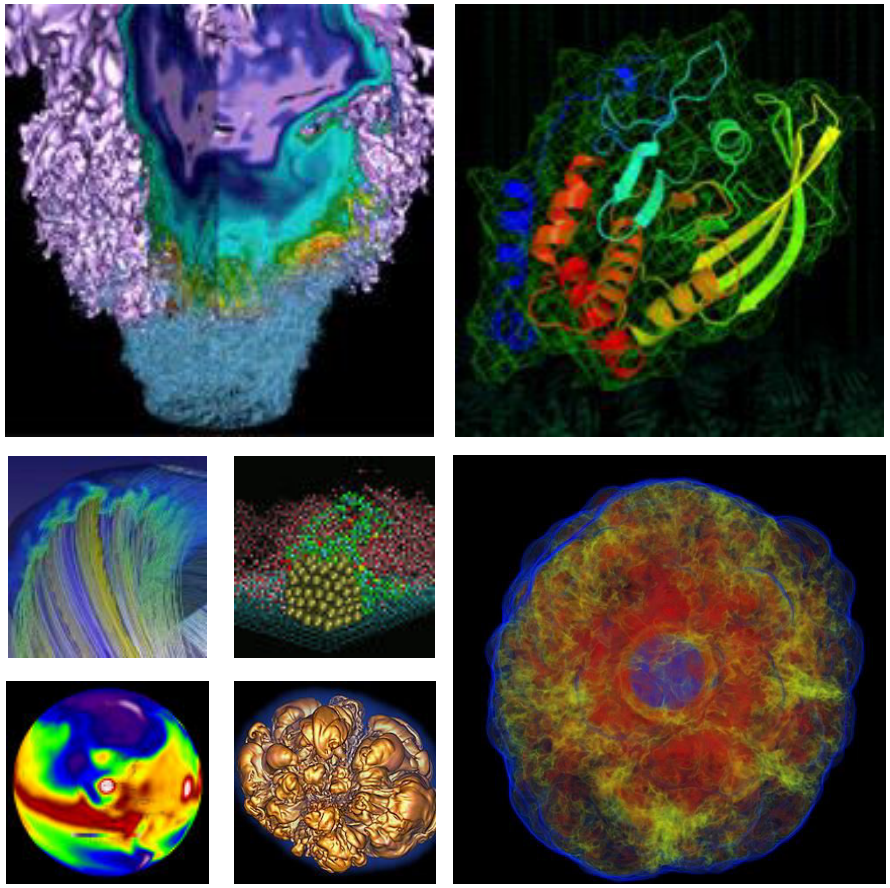
Introduction to NERSC Resources



Helen He

NERSC User Engagement Group

Computer Sciences Summer Student Program

June 24, 2019

NERSC 40 YEARS at the FOREFRONT 1974-2014

U.S. DEPARTMENT OF ENERGY | Office of Science

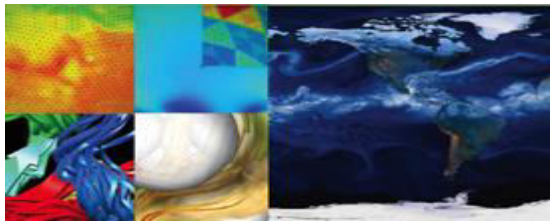
BERKELEY LAB

Outline

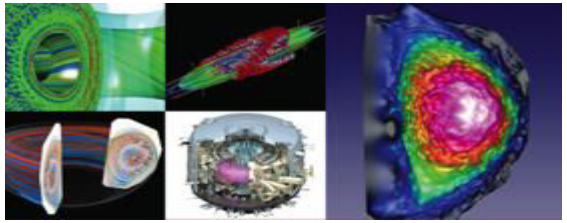
- **Systems Overview**
- Connecting to NERSC
- File Systems and Data Management/Transfer
- Data Analytics Software and Services
- Software Environment / Building Applications
- Running Jobs
- NERSC Online Resources
- Hands-on: Compiling and Running Jobs

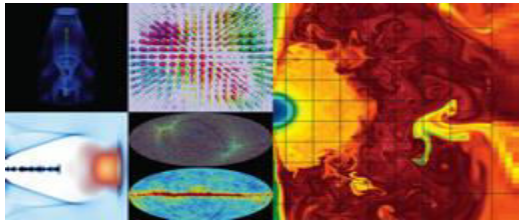
NERSC is the Mission HPC Computing Center for the DOE Office of Science

- **NERSC deploys advanced HPC and data systems for the broad Office of Science community**
- **NERSC staff provide advanced application and system performance expertise to users**
- **Approximately 7,000 users and 800 projects**
- **Over 2,000 publications cite using NERSC resources per year**
- **Founded in 1974, focused on open science**
- **Division of Lawrence Berkeley National Laboratory**



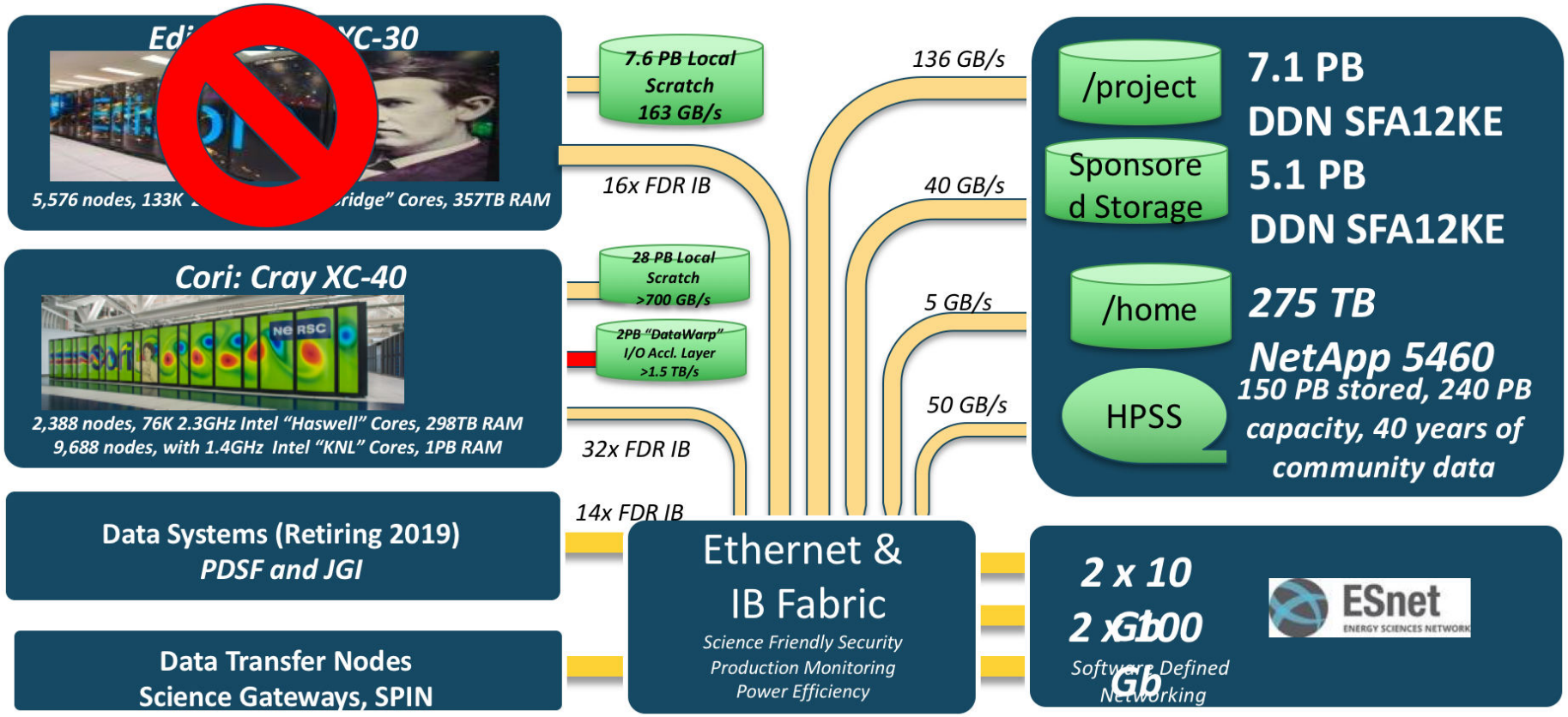
ASCR	Advanced Scientific Computing Research
BER	Biological & Environmental Research
BES	Basic Energy Sciences
FES	Fusion Energy Sciences
HEP	High Energy Physics
NP	Nuclear Physics
SBIR	Small Business Innovation Research





U.S. DEPARTMENT OF ENERGY | Office of Science

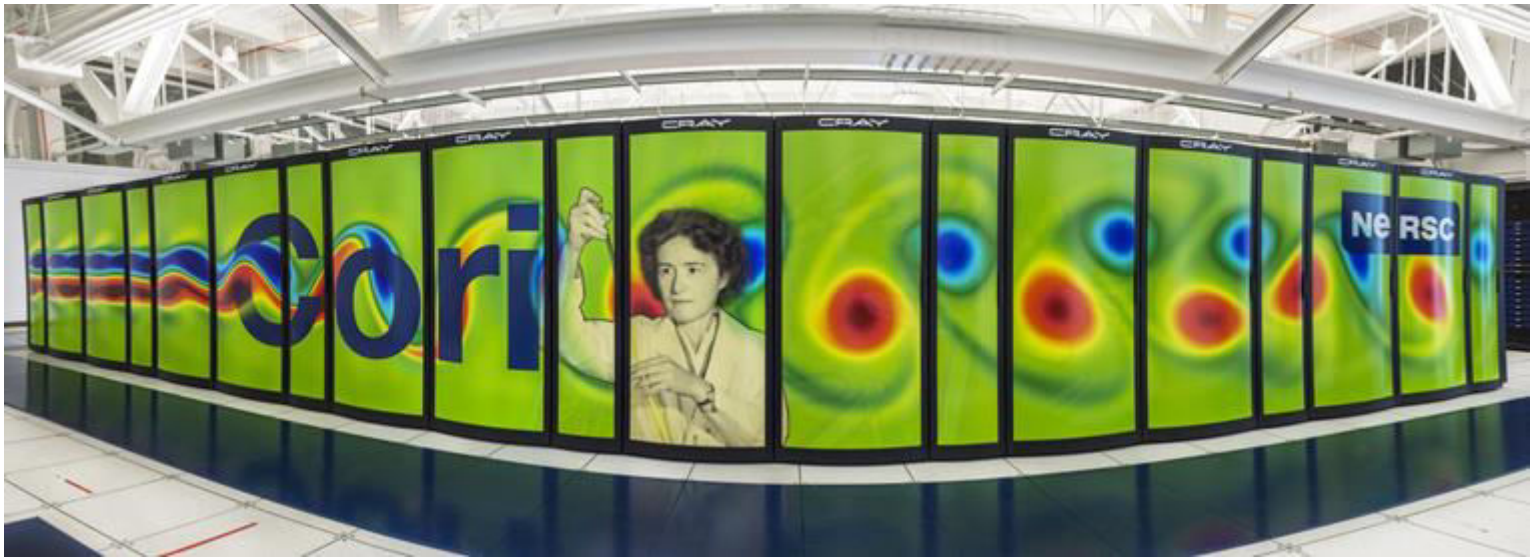
NERSC Systems



1 PF = 10^{15} Flops
1 PB = 10^{15} Bytes
1 TB = 10^{12} Bytes

Cori Brings HPC and Data Together

Cori: #14 in June 2019 (#5 in Nov 2016) Top500 list



Gerty Cori: Biochemist and first American woman to win a Nobel Prize in science

Phase I: 2388 x 32-core Intel Xeon "Haswell" 128 GB DDR4
Also known as "Data Partition" (76,416 cores total)

Phase II: 9688 x 68-core Intel Xeon Phi "KNL" 96 GB DDR4 + 16 GB MCDRAM
(658,784 total cores)

Outline

- Systems Overview
- **Connecting to NERSC**
- File Systems and Data Management/Transfer
- Data Analytics Software and Services
- Software Environment / Building Applications
- Running Jobs
- NERSC Online Resources
- Hands-on: Compiling and Running Jobs

Multi-Factor Authentication (MFA)

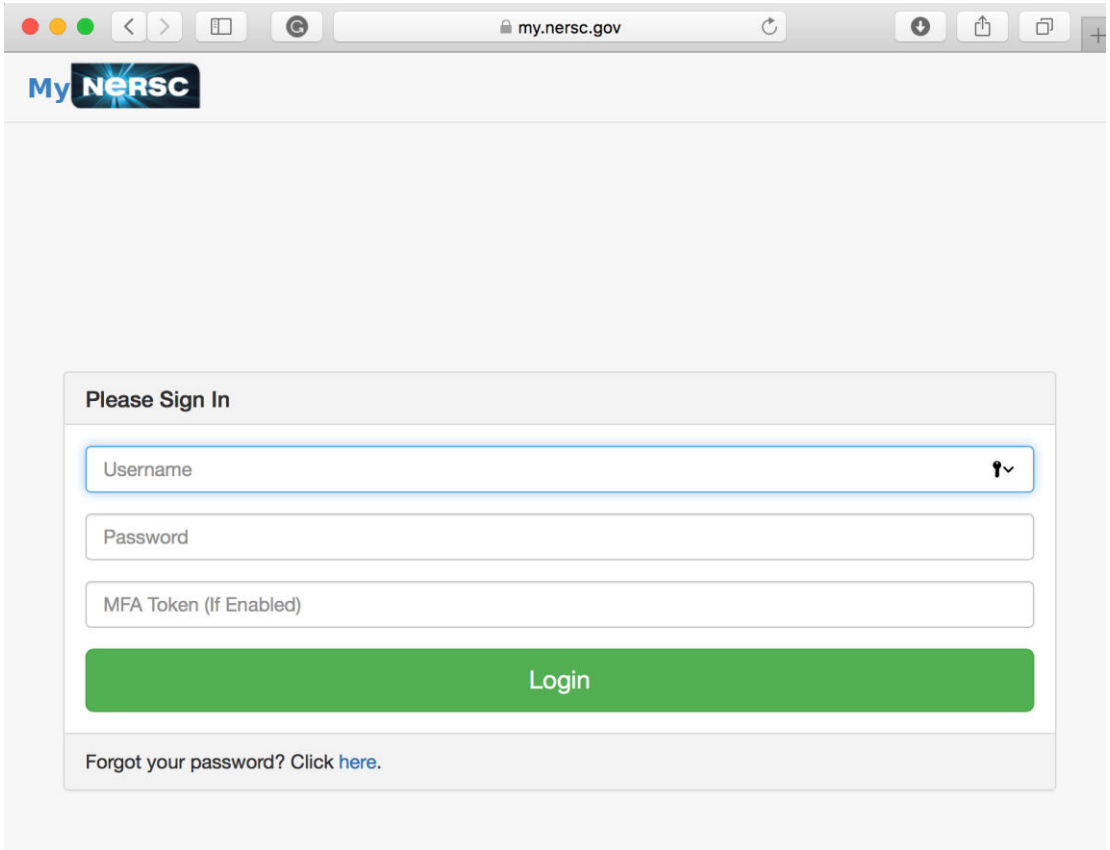
- **NIM password + OTP ("One-Time Password")**
  - OTP obtained via the “Google Authenticator” app on your smartphone
  - Alternative/backup option: Authy on desktop https://authy.com/

- **MFA is used in login to NERSC systems, web sites, and services**
  - Much harder for someone to hack your account

- **Mandatory**
  - except in special circumstances

MFA Examples

```
<laptop:yunhe>$ ssh -l yunhe cori.nersc.gov
…
Login connection to host cori01 :
Password + OTP:
```



Connecting to NERSC: SSH

- **All NERSC computational systems are accessible via `ssh`**
- **First: you need a terminal program on your desktop/laptop**
  - Mac: "terminal" (built-in) or "iTerm2" (https://www.iterm2.com/)
  - Windows: PuTTY (https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html)
  - Linux: Your own favorite :-)
- **If you will use X-forwarding (think GUI) (Note: NX is better!) then you also need an X server**
  - Mac: XQuartz (https://www.xquartz.org/)
  - Windows: Cygwin/X (http://x.cygwin.com/)
  - Linux: built in

Example Session (Terminal Only)

localhost:~elvis> ssh -l <NIM username> cori.nersc.gov

```
*****************************************************************
*                NOTICE TO USERS                                *
*                ---------------                                *
*                                                               *
*  Lawrence Berkeley National Laboratory operates this          *
*  computer system under contract to the U.S. Department of     *
*  Energy.  This computer system is the property of the United  *
*  States Government and is for authorized use only.  *Users    *
*  (authorized or unauthorized) have no explicit or implicit    *
*  expectation of privacy.*                                     *
*                                                               *
*......*                                                        *
*****************************************************************

Password: <enter your NIM password here>
```

Prompt on local system

Notification of acceptable use.

Password prompt
(reset password in NIM if needed)

You will login to one of the login nodes (12 on Cori).

To allow X-forwarding to access visualization programs, use the "-Y" flag:

localhost% ssh -l elvis -Y cori.nersc.gov

…
e/elvis> module load matlab
e/elvis> matlab
<MATLAB starts up>

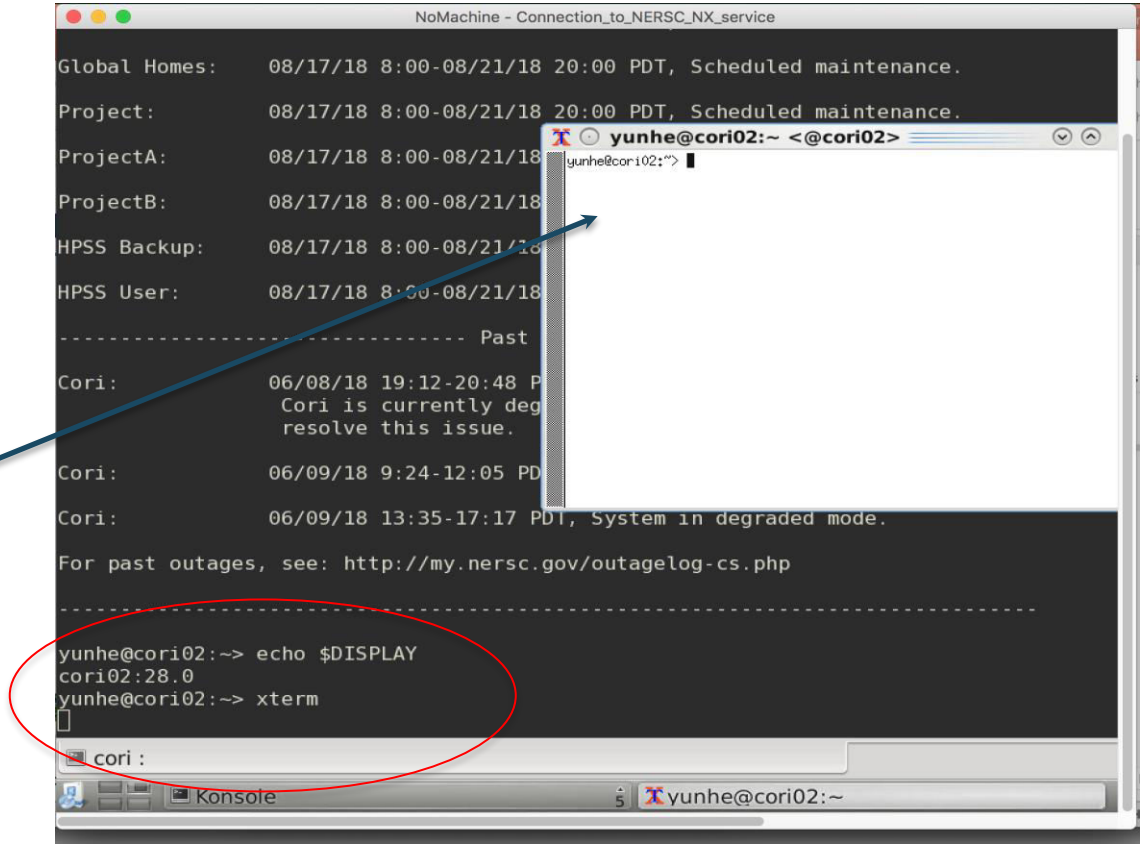
MFA (muti-factor authentication)

- SSH password + OTP (one-time-password).

Connecting to NERSC: NX (1)

- **NERSC recommends using NX instead of SSH X-forwarding since NX is faster and more reliable**
- **NX is a service for Accelerated X**

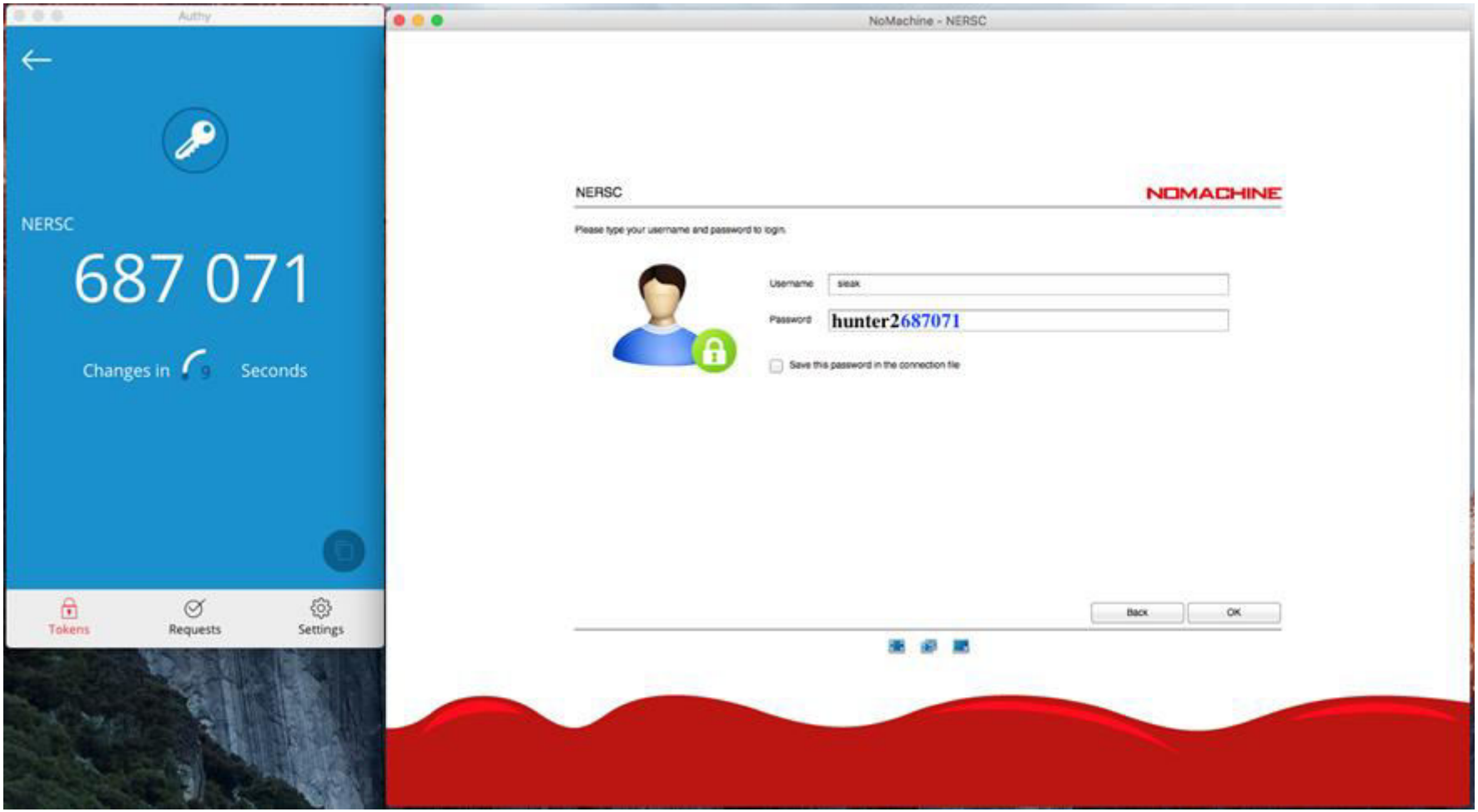
Opens a new xterm



Connecting to NERSC: NX (2)

- **NX also has the benefit of long lasting terminal sessions that can survive between lost internet connections**
  - Can reconnect later, even from a different location or computer
- **Download and install the Client software: NoMachine**
  - http://www.nersc.gov/users/connecting-to-nersc/using-nx
  - Works on Window/Mac/Linux
- **Or use NX Desktop from MyNERSC**
  - Is slower compared to NX Client

NX Login with MFA



Outline

- Systems Overview
- Connecting to NERSC
- **File Systems and Data Management/Transfer**
- Data Analytics Software and Services
- Software Environment / Building Applications
- Running Jobs
- NERSC Online Resources
- Hands-on: Compiling and Running Jobs

Global File Systems

Global Home

- Defined as $HOME (is default location when login)
- Permanent, relatively small storage
- Mounted on all platforms
- NOT tuned to perform well for parallel jobs
- Do not run large jobs from HOME
- Quota cannot be changed
- Snapshot backups (7-day history)
- Perfect for storing data such as source code, shell scripts

Global Project

- Each repo has a /project space at /global/project/projectdirs/<repo_name>
- Other custom project directories at /global/project/projectdirs/<project_name>
- Permanent, larger storage
- Mounted on all platforms
- Medium performance for parallel jobs
- Quota can be changed
- Snapshot backups (7-day history)
- Perfect for sharing data within research group

Local File Systems

Scratch

- Defined as $SCRATCH
- Large, temporary storage
- Optimized for read/write operations, NOT storage
- Not backed up
- Purge policy (Cori: 12 weeks, from last access time)
- Perfect for staging data and performing computations

Burst Buffer

- Temporary storage, can be per job or persistent for multiple users and jobs to access
- High-performance SSD file system
- Perfect for getting good performance in I/O-constrained codes

HPSS: Long Term Storage System

- **High-Performance Storage System**
- **Archival storage of infrequently accessed data**
- **Use hsi and htar to put/get files between NERSC computational systems and HPSS**
- **More info at**
  - https://docs.nersc.gov/filesystems/archive/

DTN: Dedicated Data Transfer System

- **Data Transfer Nodes (DTN) are dedicated servers tuned for moving data at NERSC**
  - Monitored bandwidth capacity between NERSC & other major facilities such as ORNL, ANL, BNL, SLAC…
  - Can be used to move data internally between NERSC systems and/or NERSC HPSS

- **Use NERSC DTNs to move large volumes of data in and out of NERSC or between NERSC systems**
- **More info at**
  - https://docs.nersc.gov/systems/dtn/

Globus Online

- **The recommended tool for moving data in & out of NERSC**
  - http://www.globus.org/ or http://globus.nersc.gov/
  - Reliable & easy-to-use web-based service:
    - Automatic retries
    - Email notification of success or failure
  - NERSC managed endpoints for optimized data transfers
- **NERSC documentation**
  - https://docs.nersc.gov/services/globus/
- **Globus extensive documentation https://docs.globus.org**
  - Web based interaction with service
  - REST/API for scripted interactions with service
  - Globus Connect Server & Personal for setting up additional remote endpoints such your personal laptop

Data Transfer General Tips

- **Use Globus Online for large, automated or monitored transfers**
- **scp is fine for smaller, one-time transfers (<100MB)**
  - But note that Globus is also fine for small transfers
- **Don’t use DTN nodes for non-data transfer purposes**
  - Use system login nodes for more general routine tasks
- **Don't use your $HOME directory**
  - Instead use /global/project, $SCRATCH … for better performance
- **Plain “cp” is still used for transfers within file systems**

Outline

- Systems Overview
- Connecting to NERSC
- File Systems and Data Management/Transfer
- **Data Analytics Software and Services**
- Software Environment / Building Applications
- Running Jobs
- NERSC Online Resources
- Hands-on: Compiling and Running Jobs

Cori's Data-Friendly Features

12 32-core Haswell
500 GB Login Nodes

Jupyter
Notebook
Node

768 GB "bigmem"
Haswell Nodes

Pipeline/Workflow
Management Nodes

Serial Queue
Shared-node Queue
Transfer Queue

Containerized
Environments

Real-Time Queues
for Co-Scheduling
w/Experiments

Streaming Data to
Compute Nodes

Cray DataWarp:
Burst Buffer for
I/O acceleration

Interactive Queue:
64 Nodes x 4 Hours

External Network
Access to/from
Compute Nodes

Production Software Stack

Capabilities	Technologies
Data Transfer + Access	globus online, GridFTP, jupyter, !m, python django, newt
Workflows	FireWorks, TaskFarmer
Data Management	HDF, netCDF, ROOT Data Analysis Framework, mongoDB, MySQL, PostgreSQL
Data Analytics	python, Spark, TensorFlow, K, R, julia, Caffe, PYTORCH
Data Visualization	visit, ParaView

Data Analytics Software and Services

- **Science Gateways**
- **Databases**
- **Shifter**
- **Burst Buffer**
- **Python**
- **Jupyter**
- **And more …**

Access for External Collaborators (Science Gateways)

- **NERSC supports project-level public http access**
  - Project specific area can be created:
    - /global/project/projectdirs/<your_project>/www
  - These are available for public access under the URL:
    - http://portal.nersc.gov/project/<your_project>

- **Each repo has a /project space, can publish as above**
- **Special Science Gateways can be created**
  - Details at: https://docs.nersc.gov/services/science-gateways/

Shifter

- **Shifter is a open-source software stack that enables users to run custom environments on HPC systems**
- **Compatible with the popular Docker container format so users can easily run Docker containers on NERSC systems**
- **Users can develop an application on own desktop, use the same stack of Linux OS and software of choice to on Cori and Edison**
- **Saves porting effort, improves reproducibility**
- **More info at**
  - https://www.nersc.gov/users/software/using-shifter-and-docker/using-shifter-at-nersc/
- **Pulls an image at run time**

Use Burst Buffer for Faster IO

- **Cori has 1.8PB of SSD-based "Burst Buffer" to support I/O intensive workloads**
- **Jobs can request a job-temporary BB filesystem, or a persistent (up to a few weeks) reservation**
- **More info at**
  - https://docs.nersc.gov/jobs/examples/#burst-buffer

Python

NERSC Python is Anaconda

- **Handy package management and deployment (conda tool)**
- **conda environment replaces virtualenv**

Do not use /usr/bin/python, instead:

```
module load python/2.7-anaconda-4.4
    (default, for no more than 6 months from now)
  or
module load python/3.6-anaconda-4.4
```

More info at

https://docs.nersc.gov/programming/high-level-environments/python/

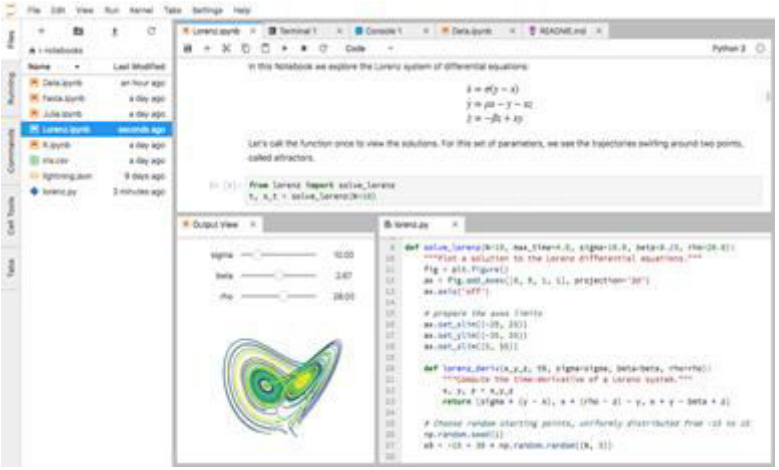
Parallel with Python

- **Within a node**
  - Use OpenMP-threaded math libs
  - Multiprocessing is OK too
- **Multi-node parallelism**
  - Best supported by mpi4py
  - Dask, PySpark work too
- **Hybrid parallelism**
  - Best route is mpi4py + threaded math libs
- **Best to use shifter to scale up Python**

Jupyter

Jupyter Notebook: "Literate Computing."
Code, text, equations, viz in a narrative.

Use JupyerHub for Jupyter at NERSC.
https://jupyter.nersc.gov/

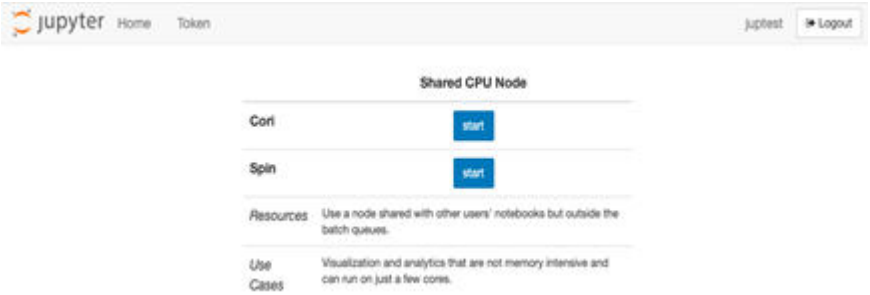


Cori Shared CPU Node:

- **Launches notebooks on Cori**
- **Can see Cori $SCRATCH**
- **Same Python env as ssh login**
- **Can submit jobs via `%sbatch`**

Spin Shared CPU Node:

- **External to Cori**
- **Can't see $SCRATCH**
- **But can see /project, $HOME**



Databases

- **Relational / SQL Databases**
  - MySQL and PostgreSQL, good for:
    - structured data (have a ‘Schema’)
    - Relational (tables of rows and columns)
    - Mid-Size, <= several GB in total
- **NoSQL / Schema-less Databases**
  - MongoDB, good for:
    - Un-Structured Data (‘Schema-less’)
    - Mid-Size to Large, e.g. 10 GB of Text
- **More info and how to request a database:**
  - https://docs.nersc.gov/services/databases/

Outline

- Systems Overview
- Connecting to NERSC
- File Systems and Data Management/Transfer
- Data Analytics Software and Services
- **Software Environment / Building Applications**
- Running Jobs
- NERSC Online Resources
- Hands-on: Compiling and Running Jobs

Software

- **Cray supercomputers OS is a version of Linux**
- **Compilers are provided on machines**
- **Libraries: many libraries are provided by vendor, many others provided by NERSC**
- **Applications: NERSC compiles and supports many software packages (such as chemistry and materials sciences packages) for our users**

Modules Environment

- **Modules are used to manage the user environment**

module	
list	To list the modules in your environment
avail avail -S	To list available modules For example: To see all available modules: % module avail To see all available *netcdf* modules: % module avail -S netcdf
load/unload	To load or unload module
show/display	To see what a module loads
whatis	Display the module file information
swap/switch	To swap two modules For example: to swap architecture target from Haswell to KNL % module swap craype-haswell craype-mic-knl
help	General help: $module help Information about a module: $ module help PrgEnv-cray

https://docs.nersc.gov/environment/#nersc-modules-environment

Default Loaded Modules

```
yunhe@cori09:~> module lis
Currently Loaded Modulefiles:
  1) modules/3.2.10.6                          13) job/2.2.3-6.0.7.1_5.44__g6c4e934.ari
  2) nsg/1.2.0                                 14) dvs/2.7_2.2.120-6.0.7.1_12.1__g74cb2cc4
  3) intel/18.0.1.163                          15) alps/6.6.43-6.0.7.1_5.46__ga796da32.ari
  4) craype-network-aries                      16) rca/2.2.18-6.0.7.1_5.48__g2aa4f39.ari
  5) craype/2.5.15                             17) atp/2.1.3
  6) cray-libsci/18.07.1                                                   18) PrgEnv-intel/6.0.4
  7) udreg/2.3.2-6.0.7.1_5.13__g5196236.ari                                19) craype-haswell
  8) ugni/6.0.14.0-6.0.7.1_3.13__gea11d3d.ari                              20) cray-mpich/7.7.3
  9) pmi/5.0.14                                                            21) gcc/7.3.0
 10) dmapp/7.1.1-6.0.7.1_6.2__g45d1b37.ari                                 22) altd/2.0
 11) gni-headers/5.0.12.0-6.0.7.1_3.11__g3b1768f.ari                       23) darshan/3.1.4
 12) xpmem/2.2.15-6.0.7.1_5.11__g7549d06.ari
```

3) Compiler 6) Cray Scientific Libraries
18) Programing Environment. 19) Target architecture Driver. 20) MPI Libraries

Cross-Compile is Needed

- **Cori: Haswell compute nodes and KNL compute nodes**
- **All Cori login nodes are Haswell nodes**
- **We need to cross-compile**
  - Directly compile on KNL compute nodes is very slow
  - Compiles on login nodes; Executables runs on compute nodes

Software Environment

- **Cross-Compilation Environment**
  - Binaries built for Haswell can run on KNL nodes, but not vice versa
  - Recommends to build separate binaries for each architecture to take advantage of optimizations unique to processor type
- **Available compilers: Intel, GNU, Cray**
- **Use compiler wrappers to build. Do not use native compilers such as ifort, mpiicc, etc.**
  - ftn for Fortran codes: **% ftn my_code.F90**
  - cc for C codes: **% cc my_code.c**
  - CC for C++ codes: **% CC my_code.cc**
  - Compiler wrappers add header files and link in MPI and other loaded Cray libraries by default
  - Builds applications statically by default. Can add "-dynamic" to build dynamically if chosen

How to Compile for KNL

- **The default loaded architecture target module is "craype-haswell" on the Haswell login nodes.**
- **This module sets CRAY_CPU_TARGET to haswell**

	Intel	GNU	Cray	Module
Cori Haswell	-xCORE-AVX2 or -xHASWELL	-march=core-avx2	-h cpu=haswell	craype-haswell
Cori KNL	-xMIC-AVX512 or -xKNL	-march=knl	-h cpu=mic-knl	craype-mic-knl
Edison IvyBridge	-xCORE-AVX-I	-march=corei7-avx	-h cpu=ivybridge	craype-ivybridge

- **Best recommendation to build for KNL target**

  % module swap craype-haswell craype-mic-knl

  which will set CRAY_CPU_TARGET to mic-knl

Building Simple Test Program Example

- **To build on Cori Haswell:**
  - Using default Intel compiler:
    % ftn -o mytest mytest_code.F90
  - Using Cray compiler, do the following first before “ftn”
    % module swap PrgEnv-intel PrgEnv-cray
    % ftn -o mytest mytest_code.F90

- **To build on Cori KNL:**
  - Using default Intel compiler
    % module swap craype-haswell craype-mic-knl
    % ftn -o mytest mytest_code.F90
  - Using Cray compiler, do the following first before “ftn”
    % module swap PrgEnv-intel PrgEnv-cray
    % module swap craype-haswell craype-mic-knl
    % ftn -o mytest mytest_code.F90

Compiler Flags

Intel	GNU	Cray	Description/ Comment
-O2	-O0	-O2	default
default , or -O3	-O2 or -O3,-Ofast	default	recommended
-qopenmp	-fopenmp	default, or -h omp	OpenMP
-g	-g	-g*	debug
-v	-v	-v	verbose

- **Default behaviors may vary between compilers**
  - Default number of OpenMP threads
    - all CPU slots available for Intel and GNU compilers
    - 1 for Cray compiler
  - * -g: Cray compiler disables all optimizations, try -G2

Compiler Recommendations

- **Will not recommend any specific compiler**
  - Intel - better chance of getting processor specific optimizations, especially for KNL
  - Cray compiler – many new features and optimizations, especially with Fortran
  - GNU - widely used by open software
- **Try different compilers for potential performance improvement**
  - Start with the compilers that vendor/code developers used to minimize the chance of hitting compiler and code bugs

Linking Considerations

- **Compiler wrapper will Linking with Cray MPI (cray-mpich module is loaded by default), Cray Scientific libraries (cray-libsci module is loaded by default), and most Cray provided libraries and some NERSC provided libraries (need to load corresponding modules) are automatically**
  - % CC parallel_hello.cpp
  - % ftn dgemmx1.f90
  - % module load cray-hdf5
  - % cc h5write.c
- **To link with most NERSC provided libraries, extra include path and libraries need to be added manually, which are usually defined in module files for convenience, such as:**
  - % module load gsl
  - % ftn test3.f90 $GSL
  - Use "module show gsl" to see how $GSL is defined
- **To link with Intel MKL (Math Kernel Libraries) with Intel compiler, use the "-mkl" flag**
  - % ftn test1.f90 -mkl # default to parallel -multi-threaded lib
  - The loaded "cray-libsci" will be ignored if -mkl is used

NERSC Software Environment Policy

- **Maintain 4 CDT (Cray Development Tools) versions on the system at a time**
  - Users have access to any given software version for at least 18 months (except when OS upgrade etc.)
- **Install new CDT every 3 months (Mar, Jun, Sept, Dec releases)**
- **Change software default versions every 6 months (Jan and July)**
  - Mostly for changing Cray provided software versions
  - Promotes to rebuild NERSC supported packages with these new defaults
- **Fix critical bugs as needed with user notices**
- **Provide pe_archive module for archived older versions**
- **Timelines to find out when changes were made**
  - Cori: https://www.nersc.gov/users/computational-systems/cori/cori-timeline/

Outline

- Systems Overview
- Connecting to NERSC
- File Systems and Data Management/Transfer
- Data Analytics Software and Services
- Software Environment / Building Applications
- **Running Jobs**
- NERSC Online Resources
- Hands-on: Compiling and Running Jobs

Jobs at NERSC

- **Most are parallel jobs (10s to 100,000+ cores)**
- **Also a number of "serial" jobs**
  - Typically "pleasantly parallel" simulation or data analysis
- **Production runs execute in batch mode**
- **Our batch scheduler is SLURM (Simple Linux Utility for Resource Management), performs regular scheduling and backfill scheduling**
- **Debug jobs and interactive jobs are supported for up to 30 min and 4 hrs respectively**
- **Typically run times are a few to 10s of hours**
  - Limits are necessary because of MTBF and the need to accommodate 6,000 users' jobs

Login Nodes and Compute Nodes

- **Login nodes (external)**
  - Edit files, compile codes, submit batch jobs, etc.
  - Run short, serial utilities and applications
  - Cori has Haswell login nodes
- **Compute nodes**
  - Execute your application
  - Dedicated resources for your job
  - Cori has Haswell and KNL compute nodes.
  - Binaries built for Haswell can run on KNL nodes, but not vice versa

Submitting Batch Jobs

- **To run a batch job on the compute nodes you must write a "batch script" that contains**
  - Directives to allow the system to schedule your job
  - An `srun` command that launches your parallel executable
- **Submit the job to the queuing system with the sbatch or salloc command**

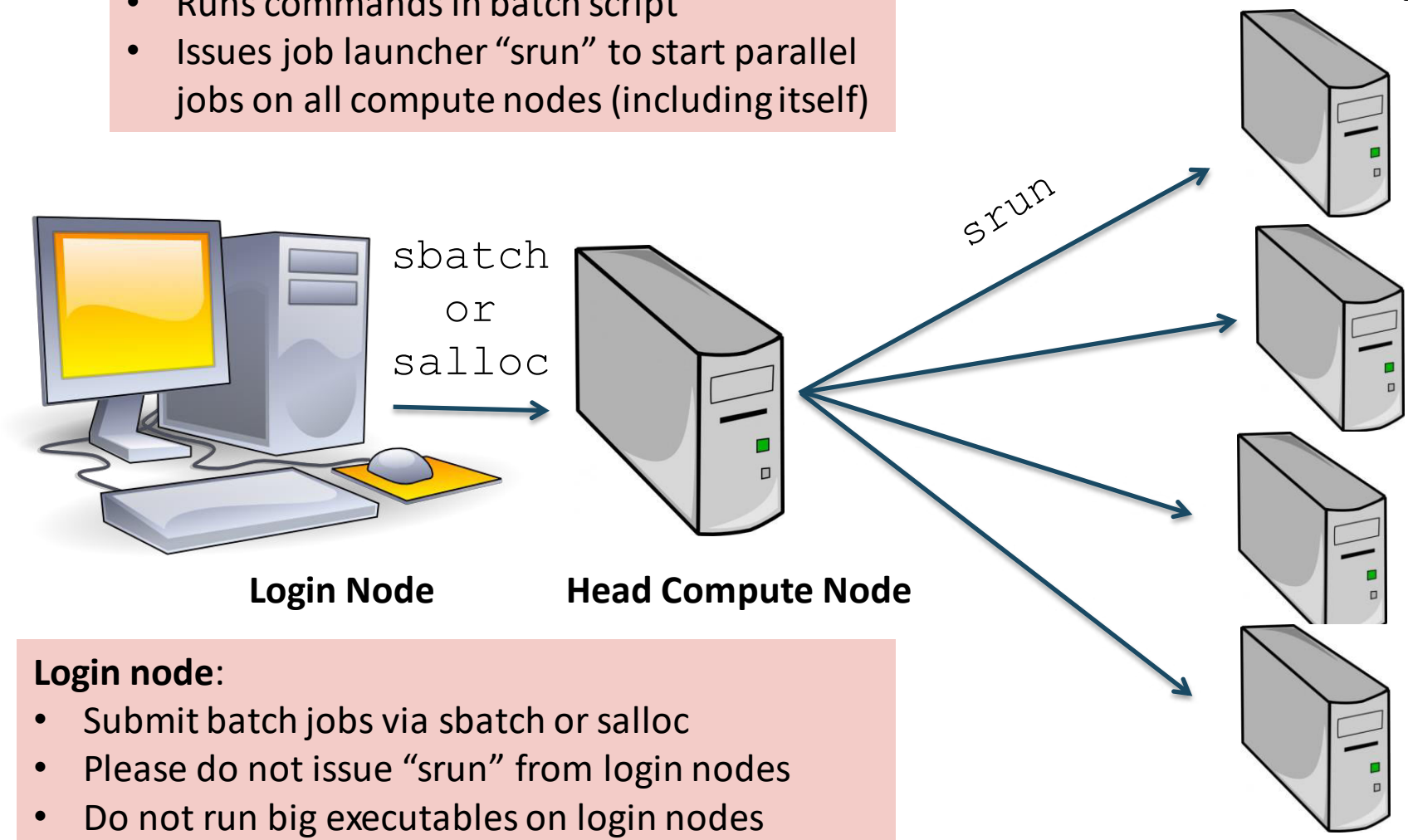
```
% sbatch my_batch_script
```

or

```
% salloc <command line options>
```

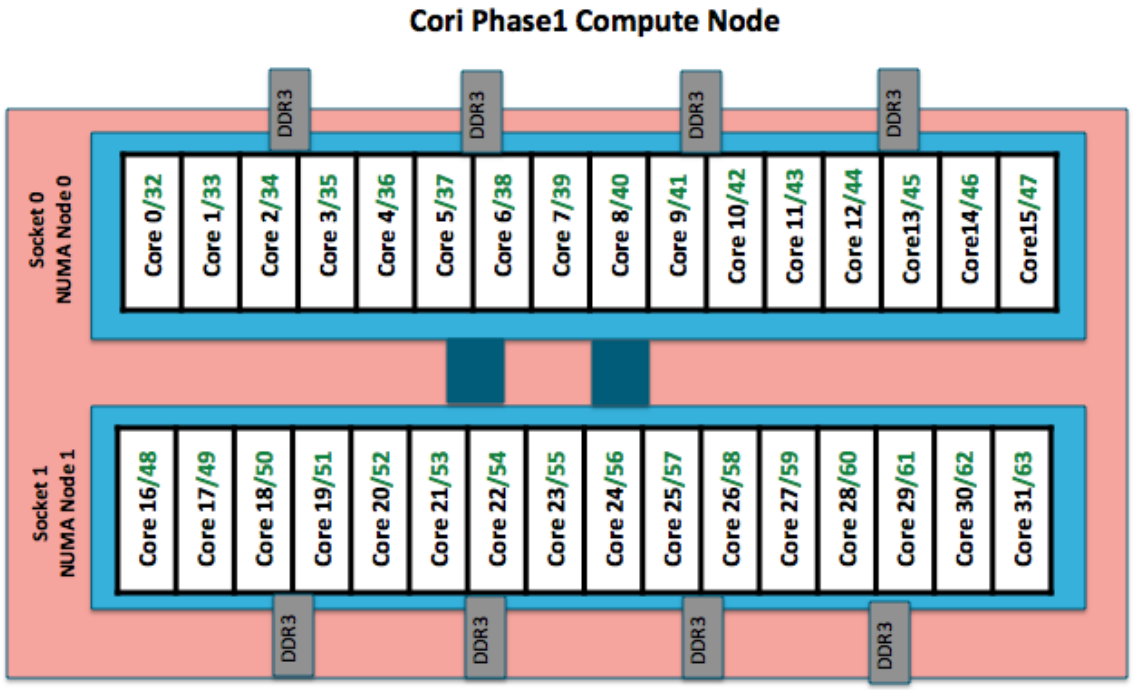
Launching Parallel Jobs with SLURM

Head compute node:
- Runs commands in batch script
- Issues job launcher "srun" to start parallel jobs on all compute nodes (including itself)



Login node:
- Submit batch jobs via sbatch or salloc
- Please do not issue "srun" from login nodes
- Do not run big executables on login nodes

Cori Haswell Compute Nodes



To obtain processor info:

Get on a compute node:
% salloc -N 1 -C …

Then:
% numactl -H
or % cat /proc/cpuinfo
or % hwloc-ls

- **Cori Haswell Each node has 2 Intel Xeon 16-core Haswell processors**
  - **2 NUMA domains (sockets) per node, 16 cores per NUMA domain. 2 hardware threads per core. Physical core**
  - **NUMA Domain 0: physical cores 0-15 (and logical cores 32-47);**
    NUMA Domain 1: physical cores 16-31 (and logical cores 48-63)
- **Memory bandwidth is non-homogeneous among NUMA domains.**

Similar for Edison, except each socket has 12 physical cores

Cori KNL Example Compute Nodes

- **A Cori KNL node has 68 cores/272 CPUs, 96 GB DDR memory, 16 GB high bandwidth on package memory (MCDRAM)**
- **Compute nodes can be booted in 3 cluster modes, all-to-all, quadrant, sub-NUMA clustering, and 3 memory modes (cache, flat, hybrid)**
- **Default mode on Cori is quad,cache**

Arrangement of Hardware Threads for 68 Core KNL

Core # →	0	1	2	3	...	16	17	18	...	33	34	35	...	50	51	52	...	65	66	67
HW Thread #	0	1	2	3	...	16	17	18	...	33	34	35	...	50	51	52	...	65	66	67
	68	69	70	71	...	84	85	86	...	101	102	103	...	118	119	120	...	133	134	135
	136	137	138	139	...	152	153	154	...	169	170	171	...	186	187	188	...	201	202	203
	204	205	206	207	...	220	221	222	...	237	238	239	...	254	255	256	...	269	270	271

- A quad,cache node (default setting) has only 1 NUMA node with all CPUs on the NUMA node 0 (DDR memory). MCDRAM is hidden from the "numactl -H" result since it is a cache.

Other combinations are by reservation only: such as quad,flat or snc2,flat

My First Hello World Program

my_batch_script

```
#!/bin/bash -l
#SBATCH -q debug
#SBATCH -N 2
#SBATCH -t 10:00
#SBATCH -C haswell
#SBATCH -L SCRATCH
#SBATCH -J myjob
srun -n 64 ./helloWorld
```

To run via batch queue

```
% sbatch my_batch_script
```

To run via interactive batch

```
% salloc -N 2 -q interactive -C haswell -t 10:00
<wait_for_session_prompt. Land on a compute node>
% srun -n 64 ./helloWorld
```

Cori Haswell Batch Script

my_batch_script

```
#!/bin/bash -l
#SBATCH -q debug
#SBATCH -N 2
#SBATCH -t 10:00
#SBATCH -C haswell
#SBATCH -L SCRATCH
#SBATCH -J myjob
srun -n 64 ./helloWorld
```

To run via batch queue

```
% sbatch my_batch_script
```

To run via interactive batch

```
% salloc -N 2 -q interactive -C haswell -t 10:00
<wait_for_session_prompt. Land on a compute node>
% srun -n 64 ./helloWorld
```

Sample Cori Haswell Batch Script - MPI

```
#!/bin/bash -l
#SBATCH -q regular
#SBATCH -N 4
#SBATCH -t 1:00:00
#SBATCH -C haswell
#SBATCH -L SCRATCH
#SBATCH -J myjob

export OMP_NUM_THREADS=1
srun -n 1280 -c 2 --cpu_bind=cores ./mycode.exe
```

- Need to specify which shell to use for batch script
- Use “-l” as login shell is optional
- Environment is automatically imported

Sample Cori Haswell Batch Script - MPI

```
#!/bin/bash -l
#SBATCH -q regular
#SBATCH -N 4
#SBATCH -t 1:00:00
#SBATCH -C haswell
#SBATCH -L SCRATCH
#SBATCH -J myjob

export OMP_NUM_THREADS=1
srun -n 1280 -c 2 --cpu_bind=cores ./mycode.exe
```

Job directives: instructions for the batch system

- Submission QOS (default is “debug”)
- How many compute nodes to reserve for your job
- How long to reserve those nodes
- What type of compute nodes to use.
- More optional SBATCH keywords

Sample Cori Haswell Batch Script - MPI

```
#!/bin/bash -l
#SBATCH -q regular
#SBATCH -N 40
#SBATCH -t 1:00:00
#SBATCH -C haswell
#SBATCH -L SCRATCH
#SBATCH -J myjob

export OMP_NUM_THREADS=1
srun -n 1280 -c 2 --cpu_bind=cores ./mycode.exe
```

SBATCH optional keywords:

- What file systems my job depends on (prevent to start when there are file system issues)
- what to name my job
- what to name STDOUT files
- what account to charge
- whether to notify you by email when your job finishes
- …

Sample Cori Haswell Batch Script - MPI

```
#!/bin/bash -l
#SBATCH -q regular
#SBATCH -N 40
#SBATCH -t 1:00:00
#SBATCH -C haswell
#SBATCH -L SCRATCH
#SBATCH -J myjob

export OMP_NUM_THREADS=1
srun -n 1280 -c 2 --cpu_bind=cores ./mycode.exe
```

- There are 64 logical CPUs (the number Slurm sees) on each node
- "-c" specifies # logical CPUs to be allocated to each MPI task
- --cpu_bind is critical especially when nodes are not fully occupied
  - use "--cpu_bind=cores" when #MPI tasks <= # physical cores per node
  - use "--cpu_bind=threads" when #MPI tasks ># physical cores per node
- With 40 nodes, using hyperthreading, up to 40*64=2,560 MPI tasks can be launched: "srun -n 2560 -c 1 --cpu_bind=threads ./mycodee.exe" is OK

Sample Cori Haswell Batch Script - Hybrid MPI/OpenMP

```
#!/bin/bash -l
#SBATCH -q regular
#SBATCH -N 40
#SBATCH -t 1:00:00

export OMP_NUM_THREADS=8
export OMP_PROC_BIND=true
export OMP_PLACES=threads

srun -n 160 -c 16 --cpu_bind=cores ./mycode.exe
```

- Set OMP_NUM_THREADS
- Use OpenMP standard settings for process and thread affinity
- Again, "-c" specifies # logical CPUs to be allocated to each MPI task
  - with 4 MPI tasks per node, set 64 logical CPUs /4 =16 for "-c"
  - "-c" value should be >= OMP_NUM_THREADS

Serial Jobs

- **The "shared" QOS allows multiple executables from different users to share a node**
- **Each serial job run on a single core of a "shared" node**
- **Up to 32 (Cori Haswell) jobs from different users depending on their memory requirements**

```
#SBATCH -q shared
#SBATCH -t 1:00:00
#SBATCH --mem=4GB
#SBATCH -C haswell
#SBATCH -J my_job
./mycode.x
```

- Do not specify #SBATCH -N"
- Default "#SBATCH -n" is 1
- Default memory is 1,300 MB for Edison and 1,952 MB for Haswell
- Use -n or --mem to request more slots for larger memory
- **Do not use "srun" for serial executable (reduces overhead)**

- **Small parallel job that use less than a full node can also run in the "shared" partition**
- **https://docs.nersc.gov/jobs/best-practices/#serial-jobs**

How to Run Debug and Interactive Jobs

- **You can run small parallel jobs interactively**
- **Debug**
  - Max 512 nodes, 30 min, run limit 2, submit limit 5

% salloc -N 20 -q debug -C knl,quad,cache -t 30:00

- **Interactive (highly recommend to use this!!)**
  - Instant allocation (get nodes in 5 min or reject), run limit 2, submit limit 2
  - max walltime 4 hrs, up to 64 nodes on Cori (Haswell and KNL) per repo

% salloc -N 2 -q interactive -C knl,quad,cache -t 2:00:00

- https://docs.nersc.gov/jobs/examples/#interactive
- https://docs.nersc.gov/jobs/interactive/

Advanced Workflow Management

- **Bundle jobs (multiple "srun"s in one script, sequential or simultaneously)**
- **Use Job Arrays for submitting and managing collections of similar jobs**
- **Use job dependency features to chain jobs that have dependency**
- **Run variable-time jobs**
- **Use Burst Buffer for faster IO**
- **Use Shifter for jobs with custom user environment**
- **Run "xfer" for transferring to/from HPSS**
- **Run "bigmem" for large memory jobs**

Bundle Jobs

Multiple Jobs Sequentially:

```
#!/bin/bash -l
#SBATCH -q regular
#SBATCH -N 100
#SBATCH -t 12:00:00
#SBATCH -J my_job
#SBATCH -o my_job.o%j
#SBATCH -L project,SCRATCH
#SBATCH -C haswell
srun -n 3200 ./a.out
srun -n 3200 ./b.out
srun -n 3200 ./c.out
```

- need to request largest number of nodes needed
- https://docs.nersc.gov/jobs/examples/#multiple-parallel-jobs-sequentially

Multiple Jobs Simultaneoulsy

```
#!/bin/bash -l
#SBATCH -q regular
#SBATCH -N 9
#SBATCH -t 12:00:00
#SBATCH -J my_job #SBATCH -o my_job.o%j
#SBATCH -L project
#SBATCH -C haswell
srun -n 44 -N 2 -c2 --cpu_bind=cores ./a.out &
srun -n 108 -N 5 -c2 --cpu_bind=cores ./b.out &
srun -n 40 -N 2 -c2 --cpu_bind=cores ./c.out &
wait
```

- need to request total number of nodes needed
- No applications are shared on the same nodes
- make sure to use “&” (otherwise run in sequential) and “wait” (otherwise job exit immediately)
- https://docs.nersc.gov/jobs/examples/#multiple-parallel-jobs-simultaneously

Job Arrays

```
# Submit a job array with index values between 0 and 31:
$ sbatch --array=0-31 -N1
# Submit a job array with index values of 1, 3, 5 and 7:
$ sbatch --array=1,3,5,7 -N1 -n2
# Submit a job array with index values between 1 and 7 with a step size of 2 (i.e. 1, 3, 5, 7)
$ sbatch --array=1-7:2 -N1 -q regular
```

```
#!/bin/bash -l
#SBATCH -q regular
#SBATCH -N 1
#SBATCH -t 00:05:00
#SBATCH --array=1-10
#SBATCH -L SCRATCH
#SBATCH -C haswell

cd test_$SLURM_ARRAY_JOB_ID
srun ./mycode.exe
```

- Better managing jobs, not necessary faster turnaround
- Each array task is considered a single job for scheduling
- Use $SLURM_ARRAY_JOB_ID for each individual array task
- https://docs.nersc.gov/jobs/examples/#job-arrays

Dependency Jobs

```
cori% sbatch job1
Submitted batch job 5547

cori06% sbatch --dependency=afterok:5547 job2
or
cori06% sbatch --dependency=afterany:5547 job2
```

https://docs.nersc.gov/jobs/examples/#dependencies

```
cori06% sbatch job1
submitted batch job 5547

cori06% cat job2
#!/bin/bash -l
#SBATCH -q regular
#SBATCH -N 1
#SBATCH -t 00:30:00
#SBATCH -d afterok:5547
#SBATCH -C haswell
srun -n 16 -c 4 ./a.out

cori06% sbatch job2
```

Variable-time Jobs

```
#!/bin/bash
#SBATCH -q regular
#SBATCH -C haswell
#SBATCH -N 2
#SBATCH --comment=96:00:00
#SBATCH --time-min=2:00:00
#SBATCH --time=48:00:00
#SBATCH --signal=B:USR1@60
#SBATCH –requeue

max_timelimit=48:00:00
ckpt_overhead=60
. /usr/common/software/variable-time-job/setup.sh
srun -n 8 -c 16 --cpu_bind=cores  ../test.exe  &
wait
```

- Allows to run multiple jobs with accumulated run time longer than max allowed wall time
- You may get run time longer than 2 hrs but shorter than 48 hrs at a time in this example.
- Job needs to have checkpoint/restart capability
- Individual jobs will be terminated with signal USR1 before time limit is reached
- Pre-terminated jobs will be requeued

https://docs.nersc.gov/jobs/examples/#variable-time-jobs

Use “flex” QOS to Run Variable-Time

- **For user jobs that can produce useful work with a relatively short amount of run time before terminating, such as jobs capable of checkpointing and restarting where left off.**
- **Helps to improve throughput by submitting jobs that can fit into cracks in Slurm job scheduling**
- **Requires to use “--time-min” of <= 2hrs, max “--time” is 48 hrs**
- **75% charging discount as of June 2019 (subject to change)**
- **Available for KNL only. More info at**
  - https://docs.nersc.gov/jobs/examples/#using-the-flex-qos-for-charging-discount-for-variable-time-jobs-on-knl
  - https://docs.nersc.gov/jobs/policy/#flex

Using Burst Buffer Example

```
#!/bin/bash
#SBATCH –q regular –N 10 –t 00:10:00
#DW jobdw capacity=1000GB access_mode=striped type=scratch
#DW stage_in source=/lustre/inputs destination=$DW_JOB_STRIPED/inputs \ type=directory
#DW stage_in source=/lustre/file.dat destination=$DW_JOB_STRIPED/ type=file
#DW stage_out source=$DW_JOB_STRIPED/outputs destination=/lustre/outputs \ type=directory
srun my.x --indir=$DW_JOB_STRIPED/inputs --infile=$DW_JOB_STRIPED/file.dat \ --
outdir=$DW_JOB_STRIPED/outputs
```

- **'type=scratch' – duration just for compute job (i.e. not 'persistent')**
- **'access_mode=striped' – visible to all compute nodes (i.e. not 'private') and striped across multiple BB nodes**
- **Data 'stage_in' before job start and 'stage_out' after**
- https://docs.nersc.gov/jobs/examples/#burst-buffer

Using Shifter Example

Submit script

```
#!/bin/bash
#SBATCH -N 16 -t 20
#SBATCH --image=scanon/myapp:1.1

module load shifter
export TMPDIR=/mnt
srun -n 16 shifter /myapp/app
```

```
cori> shifterimg pull scanon/myapp:1.1
cori> sbatch ./job.sl
```

https://docs.nersc.gov/programming/shifter/overview/

xfer Jobs

```
#!/bin/bash -l
#SBATCH -M escori
#SBATCH -q xfer
#SBATCH -t 12:00:00
#SBATCH -J my_transfer

#Archive run01 to HPSS
htar -cvf run01.tar run01
```

- Configured for the purpose of staging data from HPSS before run or archive result to HPSS after run
- Avoid large NERSC hours if done within large runs
- Runs on external login nodes, via Slurm Server ”escori”.
- Can submit jobs to the xfer QOS from inside another batch script:
  - Add to the end of batch script: “sbatch -M esedison -q xfer myarchive.sl”
- https://docs.nersc.gov/jobs/examples/#xfer-queue

bigmem Jobs

```
#!/bin/bash -l
#SBATCH -M escori
#SBATCH -q bigmem
#SBATCH -N 1
#SBATCH -t 01:00:00
#SBATCH -J my_big_job
#SBATCH -L SCRATCH
#SBATCH --mem=250GB
srun -N 1 -n 1 my_big_executable
```

- Runs on external login nodes, via Slurm Server "escori"
- https://docs.nersc.gov/jobs/examples/#large-memory

Process/Thread/Memory Affinity

- **Correct process, thread and memory affinity is the basis for getting optimal performance on KNL. It is also essential for guiding further performance optimizations.**
  - Process Affinity: bind MPI tasks to CPUs
  - Thread Affinity: bind threads to CPUs allocated to its MPI process
  - Memory Affinity: allocate memory from specific NUMA domains
- **Our goal is to promote OpenMP4 standard settings for portability. For example, OMP_PROC_BIND and OMP_PLACES are preferred to Intel specific KMP_AFFINITY and KMP_PLACE_THREADS settings.**
- **https://docs.nersc.gov/jobs/affinity/**

“numactl -H” displays NUMA domain info

68-core Quad Cache node:

NUMA Domain 0: all 68 cores (272 logic cores)

```
yunhe@cori01:> salloc -N 1 --qos=interactive -C knl,quad,cache -t 30:00
salloc: Granted job allocation 5291739

yunhe@nid02305:> numactl -H
available: 1 nodes (0)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43
44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87
88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122
123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153
154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184
185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246
247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271
node 0 size: 96762 MB
node 0 free: 93067 MB
node distances:
node 0
 0: 10
```

- The quad,cache mode has only 1 NUMA node with all CPUs on the NUMA node 0 (DDR memory)
- The MCDRAM is hidden from the numactl -H command (it is a cache).

Can We Just Do a Naïve Srun?

Example: 16 MPI tasks x 8 OpenMP threads per task on a single 68-core KNL quad,cache node:

```
% export OMP_NUM_THREADS=8
% export OMP_PROC_BIND=spread   (other choice are "close","master","true","false")
% export OMP_PLACES=threads      (other choices are: cores, sockets, and various
ways to specify explicit lists, etc.)

% srun -n 16  ./xthi |sort -k4n,6n
Hello from rank 0, thread 0, on nid02304. (core affinity = 0)
Hello from rank 0, thread 1, on nid02304. (core affinity = 144)     (on physical core 8)
Hello from rank 0, thread 2, on nid02304. (core affinity = 17)
Hello from rank 0, thread 3, on nid02304. (core affinity = 161)     (on physical core 25)
Hello from rank 0, thread 4, on nid02304. (core affinity = 34)
Hello from rank 0, thread 5, on nid02304. (core affinity = 178)     (on physical core 42)
Hello from rank 0, thread 6, on nid02304. (core affinity = 51)
Hello from rank 0, thread 7, on nid02304. (core affinity = 195)     (on physical core 59)
Hello from rank 1, thread 0, on nid02304. (core affinity = 0)
Hello from rank 1, thread 1, on nid02304. (core affinity = 144)
```

It is a mess!

Importance of -c and --cpu_bind Options

- **The reason: 68 is not divisible by #MPI tasks!**
  - Each MPI task is getting 68x4/#MPI tasks of logical cores as the domain size
  - MPI tasks are crossing tile boundaries
- **Let's set number of logical cores per MPI task (-c) manually by wasting extra 4 cores on purpose, which is 256/#MPI tasks.**
  - Meaning to use 64 cores only on the 68-core KNL node, and spread the logical cores allocated to each MPI task evenly among these 64 cores.
  - Now it looks good!
  - % srun -n 16 -c 16 --cpu_bind=cores ./xthi

```
Hello from rank 0, thread 0, on nid09244. (core affinity = 0)
Hello from rank 0, thread 1, on nid09244. (core affinity = 136)
Hello from rank 0, thread 2, on nid09244. (core affinity = 1)
Hello from rank 0, thread 3, on nid09244. (core affinity = 137)
```

Now It Looks Good!

Process/thread affinity are good! (Marked first 6 and last MPI tasks only)

- **MPI rank 0**
- **MPI rank 1**
- **MPI rank 2**
- **MPI rank 3**
- **MPI rank 4**
- **MPI rank 5**
- **....**
- **MPI rank 15**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85
136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153
204																220	221
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
86																102	103
154	155	156	157	158	159	160	161									170	171
222																238	239
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51		
104																	
172																	
240																	
52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67		
120																	
188								196	197	198	199						
256																	

And so on for other MPI tasks and threads ….

Essential Runtime Settings for Process/Thread Affinity

- **Use srun -c and --cpu_bind flags to bind tasks to CPUs**
  - **-c <n> (or --cpus-per-task=n) allocates (reserves) n CPUs per task (process). It helps to evenly spread MPI tasks**
  - Use **--cpu_bind=cores** (no hyperthreads) or **--cpu_bind=threads** (if hyperthreads are used)
- **Use OpenMP envs, OMP_PROC_BIND and OMP_PLACES to fine pin each thread to a subset of CPUs allocated to the host task**
  - Different compilers may have different implementations
  - The following provide compatible thread affinity among Intel, GNU and Cray compilers:

OMP_PROC_BIND=true # Specifying threads may not be moved between CPUs

OMP_PLACES=threads # Specifying a thread should be placed on a single CPU

Sample Job Scripts to Run on Cori KNL

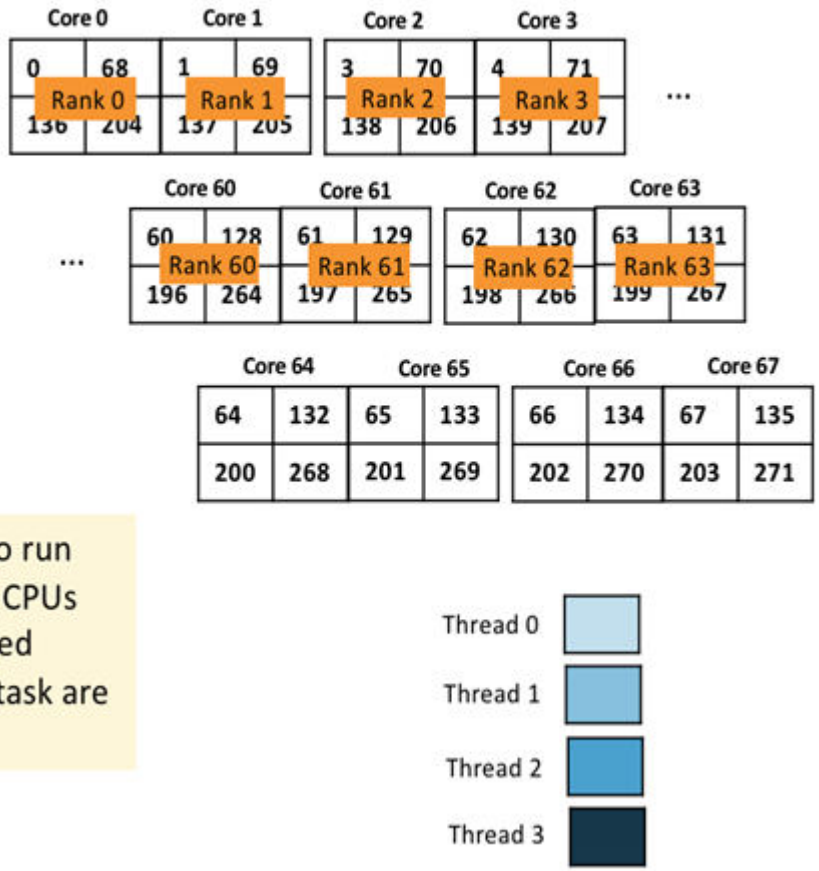
Sample Job script (MPI+OpenMP)

```
#!/bin/bash -l
#SBATCH -N 2
#SBATCH -q regular
#SBATCH -t 1:00:00
#SBATCH -L SCRATCH
#SBATCH -C knl,quad,cache

export OMP_NUM_THREADS=4
srun -n 128 -c 4 --cpu_bind=cores ./a.out
```

This job script requests 2 KNL quad, cache nodes to run 128 MPI tasks, 64 MPI tasks per node, allocating 4 CPUs per task, and binds each task to the 4 CPUs allocated within each core. The 4 OpenMP threads per MPI task are bound to the 4 CPUs in the core.

Process and thread affinity



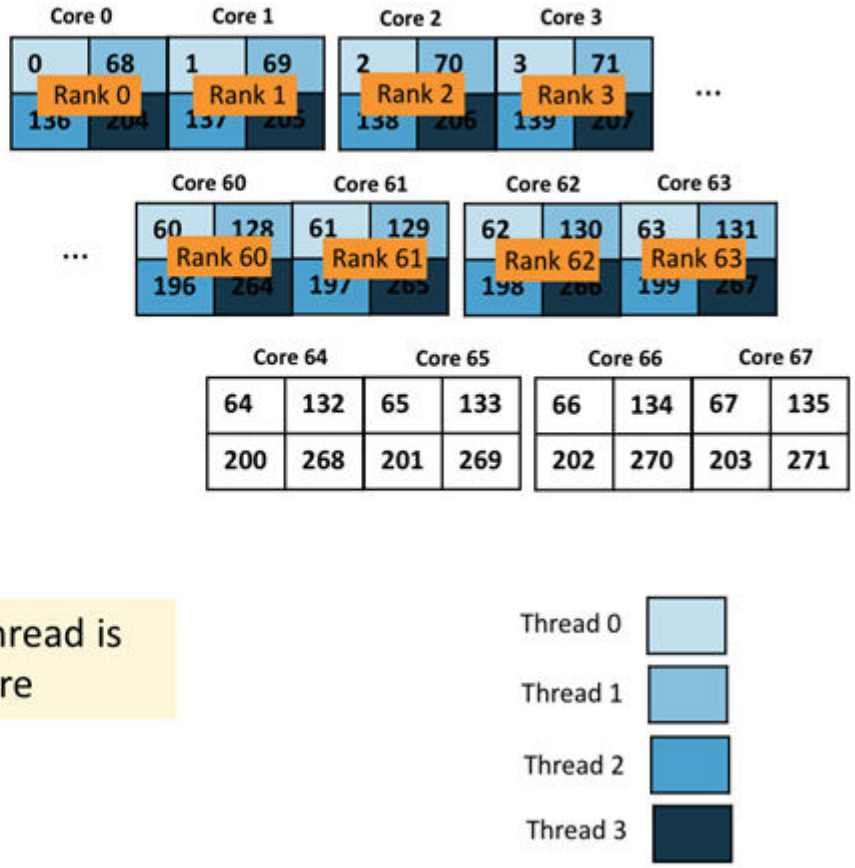
Sample Job Scripts to Run on Cori KNL

Sample Job script (MPI+OpenMP)

```
#!/bin/bash -l
#SBATCH -N 2
#SBATCH -q regular
#SBATCH -t 1:00:00
#SBATCH -L SCRATCH
#SBATCH -C knl,quad,cache

export OMP_PROC_BIND=true
export OMP_PLACES=threads
export OMP_NUM_THREADS=4
srun -n 128 -c 4 --cpu_bind=cores ./a.out
```

Process and thread affinity



With the above two OpenMP envs, each thread is now pinned to a single CPU within each core

Affinity Verification Methods

- **NERSC has provided pre-built binaries from a Cray code (xthi.c) to display process thread affinity: check-mpi.intel.cori, check-mpi.cray.cori, check-hybrid.intel.cori, etc.**

```
% srun -n 32 -c 8 --cpu_bind=cores check-mpi.intel.cori|sort -nk 4
Hello from rank 0, on nid02305. (core affinity = 0,1,68,69,136,137,204,205)
Hello from rank 1, on nid02305. (core affinity = 2,3,70,71,138,139,206,207)
```

- **OpenMP 5.0 has OMP_DISPLAY_AFFINITY and OMP_AFFINITY_FORMAT**
  - Available in Intel compiler >= 18.0.5, gcc >= 9.0, and CCE >= 9.0.0

```
% export OMP_DISPLAY_AFFINITY=true
% export OMP_AFFINITY_FORMAT="host=%H, pid=%P, thread_num=%n, thread affinity=%A"
host=nid02496, pid=150147, thread_num=0, thread affinity=0
host=nid02496, pid=150147, thread_num=1, thread affinity=4
```

NERSC Job Script Generator

https://my.nersc.gov/script_generator.php

- yunhe
- Dashboard
- Jobs
  - Jobscript Generator
  - Completed Jobs
  - Cori Queues
  - Edison Queues
  - PDSF Queues
  - Queue Backlog
  - Job Completion Stats
- Center Status
- File Browser
- My Tickets
- Data Dashboard

Jobscript Generator

Job Information

This tool generates a batch script template which also realizes specific process and thread binding configurations.

Machine

Select the machine on which you want to submit your job.

Cori - KNL

Application Name

Specify your application including the full path.

myapp.x

Job Name

Specify a name for your job.

mytest_KNL

Email Address

Specify your email address to get notified when the job enters a certain state.

```
#!/bin/bash
#SBATCH -N 150
#SBATCH -C knl
#SBATCH -q regular
#SBATCH -J mytest_KNL
#SBATCH -t 02:30:00

#OpenMP settings:
export OMP_NUM_THREADS=8
export OMP_PLACES=threads
export OMP_PROC_BIND=spread


#run the application:
srun -n 1200 -c 32 --cpu_bind=cores myapp.x
```

Monitoring Your Job

- **Once your job is submitted, it enters the queue and will start when resources are available**
- **Overall job priorities are a combination of QOS, queue wait time, job size, wall time request (and fair share).**
- **You can monitor it with:**
  - sqs
  - squeue

  On the web:

  https://my.nersc.gov

  https://www.nersc.gov/users/live-status/ : "Queue Look"

  https://www.nersc.gov/users/job-logs-and-analytics/completed-jobs/

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
- **sqs**: NERSC custom queue display with estimated job start time info
- **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
- https://docs.nersc.gov/jobs/#submitting-jobs
- https://docs.nersc.gov/jobs/#monitoring-jobs

sqs: NERSC Custom Batch Queue Script

```
yunhe@cori05:~> sqs -u fxxxx | more
JOBID     ST  USER    NAME      NODES REQUESTED USED  SUBMIT              QOS        SCHEDULED_START      FEATURES REASON
110901xx  PD  fxxxx   mxxx      1536  5:00      0:00  2018-03-20T10:49:23 regular_0  2018-03-22T06:30:00  haswell  Resources
110901xx  PD  fxxxx   run.xxx*  1537  20:00     0:00  2018-03-20T10:51:03 regular_0  2018-03-22T06:30:00  haswell  Resources
110823xx  PD  fxxxx   gxxx      300   30:00     0:00  2018-03-19T23:05:24 regular_1  avail_in_~1.6_days   haswell  Priority
110823xx  PD  fxxxx   run-xx    768   20:00     0:00  2018-03-19T23:05:33 regular_1  avail_in_~1.6_days   haswell  Priority
110901xx  PD  fxxxx   Jxxx      768   10:00     0:00  2018-03-20T10:51:33 regular_1  avail_in_~2.1_days   haswell  Priority
110921xx  PD  fxxxx   pxxxxx-xx* 150  30:00     0:00  2018-03-20T12:42:38 regular_1  avail_in_~2.1_days   haswell  Priority
111046xx  PD  fxxxx   hxxxxx    32    20:00     0:00  2018-03-21T01:20:09 regular_1  avail_in_~2.7_days   haswell  Priority
110823xx  PD  fxxxx   rxxxx     1536  20:00     0:00  2018-03-19T23:05:04 regular_0  N/A                  haswell  JobHeldUser
110823xx  PD  fxxxx   axxxxxxxx* 1536 30:00     0:00  2018-03-19T23:05:16 regular_0  N/A                  haswell  JobHeldUser
111152xx  PD  fxxxx   run.xxx    769 2:00:00    0:00  2018-03-21T09:39:29 regular_1  avail_in_~3.0_days   knl&quad&cache  None
<omitted...>

==========================================================================================
Note: The SCHEDULED_START column shows either the estimated start time provided by Slurm,
    or that an estimated start time will be "available in xxx hrs/days", meaning your job
    won't be scheduled by then (due to priority) unless via backfill, and the actual job
    start time could still be some period of time after then.

Second Note: Upcoming system maintenance on 03-22-2018 07:00 AM, SCHEDULED_START will be
        updated after the maintenance is completed.
==========================================================================================
```

- sqs combines information from squeue and sinfo, provides formatted output with various options
- The scheduled_start gives users an estimate for when their jobs will start (or when will be considered for scheduling based on distance to priority threshold).

sqs Example Commands

It has many options, such as no running jobs, yes or no for a specific qos, wider with more fields, compact format, custom fields, expand array jobs, csv and json output, etc.

See "sqs --help" or "man sqs" for more details.

% sqs (show user's own jobs)

% sqs -a (shows all jobs)

% sqs -a -q debug (shows only debug jobs)

% sqs -a -nr -nq shared (no running jobs, no shared jobs)

% sqs -w (shows all my jobs in wide format with more info)

% sqs -s (short summary of queued jobs)

scontrol: Show Job Details

- **% scontrol show job <jobid> for details of a job**

```
yunhe@cori01:~> scontrol show job 11082385
JobId=11082385 JobName=gtc19200
   UserId=fbench(42034) GroupId=fbench(42034) MCS_label=N/A
   Priority=66669 Nice=0 Account=mpcray QOS=regular_1
   JobState=PENDING Reason=Priority Dependency=(null)
   Requeue=0 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:00:00 TimeLimit=00:30:00 TimeMin=N/A
   SubmitTime=2018-03-19T23:05:24 EligibleTime=2018-03-19T23:05:24
   StartTime=Unknown EndTime=Unknown Deadline=N/A
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   LastSchedEval=2018-03-21T06:18:17
   Partition=regular AllocNode:Sid=cori01:159414
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=(null)
   NumNodes=300-300 NumCPUs=300 NumTasks=300 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=300,node=300
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryNode=0 MinTmpDiskNode=0
   Features=haswell DelayBoot=00:00:00
   Gres=craynetwork:1 Reservation=(null)
   OverSubscribe=NO Contiguous=0 Licenses=cscratch1:1 Network=(null)
   Command=/global/cscratch1/sd/fbench/GTC/run/run19200opt
   WorkDir=/global/cscratch1/sd/fbench/GTC/run
   StdErr=/global/cscratch1/sd/fbench/GTC/run/slurm-11082385.out
   StdIn=/dev/null
   StdOut=/global/cscratch1/sd/fbench/GTC/run/slurm-11082385.out
   Power=
```

sacct: Query Completed and Pending Jobs Information

```
yunhe@cori05:~> sacct -u fbench -S 2018-03-19 -E 2018-03-20 -o
user,jobid,start,end,elapsed,timelimit,nnodes,exitcode,state -X
    User       JobID           Start             End    Elapsed  Timelimit   NNodes ExitCode      State
--------- ------------ ------------------- ------------------- ---------- ---------- -------- -------- ----------
  fbench 10971468    2018-03-19T10:51:07 2018-03-19T10:56:30  00:05:23  00:30:00      32      0:0  COMPLETED
  fbench 11006329    2018-03-19T14:06:28 2018-03-19T14:06:56  00:00:28  00:30:00     150      0:0  COMPLETED
  fbench 11038987    2018-03-19T23:19:34 2018-03-19T23:24:54  00:05:20  00:30:00      32      0:0  COMPLETED
  fbench 11062282    2018-03-19T03:35:01 2018-03-19T03:37:56  00:02:55  00:30:00       1      0:0  COMPLETED
  fbench 11076856    2018-03-19T16:50:30 2018-03-19T16:53:10  00:02:40  00:30:00       3      0:0  COMPLETED
  fbench 11076857    2018-03-19T16:50:30 2018-03-19T17:04:48  00:14:18  00:30:00       3      1:0     FAILED
   fbench 11082385             Unknown             Unknown  00:00:00  00:30:00     300      0:0    PENDING
  fbench 11082394             Unknown             Unknown  00:00:00  00:20:00     768      0:0    PENDING
  fbench 11082403    2018-03-20T00:19:32 2018-03-20T00:32:33  00:13:01  00:30:00      63      0:0  COMPLETED
   ...
```

- **Maximum query duration is one month (subject to change)**
- **Detailed job steps info will be displayed without "-X" flag**
- **Many more job fields can be queried. See "man sacct"**

Where to Run My Jobs

- **Queue configuration and policies are subject to further tuning for max throughput and system utilization**
- **Factors to consider: queue wait time (much shorter on KNL than on Haswell), throughput, charging, code readiness on KNL**
  - Cori Haswell (also known as the "Cori Data Partition") system is designed to accelerate data-intensive applications; 2388 total compute nodes
  - Cori KNL: Large capability and performance; 9688 total compute nodes
  - Jobs use 1024+ nodes on Cori KNL get 20% charging discount
  - “flex” and “low” qos with discounts only available on Cori KNL
  - “shared” and ”realtime” available on Cori Haswell only
  - “interactive” nodes available on Cori Haswell and KNL; 192 nodes each
  - “bigmem” and “xfer” available on Cori (run on external login nodes)
- **Remember to compile separately for each type of compute nodes**

Cori Haswell Queue Policy (as of June 2019)

QOS	Max nodes	Max time (hrs)	Submit limit	Run limit	Priority	Charge
regular	1932	48	5000	-	4	90
shared [1]	0.5	48	10000	-	4	90
interactive [4]	64	4	2	2	-	90
debug	64	0.5	5	2	3	90
premium	1772	48	5	-	2	180
scavenger [2]	1772	48	5000	-	5	0
xfer	1 (login)	48	100	15	-	0
bigmem	1 (login)	72	100	1	-	0
realtime [3]	custom	custom	custom	custom	1	custom
special [5]	custom	custom	custom	custom	-	custom

Cori KNL Queue Policy (as of June 2019)

QOS	Max nodes	Max time (hrs)	Submit limit	Run limit	Priority	Charge
regular	9489	48	5000	-	4	90 [6]
interactive [4]	64	4	2	2	-	90
debug	512	0.5	5	2	3	90
premium	9489	48	5	-	2	180 [6]
low	9489	48	5000	-	5	67.5 [7]
flex	256	48	5000	-	6	22.5 [8]
scavenger [2]	9489	48	5000	-	7	0
special [5]	custom	custom	custom	custom	-	custom

Tips for Getting Better Throughput

- **Line jumping is allowed, but it may cost more (with "premium" QOS)**
- **Submit shorter jobs, they are easier to schedule**
  - Checkpoint to break up long jobs, use variable time
  - Short jobs can take advantage of 'backfill' opportunities
  - Run short jobs just before maintenance
  - Run variable-time jobs; use "flex" QOS
- **Make sure the wall clock time you request is accurate**
  - Larger shorter jobs are easier to schedule than long smaller jobs
  - Many users unnecessarily enter the largest wall clock time possible as default
- **Queue wait time statistics**
  - https://my.nersc.gov/queuewaittimes.php
  - https://www.nersc.gov/users/queues/queue-wait-times/

Outline

- Systems Overview
- Connecting to NERSC
- File Systems and Data Management/Transfer
- Data Analytics Software and Services
- Software Environment / Building Applications
- Running Jobs
- **NERSC Online Resources**
- Hands-on: Compiling and Running Jobs

Online Resources: www.nersc.gov

- **NIM:** https://nim.nersc.gov
- **Contact Us:** https://www.nersc.gov/about/contact-us/
- **MOTD:** https://www.nersc.gov/live-status/motd/

- https://www.nersc.gov/users/ **and** https://docs.nersc.gov
  - Connecting to NERSC
  - Computational-Systems
  - Programming
  - Running Jobs
  - Job Logs and Statistics
  - Storage and File Systems
  - Data, Analytics & Services
  - Training and Tutorials
  - Software

Online Resources: my.nersc.gov

- **Dashboard /Data Dashboard**
- **Jobs**
  - Job Script Generator
  - Completed Jobs
  - Queues
  - Queue Backlogs
- **Center Status**
  - Announcements / Outage Log
  - Filesystems Monitor
  - Job Size Charts
  - Queue Wait Times
  - Application Usage
- **My Tickets**
- **File Browser**
- **NX Desktop**
- **Jupyter Hub**

Online Resources: NIM and Help Portal

- **NIM: NERSC Information Management: https://nim.nersc.gov**
  - Change password
  - Change contact info
  - SSH Keys
  - MFA
  - Change default shell
  - Check usage information
- **Help Portal: https://help.nersc.gov**
  - Submit tickets (ask questions)
  - Request forms:
    - Quota Increase
    - Reservations
    - Allocation request

Contact NERSC

Technical Questions, Computer Operations, Passwords, Account Support
1-800-666-3772 (or 1-510-486-8600)

Computer Operations

menu option 1
(for off-hours problem reports or inquiries)
Hours: 24/7, 365 days a year

Account Support

menu option 2 or
accounts@nersc.gov or

http://nim.nersc.gov
Hours: Weekdays 8-5 Pacific

HPC Consulting

menu option 3 or
consult@nersc.gov
Hours: Weekdays 8-5 Pacific
or
http://help.nersc.gov (On-line Help Desk)
or
https://my.nersc.gov/tickets.php

Outline

- Systems Overview
- Connecting to NERSC
- File Systems and Data Management/Transfer
- Data Analytics Software and Services
- Software Environment / Building Applications
- Running Jobs
- NERSC Online Resources
- **Hands-on: Compiling and Running Jobs**

Hands-on Exercises

- **% cd $SCRATCH**
- **% cp -r /project/projectdirs/training/2019_CSSS .**
- **Beginner users follow: run-hello.README**
- **Advanced users follow: run-xthi.README**
- **References**
  - Running Jobs
    - https://docs.nersc.gov/jobs/
  - Interactive Jobs:
    - https://docs.nersc.gov/jobs/examples/#interactive

Acknowledgement

- **Used / Adapted some slides from NERSC New user training on June 21, 2019**
  - Thanks Rebecca Hartman-Baker, Steve Leak, Zhengji Zhao, Prabhat, Shreyas Cholia, Rollin Thomas
  - Slides: https://www.nersc.gov/users/training/events/new-user-training-june-21-2019/
    - Links to the YouTube videos will be available soon

Thank you.