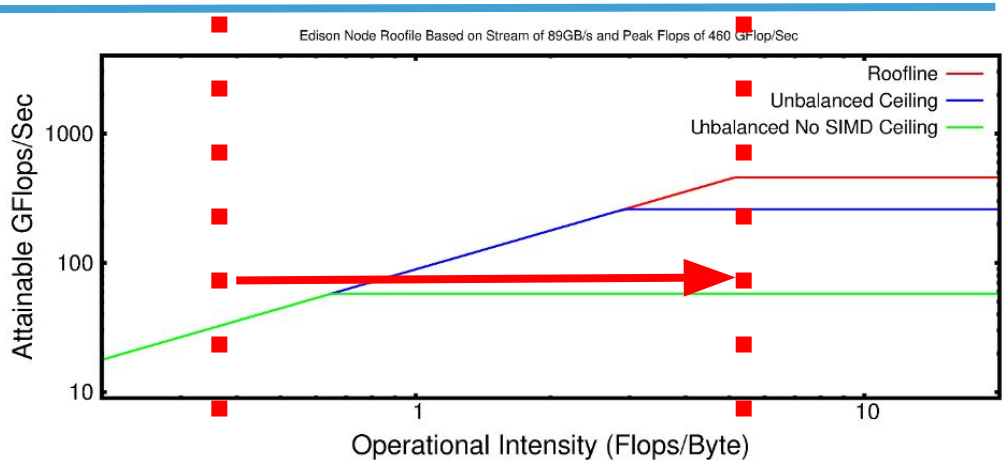
VS

```
aprun --p-state=1900000 ...
```

If your performance changes, you are at least partially compute bound

What to do?

Memory Bound?



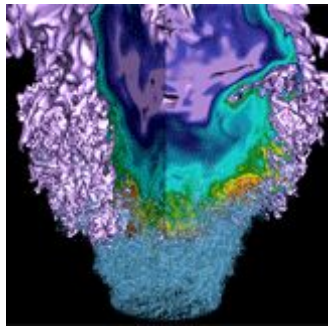
1. Try to improve memory loc
   cache reuse

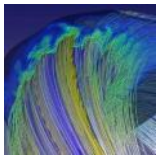
2. Identify the key arrays leading to high memory bandwidth usage and make sure they are/will-be allocated in HBM on Cori.

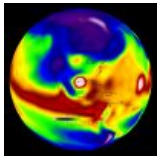
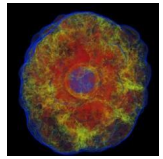
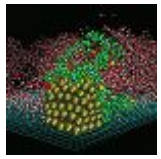
Compute Bound?

Make sure you have low CPI and high VPU utilization. Good thread scaling, well load balanced.

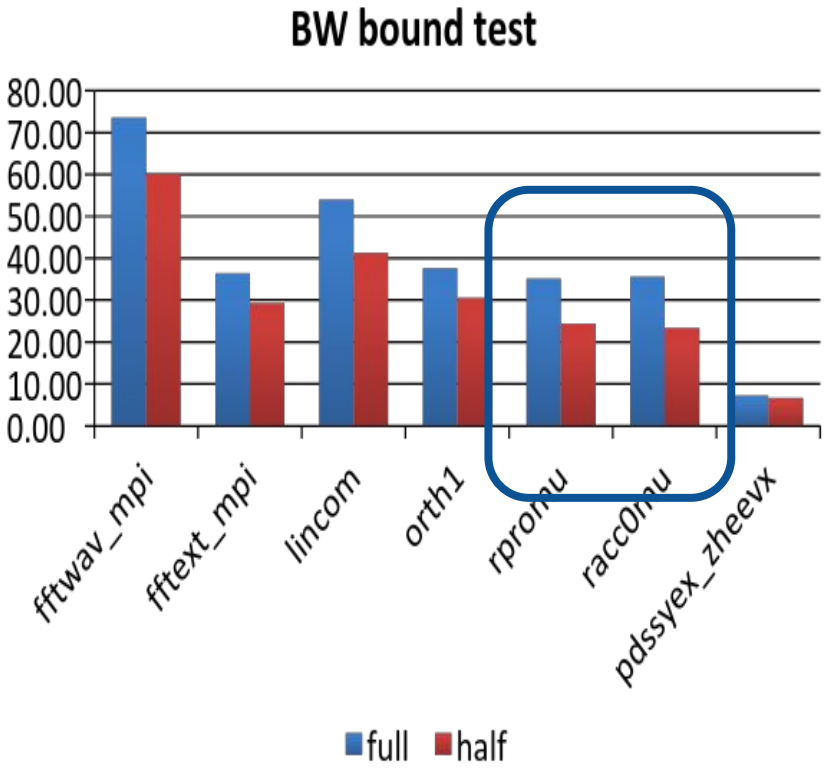
Code Example 1
VASP

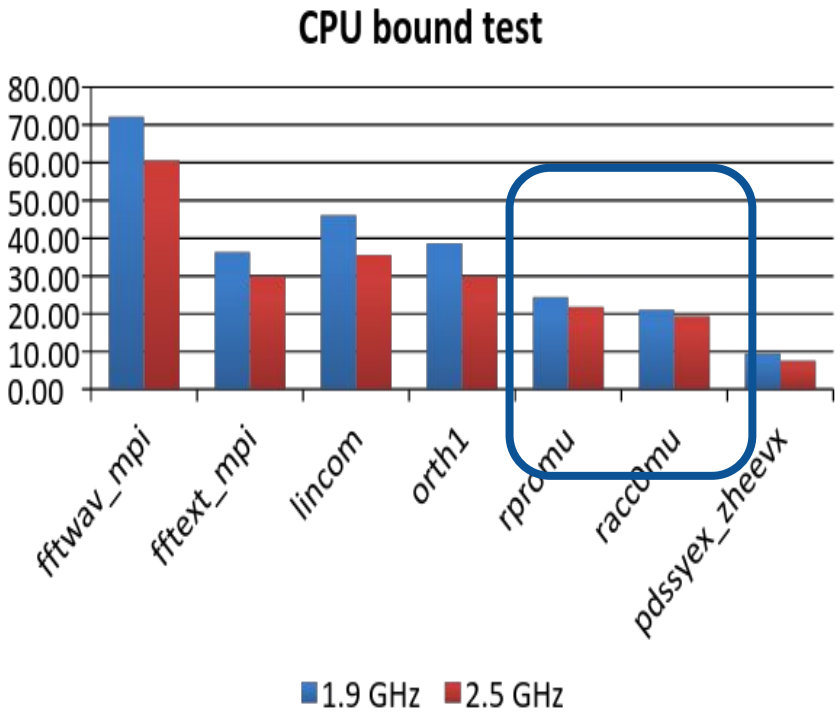




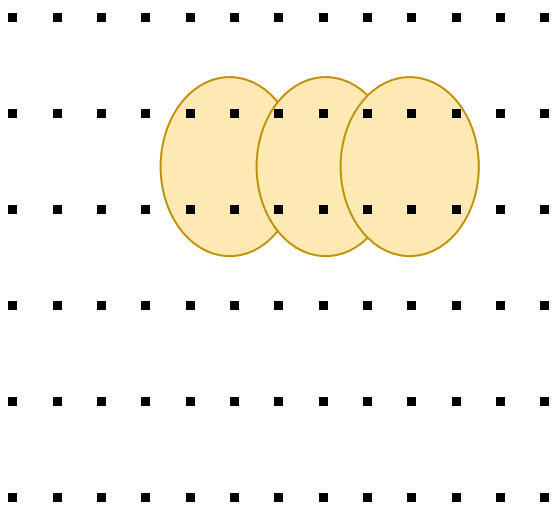
Identify Memory Bandwidth Bound Regions of Code





Memory Bandwidth Bound Regions of Code

```
do (all Ions/projectors, k)
    dgemm to create W^k_n(i)
    do n = 1, num_bands_per_block
        do i = 1, num_grid_points_in_sphere
            Result(index(i)) = ψ_n(Index(i)) * W^k_n(i)
        enddo
    enddo
enddo
```



- ψ_n - 4MB per n, 64 MB for whole block (too big for L3). It is streamed in.
- $W^k_n(i)$ - 1 MB

Memory Bandwidth Bound Regions of Code

1. Make sure W and ψ_n a are in HBM

```
do (all Ions/projectors, k)
    dgemm to create W^k_n(i)
    do n = 1, num_bands_per_block
        do i = 1, num_grid_points_in_sphere
            Result(index(i)) = ψ_n(Index(i)) +/* W^k_n(i)
        enddo
    enddo
enddo
```

Memory Bandwidth Bound Regions of Code

- **Identify the candidate (key arrays) for HBM**
  - Using NUMA affinity to simulate HBM on a dual socket system
  - Use FASTMEM directives and link with

On Edison (NERSC Cray XC30):

```
real, allocatable :: a(:,:), b(:,:), c(:)
!DIR$ ATTRIBUTE FASTMEM :: a, b, c
% module load memkind jemalloc
% ftn -dynamic -g -O3 -openmp mycode.f90
% export MEMKIND_HBW_NODES=0
% aprun -n 1 -cc numa_node numactl --membind=1 --cpunodebind=0 ./myexecutable
```

On Haswell:

Link with '-ljemalloc -lmemkind -lpthread –lnuma "

```
% numactl --membind=1 --cpunodebind=0 . /myexecutable
```

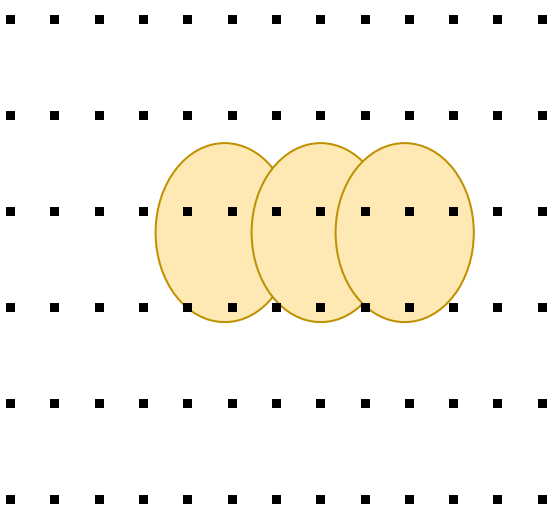
Memory Bandwidth Bound Regions of Code

```
do (all Ions/projectors, k)
    dgemm to create W^k_n(i)
    do n = 1, num_bands_per_block
        do i = 1, num_grid_points_in_sphere
            Result(index(i)) = ψ_n(Index(i)) +/* W^k_n(i)
        enddo
    enddo
enddo
```

1. Make sure W and ψ_n a are in HBM
2. Reorder loops to allow ψ_n to be reused.

Memory Bandwidth Bound Regions of Code

```
do (all Ions/projectors, k)
    dgemm to create W^k_n(i)
    do n = 1, num_bands_per_block
        do i = 1, num_grid_points_in_sphere
            Result(index(i)) = ψ_n(Index(i)) +/* W^k_n(i)
        enddo
    enddo
enddo
```



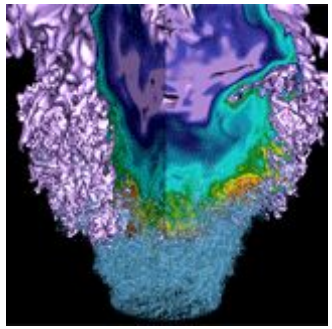
1. Make sure W and ψ_n a are in HBM
2. Reorder loops to allow ψ_n to be reused.
3. Sort loop over ions ψ_n such that we use a fraction that can broken down and stored in real-space tiles.

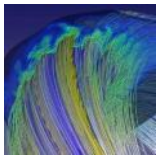
Memory Bandwidth Bound Regions of Code

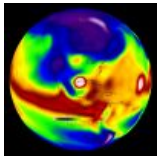
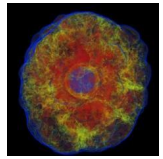
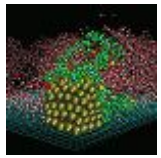
```
do (all Ions/projectors, k)
    dgemm to create W^k_n(i)
    do n = 1, num_bands_per_block
        do i = 1, num_grid_points_in_sphere
            Result(index(i)) = ψ_n(Index(i)) +/* W^k_n(i)
        enddo
    enddo
enddo
```

1. Make sure W and ψ_n a are in HBM
2. Reorder loops to allow ψ_n to be reused.
3. Sort loop over ions ψ_n such that we use a fraction of that can broken down and stored in real-space tiles.
4. Vectorize inner loop by taking advantage of stride 1 block in Index array. Pad sphere into cube.

Code Example 2
EMGEO





Stencil SPMV

- **Observation: Have to stream through the matrix data in SpMV (reuse not an option)**
  - Can we still reduce the bytes / flop somehow?
- **Idea: This is a stencil computation... If we compute the sparsity pattern on the fly, then no need to stream through the indirect indexing**
  - It turns out this should be easy away from portions of the matrix where B.C.s are enforced
  - Still use indirect indexing array in these regions

Stencil SPMV

```
subroutine ell_spmv(mat, ind, x, z,
m, n, ndiag)
  implicit none
  ! --
  integer :: m, n, ndiag
  integer, dimension(ndiag, m) ::
ind
  complex*16, dimension(n) :: x
  complex*16, dimension(m) :: z
  complex*16, dimension(ndiag, m)
:: mat
  ! --
  integer :: i, j
  complex*16 :: ztmp
!$omp parallel do private(ztmp)
  do i = 1, m
    ztmp =  (0.0d0, 0.0d0)
    do j = 1, ndiag
      ztmp = ztmp + mat(j,i) * x
(ind(j,i))
    end do
```

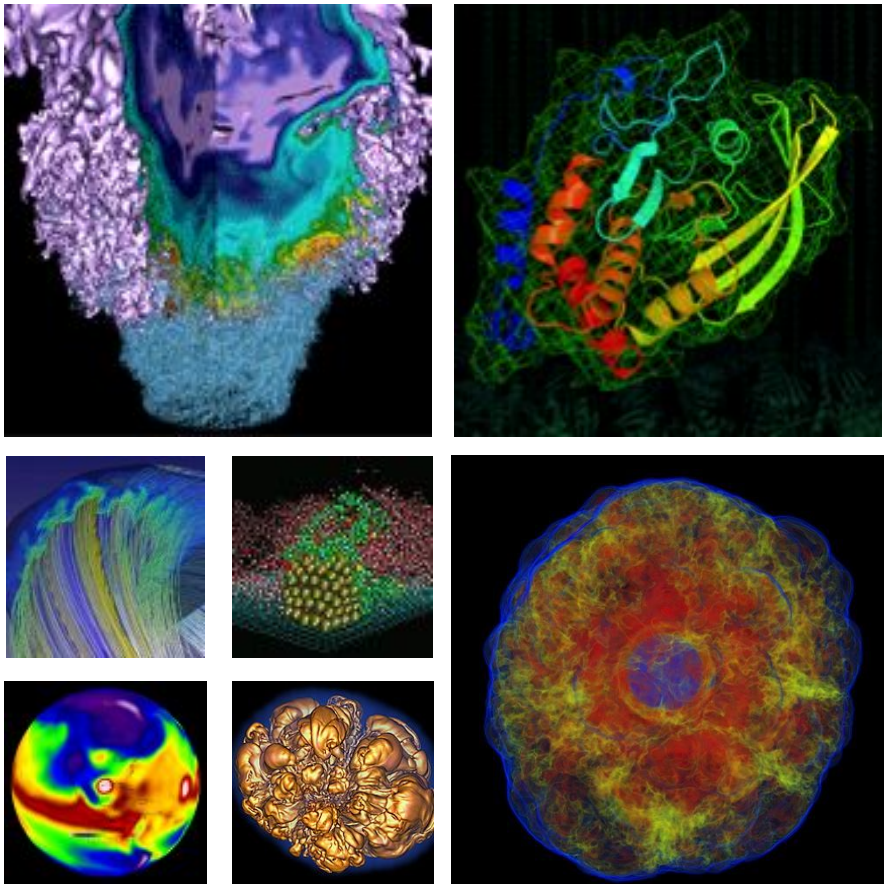
Vector loads when vectorized in `i`

```
!$omp parallel do private(ztmp)
  do i = 1, 2 * nx * ny
    ztmp =  (0.0d0, 0.0d0)
    do j = 1, ndiag
      ztmp = ztmp + mat(i,j) * x
(ind(i,j))
    end do
    z(i) = ztmp
  end do
!$omp parallel do private(ztmp)
  do i = 2 * nx * ny + 1, m -  nx *
ny
    ztmp =  (0.0d0, 0.0d0)
    ! stride 1
    ztmp = ztmp + mat(i,1 ) * x(i -
2)
    ztmp = ztmp + mat(i,2 ) * x(i -
1)
    ztmp = ztmp + mat(i,3 ) * x(i)
    ztmp = ztmp + mat(i,4 ) * x(i +
1)
    ! stride nx
    ztmp = ztmp + mat(i,5 ) * x(i -
2 * nx)
    ztmp = ztmp + mat(i,6 ) * x(i -
nx)
    ztmp = ztmp + mat(i,7 ) * x(i)
    ztmp = ztmp + mat(i,8 ) * x(i +
nx)
    ! stride nx * ny
    ztmp = ztmp + mat(i,9 ) * x(i -
2 * nx * ny)
    ztmp = ztmp + mat(i,10) * x(i -
nx * ny)
    ztmp = ztmp + mat(i,11) * x(i)
    ztmp = ztmp + mat(i,12) * x(i +
nx * ny)
```

Stencil SPMV

- **Speedup ~ 20-25% over original SpMV on HSW EP**
  - Additional gains (few %) when matrix data is transposed (vec)
  - Benefit comes from saved bandwidth due to index array and improved prefetching success
- **Process**
  - Comparing multiple versions of same routine to test on Haswell and Emulator
  - Using vtune to count L3 Misses from demand load vs prefetch

On-the-fly data post-processing in simulations using “sidecars” in Nyx/BoxLib



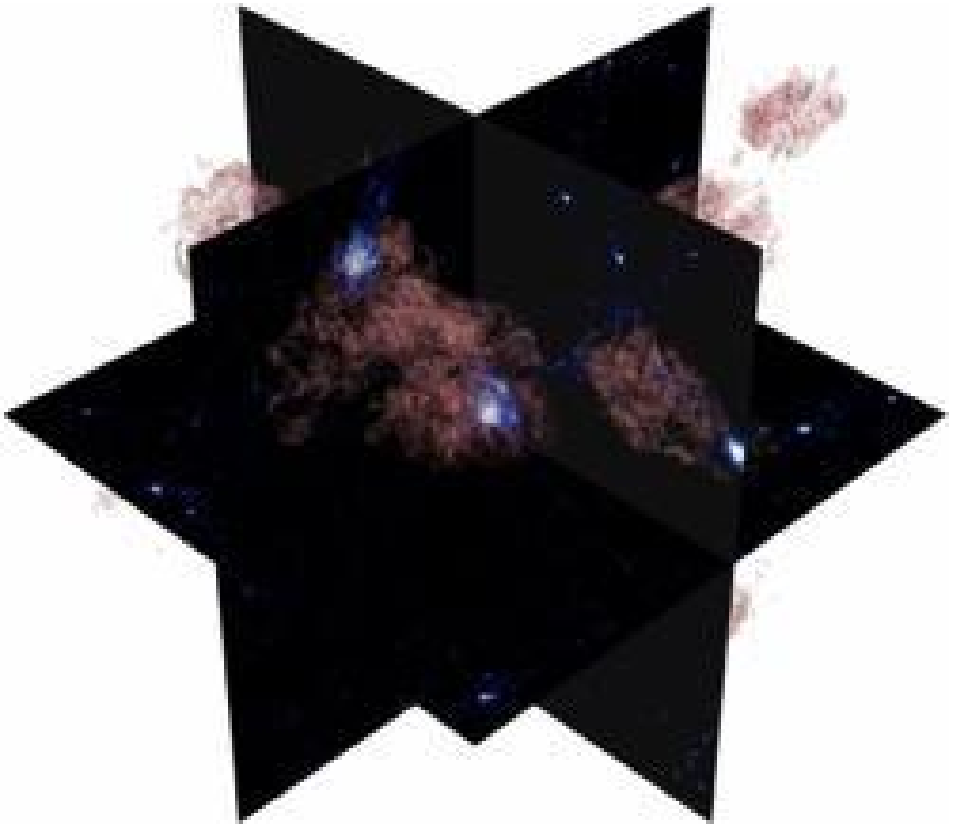
NERSC 40 YEARS at the FOREFRONT 1974-2014

Brian Friesen

2015 Nov 6

Problem: we're running out of disk space

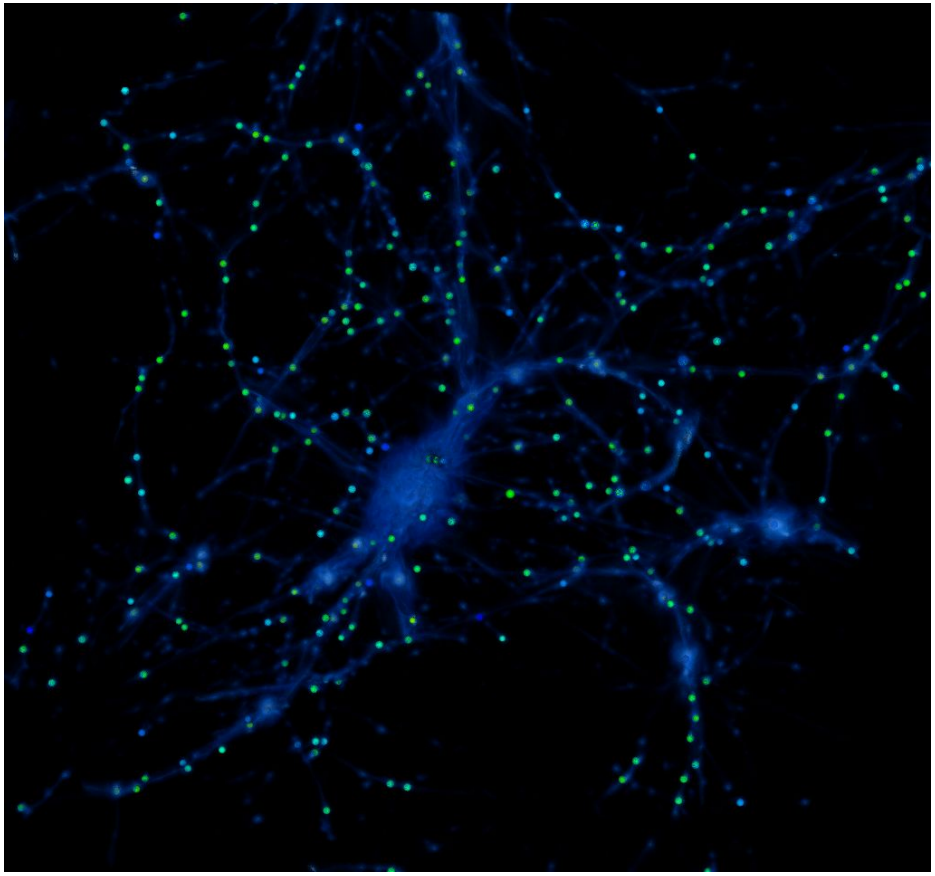
- **The problem**:
  - Disk *capacity* is a bigger problem for us than *bandwidth*
  - Nyx cosmology simulation on a 4096^3 grid generates 4.4 TB plot files per time step
    - takes $O(10^3)$ time steps to evolve from early universe (z=150) to the present (z=0)
  - Recent "Q Continuum" simulation with HACC generated 101 plot files at 20 TB each (2 PB total)



Almgren, Bell,+(2013)

Opportunity: be smart about data selection

- In cosmological simulations, there are only a few observables most people care about, e.g.,
  - size and distribution of dark matter “halos”
  - radiative properties of Lyman-alpha “forest”
- Historically, halo-finding has been relegated to a post-processing step
  - save the raw plot files, then apply halo finder algorithm later to generate a much reduced data set
  - In Nyx, reduction of plot file to list of halos is $O(10^3)$

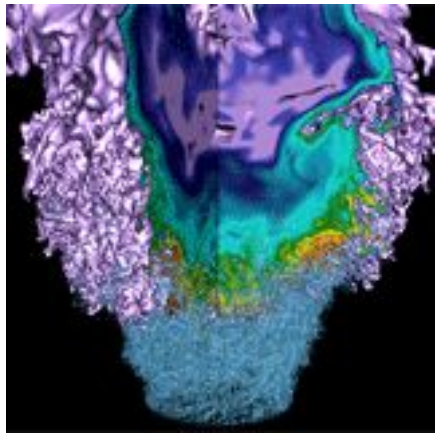


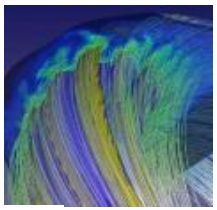
In situ vs. in-transit

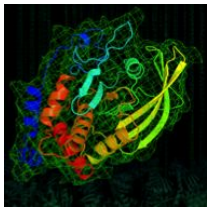
- Dmitriy Morozov and Gunther Weber wrote halo finding tool for Nyx ("Reeber")
  - Calculates level sets of a scalar field (density) to identify local extrema (halos)
  - Originally operated on Nyx plot files; Gunther ported it to run *in situ*
    - All simulation processes stop and find halos every *N* time steps
  - We then ported it run "in-transit" on the BoxLib "sidecars"
    - Dedicated partition of processes which receive grid data and perform halo find while remaining partition continues on with simulation

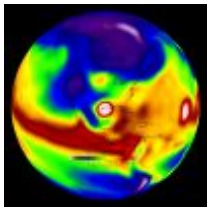
- Whether *in situ* or in-transit is faster depends on many things:
  - Scaling behavior of analysis algorithm vs. simulation algorithm
  - Frequency of analysis (how often do you have to move data?)
  - Speed/latency of interconnect (or burst buffer!)
  - Is there a sweet spot for the *simulation* partition?
    - Does your problem divide nicely onto a particular # of tasks?
  - Is there a sweet spot for the *total* partition?
    - Do you want your job to fit into a particular queue?

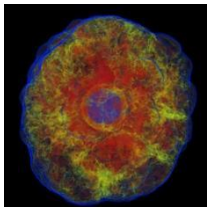
SLURM Resource Manager is Coming to NERSC

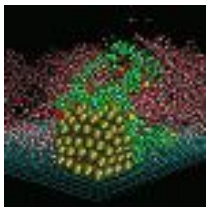












NERSC 40 YEARS at the FOREFRONT

Helen He

NUG Meeting, 11/06/2015

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB
Lawrence Berkeley National Laboratory

What is SLURM

- In simple word, SLURM is a workload manager, or a batch scheduler.
- SLURM stands for Simple Linux Utility for Resource Management.
- SLURM unites the cluster resource management (such as Torque) and job scheduling (such as Moab) into one system. Avoids inter-tool complexity.
- As of June 2015, SLURM is used in 6 of the top 10 computers, including the #1 system, Tianhe-2, with over 3M cores.

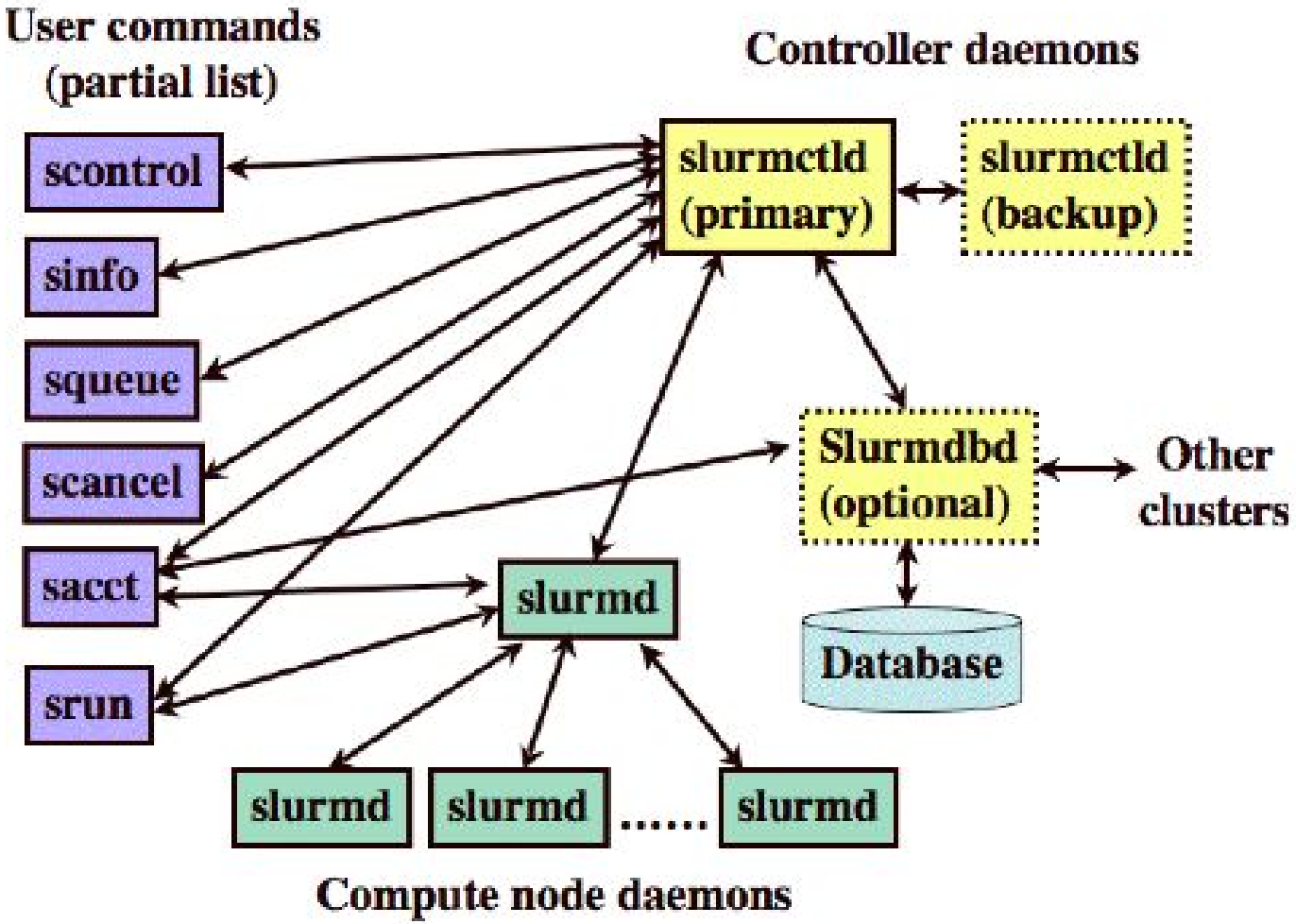
NERSC's Plan to Adopt SLURM

- SLURM has been in production on Babbage (Intel Xeon Phi KNC test bed) as of August 14.
- SLURM is the batch scheduler for Cori Phase 1.
- Hopper stays with Torque/Moab until retire.
- Edison stays with Torque/Moab before moving to CRT. Edison will come back online at CRT with SLURM.
- We use the "native" SLURM (as compared to the "hybrid" SLURM).

Advantages of Using SLURM

- **Fully open source.**
- **SLURM is extensible (plugin architecture)**
- **Low latency scheduling. Highly scalable.**
- **Integrated "serial" or "shared" queue**
- **Integrated Burst Buffer support**
- **Good memory management**
- **Built-in accounting and database support**
- **"Native" SLURM runs without Cray ALPS (Application Level Placement Scheduler)**
  - Batch script runs on the head compute node directly
  - Easier to use. Less chance for contention compared to shared MOM node.

Native SLURM Architecture



Running with SLURM

- Use “sbatch” (as “qsub” in Torque) to submit batch script or “salloc” (as “qsub -I” in Torque) to request interactive batch session.
- Use “srun” to launch parallel jobs (as with “aprun” with Torque/Moab or with hybrid SLURM)
- Most SLURM command options have two formats (long and short)
- Need to specify which shell to use for batch script.
- Environment is automatically imported (as “#PBS -V” in Torque)
- Lands on the submit directory (“cd $PBS_O_WORKDIR” not needed as in Torque)
- Batch script runs on the head compute node
- No need to repeat flags in the srun command if already defined in SBATCH keywords.
- srun flags overwrite SBATCH keywords
- srun does most of optimal process and thread binding automatically. Only flags such as “-n” “-c”, along with OMP_NUM_THREADS are needed for most applications. Advanced users can experiment more options such as –num_tasks_per_socket, –cpu_bind, --mem, etc.
- Hyperthreading is enabled by default. Jobs requesting more than 32 cores (MPI tasks * OpenMP threads) per node will use hyperthreads automatically.

Sample SLURM Batch Script

Long command options

```
#!/bin/bash -l

#SBATCH --partition=regular
#SBATCH --job-name=test
#SBATCH --account=mpccc
#SBATCH --nodes=2
#SBATCH --time=00:30:00

srun -n 16 ./mpi-hello
export OMP_NUM_THREADS=8
srun -n 8 -c 8 ./xthi
```

Short command options

```
#!/bin/bash -l

#SBATCH -p regular
#SBATCH -J test
#SBATCH -A mpccc
#SBATCH -N 2
#SBATCH -t 00:30:00

srun -n 16 ./mpi-hello
export OMP_NUM_THREADS=8
srun -n 8  -c 8 ./xthi
```

To submit a batch job:
% sbatch mytest.sl
Submitted batch job 15400

salloc and srun Example (interactive batch)

```
yunhe/> salloc -N 2 -p debug -t 30:00
salloc: Granted job allocation 16180
salloc: Waiting for resource configuration
salloc: Nodes nid00[408-409] are ready for job
yunhe@nid00408> export OMP_NUM_THREADS=8
yunhe@nid00408> srun –n 8 –c 8 ./xthi | sort -k4n -k6n
Hello from rank 0, thread 0, on nid00408. (core affinity = 0-7,32-39)
Hello from rank 0, thread 1, on nid00408. (core affinity = 0-7,32-39)
Hello from rank 0, thread 2, on nid00408. (core affinity = 0-7,32-39)
Hello from rank 0, thread 3, on nid00408. (core affinity = 0-7,32-39)
Hello from rank 0, thread 4, on nid00408. (core affinity = 0-7,32-39)
Hello from rank 0, thread 5, on nid00408. (core affinity = 0-7,32-39)
Hello from rank 0, thread 6, on nid00408. (core affinity = 0-7,32-39)
Hello from rank 0, thread 7, on nid00408. (core affinity = 0-7,32-39)
Hello from rank 1, thread 0, on nid00408. (core affinity = 16-23,48-55)
Hello from rank 1, thread 1, on nid00408. (core affinity = 16-23,48-55)
…
Hello from rank 1, thread 6, on nid00408. (core affinity = 16-23,48-55)
Hello from rank 1, thread 7, on nid00408. (core affinity = 16-23,48-55)
…
Hello from rank 4, thread 0, on nid00409. (core affinity = 0-7,32-39)
Hello from rank 4, thread 1, on nid00409. (core affinity = 0-7,32-39)
…
Hello from rank 4, thread 6, on nid00409. (core affinity = 0-7,32-39)
Hello from rank 4, thread 7, on nid00409. (core affinity = 0-7,32-39)
…
```

SLURM vs. Torque/Moab Keywords

Description	#PBS	#SBATCH
Queue	-q [queue]	--partition=[queue] or -p [queue]
Number of tasks	-n [count]	--ntasks=[count] or -n [count]
Node count	-l mppwidth=[count]*24	--nodes=[count] or -N [count]
Tasks per node	-l mppnppn=[count]	--ntasks-per-node=[count]
CPUs per task	(-d option in aprun)	--cpus-per-task=[count] or -c [count]
Wall clock limit	-l walltime=[hh:mm:ss]	--time=[days-hh:mm:ss] or -t [hh:mm:ss]
Standard output	-o [file]	--output=[file] or -o [file]
Standard error	-e [file]	--error=[file] or -e [file]
Combine stdout/stderr	-j oe (both to stdout)	default behavior if no --error or -e
Event notification	-m abe	--mail-type=[events]
Email address	-M [address]	--mail-user=[address]
Job name	-N [name]	--job-name=[name] or -J [name]
Account to charge	-A [account]	--account=[account] or -A [account]
Job restart	-r[y/n] (NERSC default is n)	--requeue or –no-requeue (NERSC default)
Job dependency	-W depend=[state:jobid]	--depend=[state:jobid] or -d [state:jobid]
Job arrays	-t [array_spec]	--array=array_spec or -a[array_spec]

SLURM vs. Torque/Moab Environment Variables

Description	Torque/Moab	SLURM
Job id	$PBS_JOBID	$SLURM_JOB_ID
Job name	$PBS_NODENAME	$SLURM_JOB_NAME
Submit directory	$PBS_O_WORKDIR	$SLURM_SUBMIT_DIR
Node list	$PBS_NODEFILE	$SLURM_NODELIST
Host submitted from	$PBS_O_HOST	$SLURM_SUBMIT_HOST
Nodes allocated	$PBS_NUM_NODES	$SLURM_JOB_NUM_NODES
Number cores/nodes	$PBS_NUM_PPN	$SLURM_CPUS_ON_NODE

SLURM User Commands

- **sbatch:** submit a batch script
- **salloc:** request nodes for an interactive batch session
- **srun:** launch parallel jobs
- **scancel:** delete a batch job
- **squeue:** display info about jobs in the queue
- **sinfo:** view SLURM configuration about nodes and partitions
- **scontrol**: view and modify SLURM configuration and job state
- **sacct:** display accounting data for jobs and job steps
- **sqs**: NERSC custom queue display *(subject to change)*

squeue and sinfo Examples

yunhe> squeue

```
    JOBID       USER  ACCOUNT       NAME  PARTITION    QOS NODES    TIME_LIMIT        TIME    ST
16179     keskital   planck     reproc      debug     low   128         30:00       10:51     R
    14645   psteinbr    m2078 l328f21b64    regular normal    32       2:30:00     1:47:53     R
    ...
    15974     heikki    m1820 ipnonsa_1e     shared normal     1       6:00:00       22:48     R
    15975     heikki    m1820 ipnonsa_1e     shared normal     1       6:00:00       22:29     R
    ...
    14734   psteinbr    m2078 l328f21b64    regular normal    32       2:30:00        0:00    PD
    14735   psteinbr    m2078 l328f21b64    regular normal    32       2:30:00        0:00    PD
    ...
    15944      zuntz      des spteg_disc    regular normal    32       6:00:00        0:00    PD
    15945      zuntz      des spteg_bulg    regular normal    32       6:00:00        0:00    PD
    15952      zuntz      des spteg_bulg    regular normal    32       6:00:00        0:00    PD
    14651   psteinbr    m2078 l328f21b64    regular normal     1       2:30:00     1:47:44    CG
```

yunhe> sinfo

```
PARTITION AVAIL JOB_SIZE  TIMELIMIT   CPUS  S:C:T   NODES STATE      NODELIST
debug*    up    1-infini      30:00     64 2:16:2    1478 allocated  nid[00024-00063,…]
debug*    up    1-infini      30:00     64 2:16:2     150 idle       nid[00408-00409,…]
regular   up    1-infini   12:00:00     64 2:16:2    1478 allocated  nid[00024-00063,…]
regular   up    1-infini   12:00:00     64 2:16:2     150 idle       nid[00408-00409,…]
realtime  down  1-infini    6:00:00     64 2:16:2    1478 allocated  nid[00024-00063,…]
realtime  down  1-infini    6:00:00     64 2:16:2     150 idle       nid[00408-00409,…]
shared    up    1          12:00:00     64 2:16:2      40 allocated  nid[00188-00191,…]
```

There are many other options. See man page.

scontrol Example

```
yunhe> scontrol show partition debug
PartitionName=debug
   AllowGroups=ALL AllowAccounts=ALL AllowQos=ALL
   AllocNodes=ALL Default=YES QoS=part_debug
   DefaultTime=00:10:00 DisableRootJobs=YES ExclusiveUser=NO GraceTime=0 Hidden=NO
   MaxNodes=UNLIMITED MaxTime=00:30:00 MinNodes=1 LLN=NO MaxCPUsPerNode=UNLIMITED
   Nodes=nid0[0024-0063,0080-0083,….2128-2175,2192-2239,2256-2303]
   Priority=2000 RootOnly=NO ReqResv=NO Shared=EXCLUSIVE PreemptMode=REQUEUE
   State=UP TotalCPUs=104192 TotalNodes=1628 SelectTypeParameters=N/A
   DefMemPerNode=UNLIMITED MaxMemPerNode=124928

yunhe> scontrol show job 16178
JobId=16178 JobName=mpi-hello.sl
   UserId=yunhe(18456) GroupId=yunhe(1018456)
   Priority=834180 Nice=0 Account=mpccc QOS=normal_regular_4
  …
   RunTime=00:00:00 TimeLimit=00:30:00 TimeMin=N/A
   SubmitTime=2015-11-06T05:48:11 EligibleTime=2015-11-06T05:48:11
   StartTime=2015-11-06T05:48:37 EndTime=2015-11-06T06:18:37
   …
   NodeList=nid00[408-409]
   BatchHost=nid00408
   NumNodes=2 NumCPUs=128 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=128,mem=249856,node=2
   Socks/Node=* NtasksPerN:B:S:C=2:0:*:* CoreSpec=*
   Command=/global/u1/y/yunhe/shared/mpi-hello.sl
   WorkDir=/global/u1/y/yunhe/shared
   StdErr=/global/u1/y/yunhe/shared/slurm-16178.out
   StdOut=/global/u1/y/yunhe/shared/slurm-16178.out
```

sacct Example

yunhe> sacct -u yunhe –start=11/3/15T9:00 –end=1/3/15T15:00 -o JobID,elapsed

JobID	Elapsed
13230	00:17:41
13230.0	00:18:08
13230.1	00:00:15
13230.2	00:08:17
13234	00:05:43
13234.0	00:06:13
13234.1	00:04:44
13237	00:30:23
13237.0	00:30:50
13237.1	00:00:14
13237.2	00:00:35
13237.3	00:00:03

There are many other options. See man page.

SLURM Features on Cori

- The Cluster Compatibility Mode (CCM) capability is implemented as an SBATCH command line option (--ccm), which enables SSH to compute nodes. No separate CCM partition is needed. CCM jobs can run in any partition now.
- Since sbatch or salloc lands on a compute node, applications such as “matlab” can be launched directly without CCM support.
- The native “shared” partition on Cori is not implemented as the Edison “serial” queue via CCM. It allows multiple users to share one node. Users can request more than 1 slot (either by tasks or memory) and be charged accordingly.
- Potential to expand or shrink job size at run time.
- Another batch server will be implemented to run “xfer”, “bigmem”, “workflow” jobs on one or more login nodes. Users can submit to either batch server in a single batch script.
- We will experiment batch job priority with combination of job size, queue wait time and fare share initially.

Current Batch Configuration

Partiton	Nodes	Physical Cores	Max Walltime	Relative Priority	Charge Factor
debug	1-64	1-2,048	30 min	2	Free
	65-128	2,049-4,096	30 min	3	Free
regular	1-2	1-64	12 hrs	4	Free
	3-256	65-8,192	6 hrs	4	Free
	257-512	8,193-16,384	6 hrs	4	Free
	513-1,024	16,385-32,768	4 hrs	4	Free
	1,025-1,428	32,769-45,696	2 hrs	4	Free
	1,429-1,628	45,697-52,096	2 hrs	4	Free
shared	1	1-32	12 hrs	4	Free
realtime	1-32	1-1,024	6 hrs	1	Free

Current Batch Policies

- Users can request modification to job priority by adding "#SBATCH --qos=premium" (disabled during free charging time) or "#SBATCH --qos=low" or "$SBATCH --qos=scavenger" in the batch script.
- Overall job priorities are a combination of partition, QOS, queue wait time, job size, wall time request, and fair share.
- There are no specific run limits or idle limits per job type/size category per user (except the debug run limit of 1 per user). Instead, we are experimenting with a RunMins limit (total remaining CPU minutes of all running jobs) on the per user/repo base.
- Debug jobs requesting more than 64 nodes have lower priority to discourage using the entire system for small number of jobs. No more than 128 nodes can be used for all debug jobs in the system at any time.
- Relative priorities for different sizes of the “regular” jobs will be adjusted in the future.
- Jobs from different users can share a node to run parallel or serial jobs in the "shared" partition. The maximum number of nodes a single job can use is 1. There are a total of 40 nodes in the system can be used for "shared" jobs.
- The "realtime" partition usage is by special permission only. There are a total of 32 nodes in the system can be used for "realtime" jobs. "realtime" jobs can choose to share a node by adding "#SBATCH --shared" in the batch script.

Summary

- SLURM provides equivalent or similar functionality with Torque/Moab.
- srun provides equivalent or similar process and thread affinity with aprun.
- Please let us know if you have an advanced or complicated workflow, and anticipate potential porting issues. We can work with you to migrate your scripts.
- Batch configurations are still subject to tunings and modifications before the system is in full production.

Documentations

- **SchedMD web page:**
  - http://www.schedmd.com/
- **Running Jobs on Cori**
  - https://www.nersc.gov/users/computational-systems/cori/running-jobs/
- **Man pages for slurm, sbatch, salloc, squeue, sinfo, sacct, scontrol, scancel, etc.**
- **Torque/Moab *vs.* SLURM Comparisons**
  - https://www.nersc.gov/users/computational-systems/cori/running-jobs/for-edison-users/torque-moab-to-slurm-transition-guide/
- **Running jobs on Babbage using SLURM:**
  - https://www.nersc.gov/users/computational-systems/testbeds/babbage/running-jobs-under-slurm-on-babbage/
- **Running iobs on Edison's test system (Alva) with native SLURM**
  - https://www.nersc.gov/users/computational-systems/edison/alva-test-and-development-system-for-edison/#toc-anchor-7

Thank you!

Popular features of a data intensive system and supporting them on Cori

Data Intensive Workload Need	Cori Solution
Local Disk	NVRAM 'burst buffer'
Large memory nodes	128 GB/node on Haswell; Option to purchase fat (1TB) login node
Massive serial jobs	Shared-Node/Serial queue on cori via SLURM
Complex workflows	***User Defined Images*** CCM mode Large Capacity of interactive resources
Communicate with databases from compute nodes	Advanced Compute Gateway Node
Stream Data from observational facilities	Advanced Compute Gateway Node
Easy to customize environment	***User Defined Images***
Policy Flexibility	Improvements coming with Cori: Rolling upgrades, CCM, MAMU, above COEs would also contribute

Shifter Delivers Performance – Pynamic