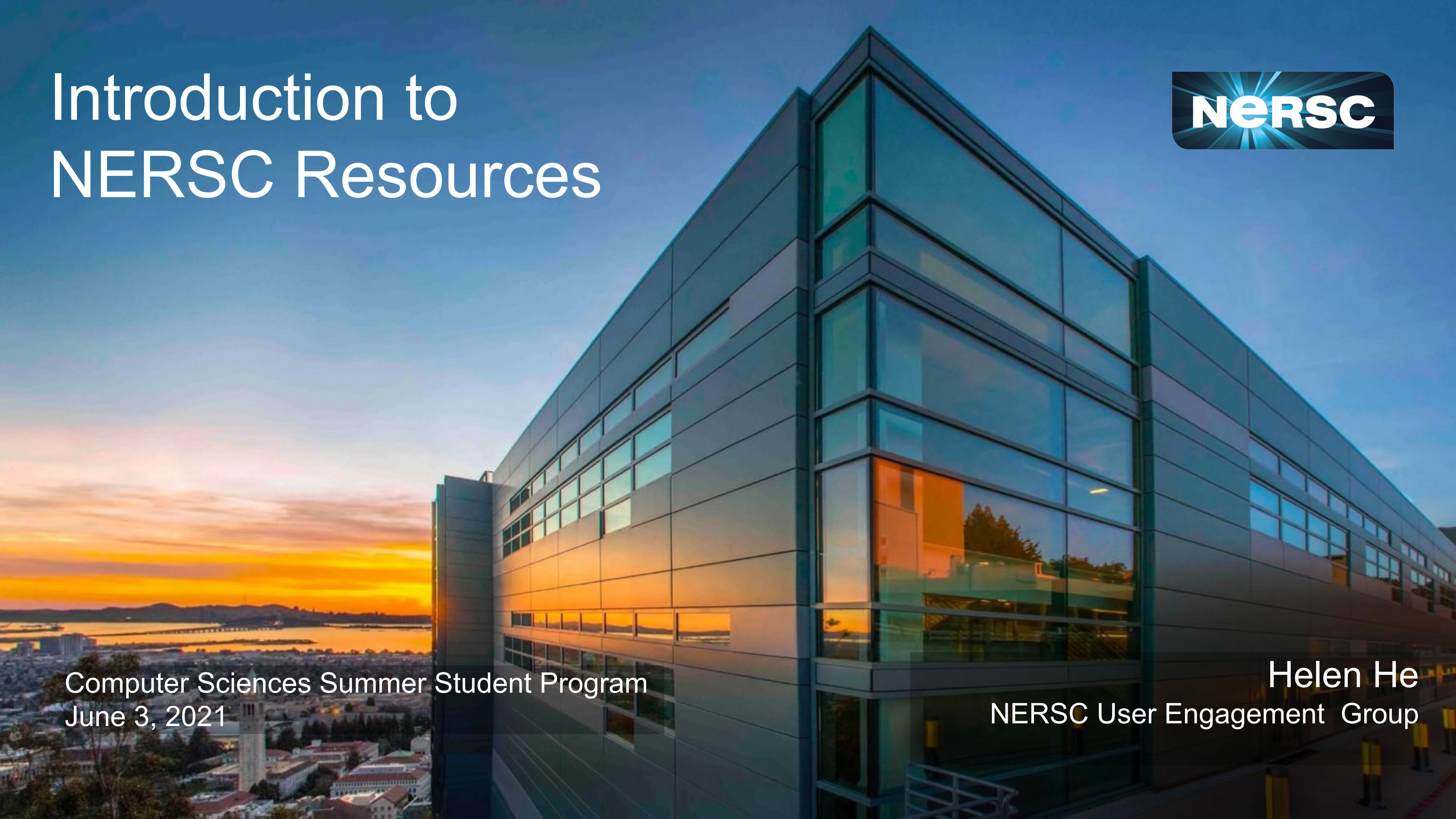


Introduction to NERSC Resources

Computer Sciences Summer Student Program
June 3, 2021

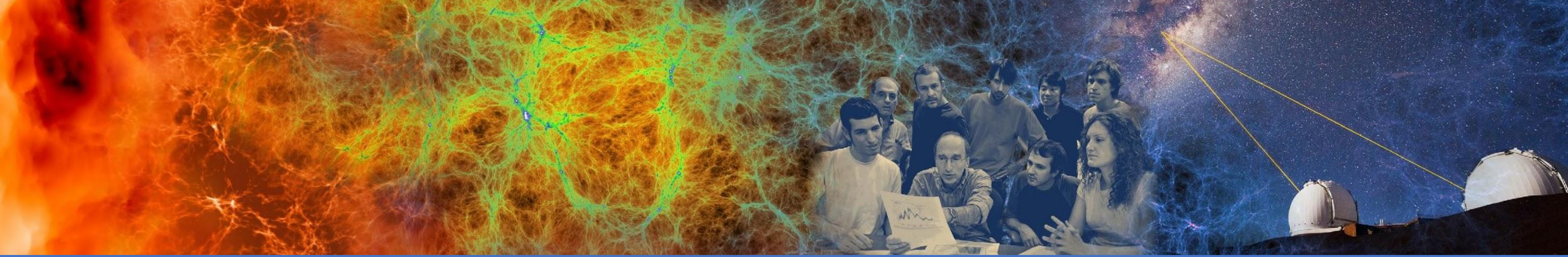
Helen He
NERSC User Engagement Group

Some Logistics

- Users are muted upon joining Zoom (can unmute to speak)
- Please change your name in Zoom session
    - to: first_name last_name
    - Click “Participants”, then “More” next to your name to rename
- Click the CC button to toggle captions and View Full Transcript
- GDoc is used for Q&A (instead of Zoom chat)
    - https://tinyurl.com/QA-intro-nersc-resources
- Slides and videos will be available on the Training Event page
    - https://www.nersc.gov/users/training/events/nersc-resources-june-2021/
- Apply for a training account if no NERSC account yet
    - https://iris.nersc.gov/train, and use the 4-letter code "aMAa"

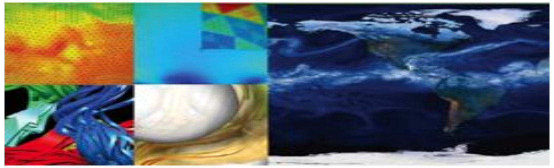
Outline

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



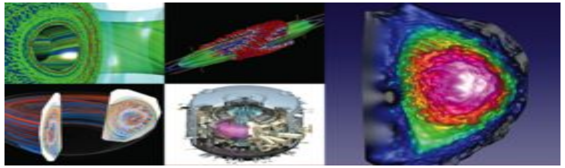
NERSC and Systems Overview

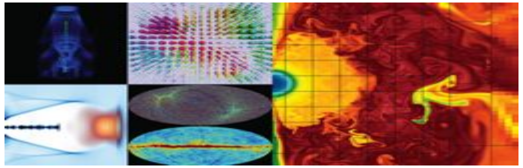
NERSC is the Mission HPC Computing Center for the DOE Office of Science



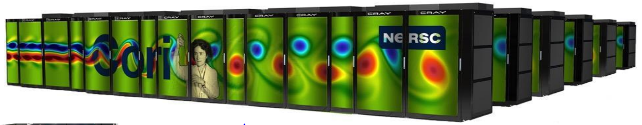
- NERSC deploys advanced HPC and data systems for the broad Office of Science community
- NERSC staff provide advanced application and system performance expertise to users
- Approximately 7,000 users and 800 projects
- Over 2,000 publications cite using NERSC resources per year
- Founded in 1974, focused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

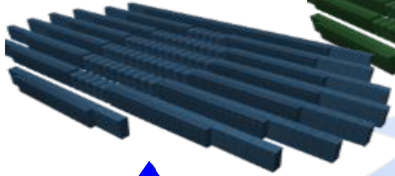
ASCR	Advanced Scientific Computing Research
BER	Biological & Environmental Research
BES	Basic Energy Sciences
FES	Fusion Energy Sciences
HEP	High Energy Physics
NP	Nuclear Physics
SBIR	Small Business Innovation Research

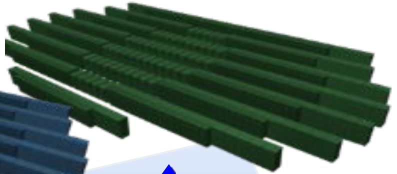




NERSC Systems Roadmap







2013

NERSC-7: Edison Multicore CPU

2016

NERSC-8: Cori Manycore CPU NESAP Launched: transition applications to advanced architectures

2021

NERSC-9: CPU and GPU nodes Continued transition of applications and support for complex workflows

2024

NERSC-10: Exa system

2028

NERSC-11: Beyond Moore

Increasingly energy-efficient architectures

Cori Brings HPC and Data Together

Cori: #20 in Nov 2020 (#5 in Nov 2016) Top 500 list

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

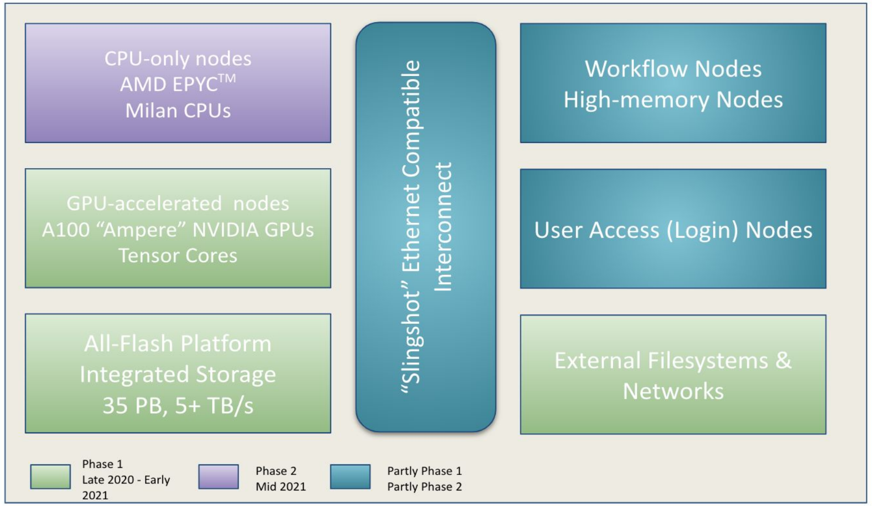
Phase I: 2388 x 32-core Intel Xeon "Haswell" 128 GB DDR4
Also known as "Data Partition" (76,416 cores total)
Phase II: 9688 x 68-core Intel Xeon Phi "KNL" 96 GB DDR4 + 16 GB MCDRAM
(658,784 total cores)

NERSC-9 is named after Saul Perlmutter

- Shared 2011 Nobel Prize in Physics for discovery of the accelerating expansion of the universe.
- Works at LBL, as a NERSC user
- Supernova Cosmology Project, lead by Perlmutter, was a pioneer in using NERSC supercomputers combine large scale simulations with experimental data analysis
- Login “saul.nersc.gov”

First NERSC system designed to meet needs of both large scale simulation and data analysis from experimental facilities

Perlmutter -- an HPE Cray EX System


CPU-only nodes
AMD EPYC™
Milan CPUs
GPU-accelerated nodes
A100 "Ampere" NVIDIA GPUs
Tensor Cores
All-Flash Platform
Integrated Storage
35 PB, 5+ TB/s
"Slingshot" Ethernet Compatible Interconnect
Workflow Nodes
High-memory Nodes
User Access (Login) Nodes
External Filesystems & Networks
Phase 1 Late 2020 - Early 2021
Phase 2 Mid 2021
Partly Phase 1 Partly Phase 2

- Perlmutter dedication was on May 27
- NERSC staff are continuously configuring the Phase 1 system
- Users will be enabled in multiple phases

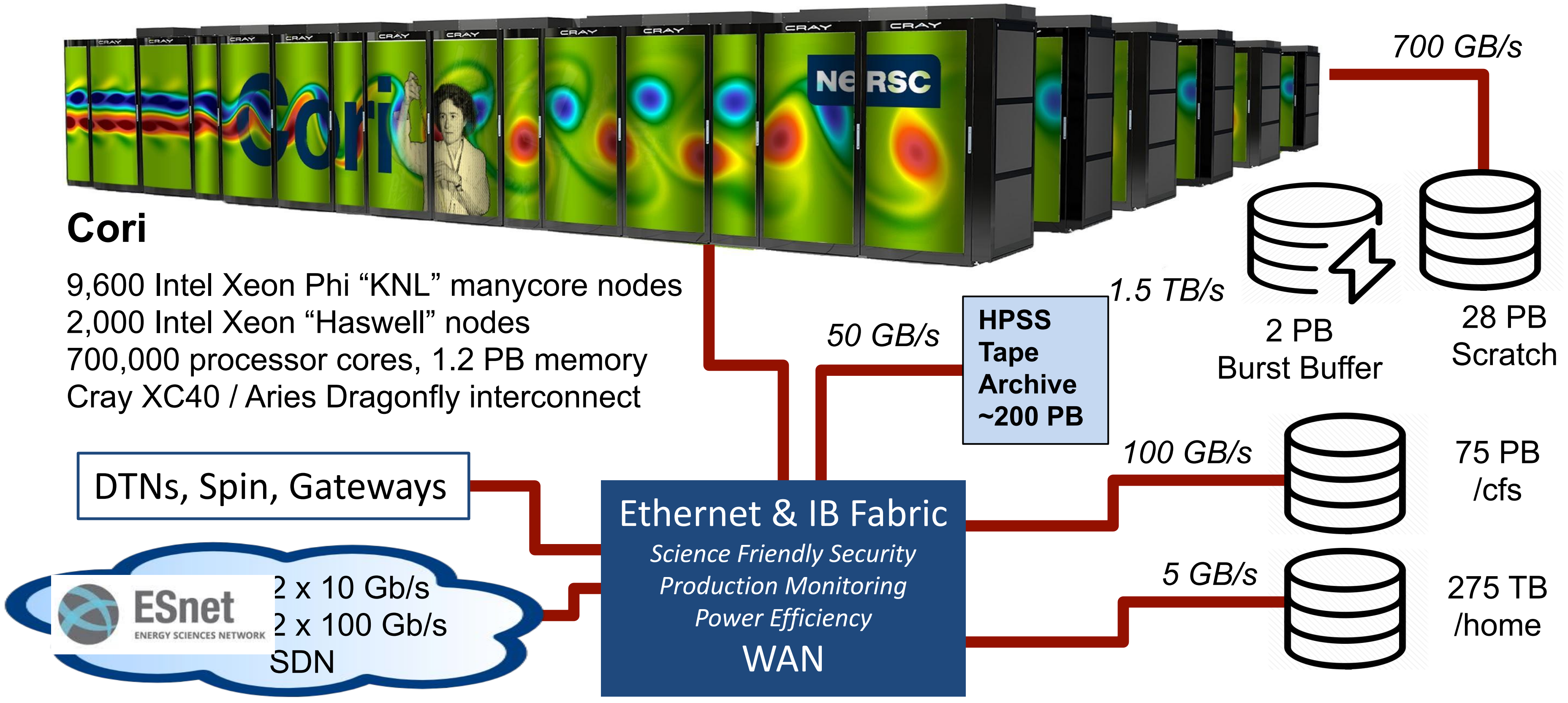
Phase I: Arrived, Nov 2020 -Mar 2021

- 1,536 GPU-accelerated nodes
- 1 AMD "Milan" CPU + 4 NVIDIA A100 GPUs per node
- 256 GB CPU memory and 40 GB GPU high BW memory
- 35 PB FLASH scratch file system
- User access and system management nodes

Phase II Addition: Arrives later 2021

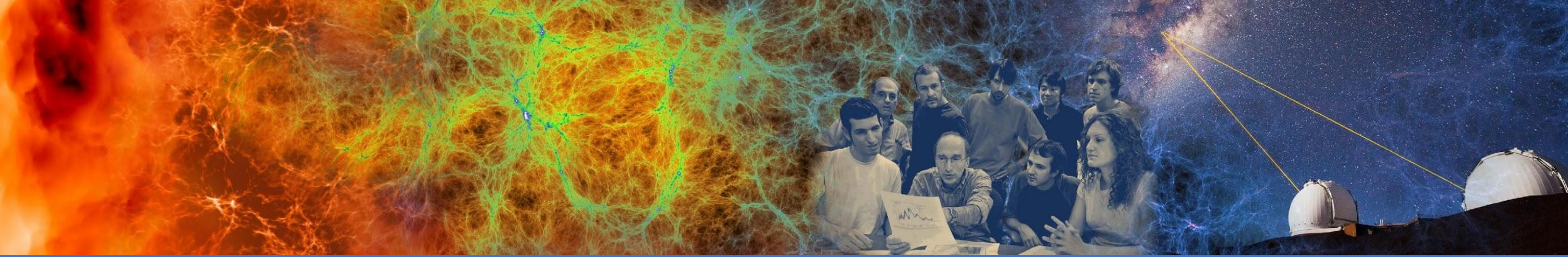
- 3,072 CPU only nodes
- 2 AMD "Milan" CPUs per node
- 512 GB memory per node
- Upgraded high speed network
- CPU partition will match or exceed performance of entire Cori system

NERSC Systems



Cori

9,600 Intel Xeon Phi “KNL” manycore nodes
2,000 Intel Xeon “Haswell” nodes
700,000 processor cores, 1.2 PB memory
Cray XC40 / Aries Dragonfly interconnect



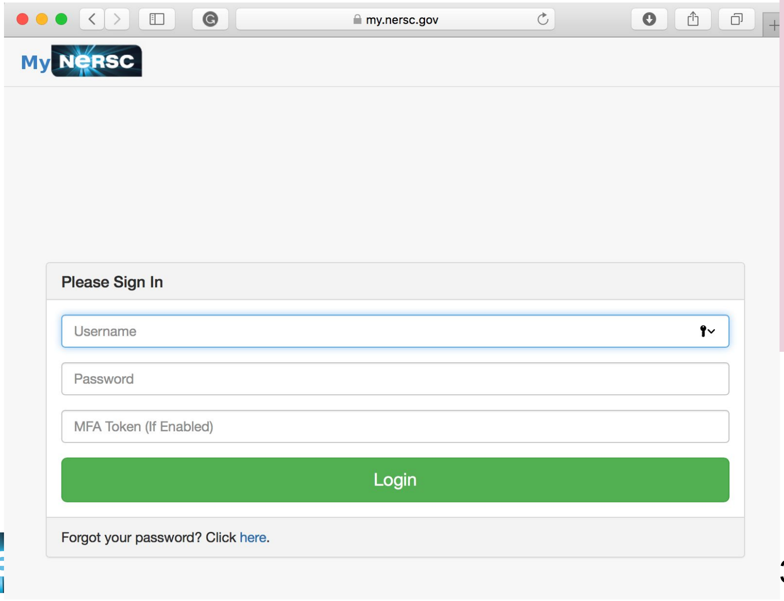
Connecting to NERSC

Multi-Factor Authentication (MFA) and sshproxy

- NERSC 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
  - Setup MFA https://docs.nersc.gov/connect/mfa/
- sshproxy.sh creates a short-term certificate
  - Run sshproxy.sh once, then you can ssh to NERSC systems for the next 24 hours before being asked for password+OTP again
  - https://docs.nersc.gov/connect/mfa/#sshproxy

SSH and MFA Examples

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



Please Sign In

Username

Password

MFA Token (If Enabled)

Login

Forgot your password? Click here.

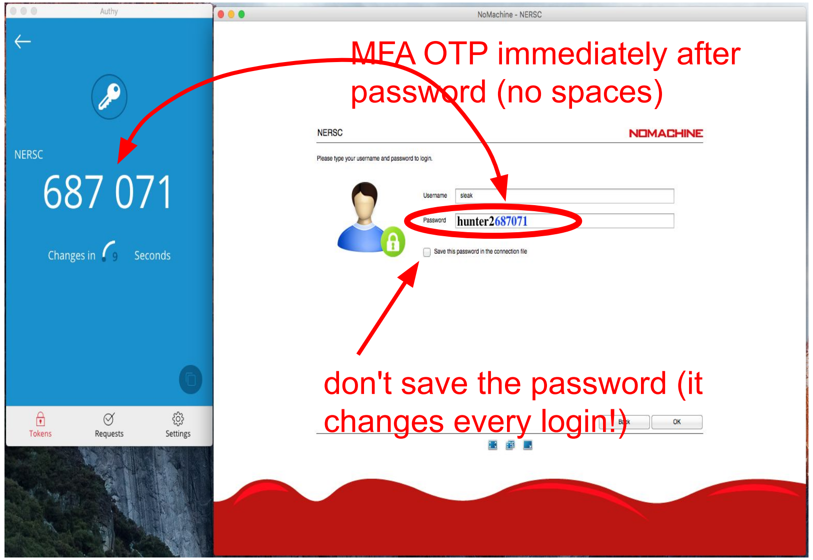
You will login to one of the login nodes (12 on Cori).

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>
```

Connecting to NERSC: NX

- NERSC recommends using NX instead of SSH X-forwarding since NX is faster and more reliable
- NX is a service for Accelerated X
- 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
  - https://docs.nersc.gov/connect/nx
  - Works on Window/Mac/Linux



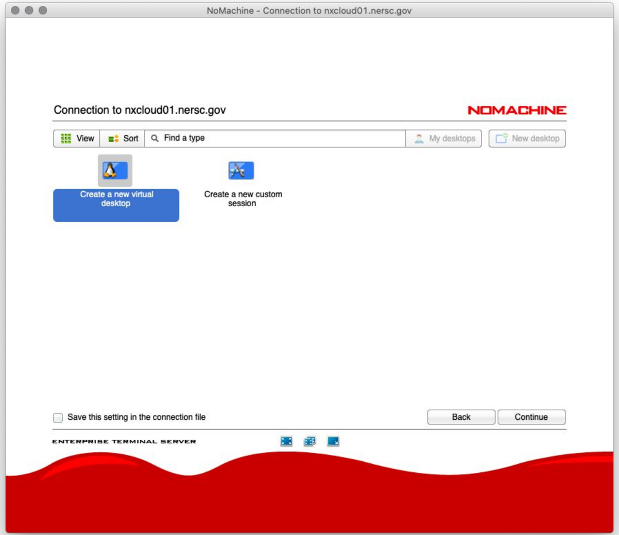
Please type your username and password to login.

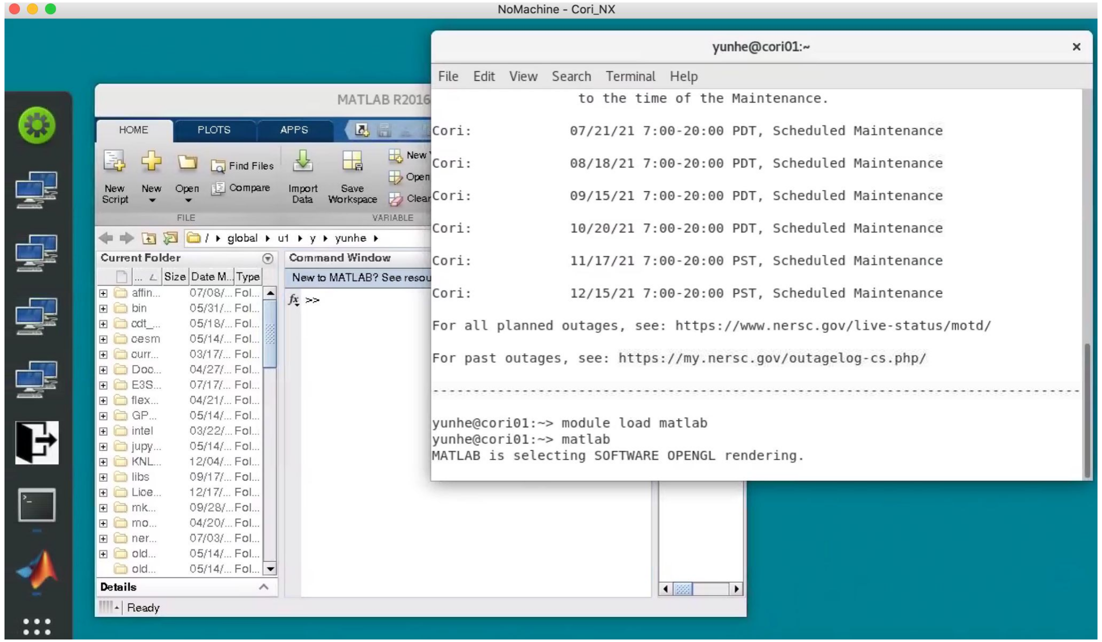
Username: sleak

Password: hunter2687071

[ ] Save this password in the connection file

NoMachine


NoMachine - Connection to nxcloud01.nersc.gov
Connection to nxcloud01.nersc.gov
NOMACHINE
View
Sort
Find a type
My desktops
New desktop
Create a new virtual desktop
Create a new custom session
Save this setting in the connection file
Back
Continue
ENTERPRISE TERMINAL SERVER



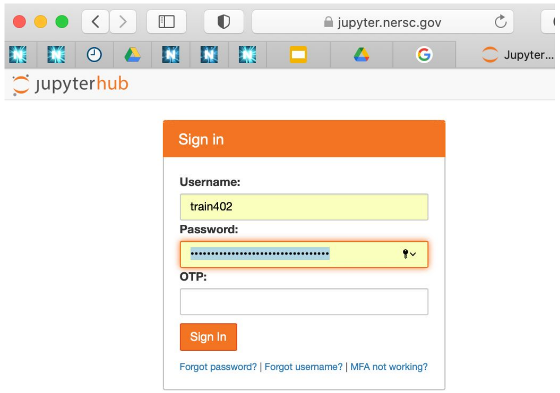
```
                        to the time of the Maintenance.

Cori:           07/21/21 7:00-20:00 PDT, Scheduled Maintenance

Cori:           08/18/21 7:00-20:00 PDT, Scheduled Maintenance

Cori:           09/15/21 7:00-20:00 PDT, Scheduled Maintenance

Cori:           10/20/21 7:00-20:00 PDT, Scheduled Maintenance

Cori:           11/17/21 7:00-20:00 PST, Scheduled Maintenance

Cori:           12/15/21 7:00-20:00 PST, Scheduled Maintenance

For all planned outages, see: https://www.nersc.gov/live-status/motd/

For past outages, see: https://my.nersc.gov/outagelog-cs.php/

--------------------------------------------------------------------------------

yunhe@cori01:~> module load matlab
yunhe@cori01:~> matlab
MATLAB is selecting SOFTWARE OPENGL rendering.
```

BERKELEY LAB
Bringing Science Solutions to the World

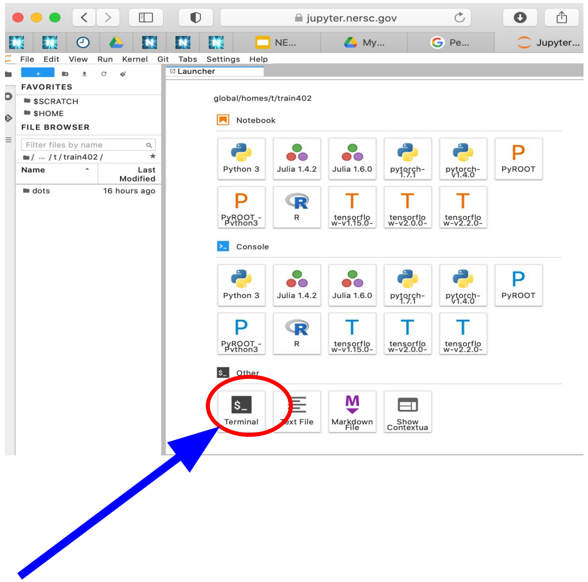
U.S. DEPARTMENT OF ENERGY
Office of Science

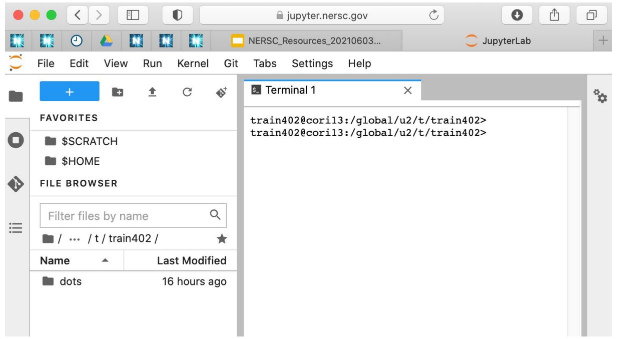
Terminal in Jupyter

You can access Cori from any web browser, via https://jupyter.nersc.gov



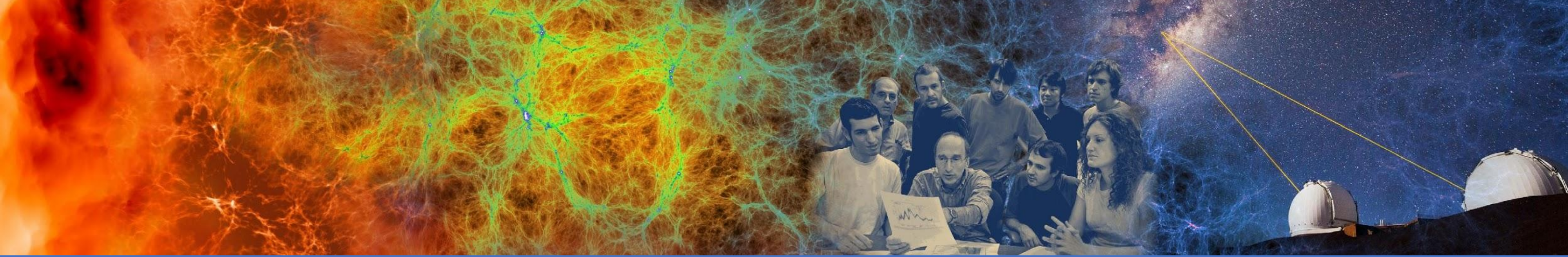
Username: train402
Password:
OTP:





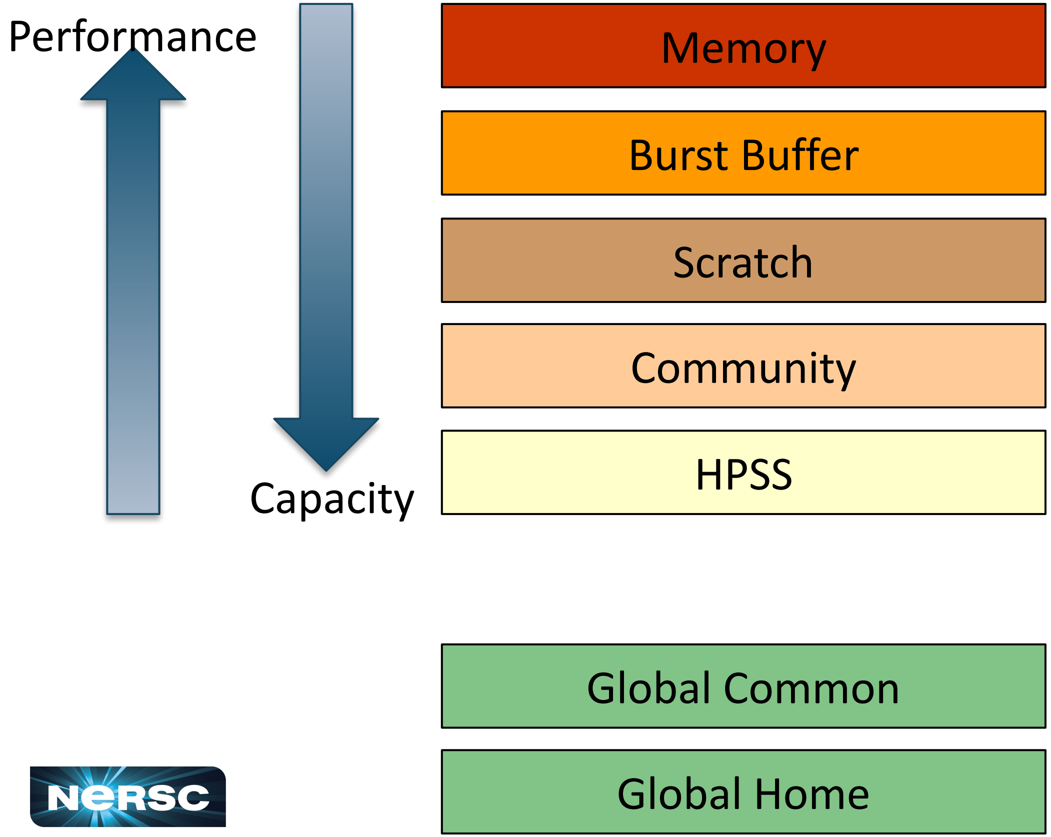
```
train402@cori13:/global/u2/t/train402>
train402@cori13:/global/u2/t/train402>
```

Terminal



File Systems and Data Management / Transfer

Simplified NERSC File Systems



1.8 PB SSD Burst Buffer on Cori
Cray Datawarp 1.8 TB/s,
temporary for job or campaign

28 PB (Cori) HDD Scratch
Lustre 700 GB/s,
temporary (12 wk purge)

157 PB HDD Community
Spectrum Scale (GPFS)
150 GB/s, permanent

150 PB Tape Archive
HPSS Forever

20 TB SSD Software
Spectrum Scale
Permanent
Faster compiling / Source Code

Global File Systems

Global Home

- Permanent, relatively small storage
- Mounted on all platforms
- NOT tuned to perform well for parallel jobs
- Quota cannot be changed
- Snapshot backups (7-day history)
- **Perfect for storing data such as source code, shell scripts**

Community File System (CFS)

- 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

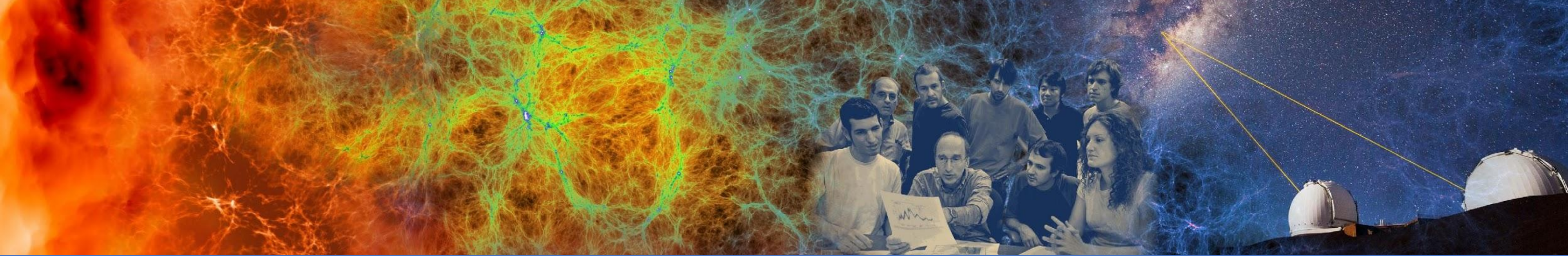
- Large, temporary storage
- Optimized for read/write operations, NOT storage
- Not backed up
- Purge policy (12 weeks)
- **Perfect for staging data and performing computations**

Burst Buffer

- Temporary storage
- 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
- https://docs.nersc.gov/filesystems/archive/



Software Environment and Building Applications

Software

- Cray supercomputers OS is a version of Linux
- Compilers are provided on machines
- Libraries: many libraries provided by vendor and by NERSC
- Applications: NERSC compiles and supports many software packages (such as chemistry and materials sciences packages) for our users
- DOE Extreme-scale Scientific Software Stack (E4S): open-source projects, including xSDK, dev-tools, math-libraries, compilers, and more
-
-

Modules Environment

- Modules are used to manage the user environment
  - https://docs.nersc.gov/environment/#nersc-modules-environment

`module`	
`list`	To list the modules in your environment

`avail`
 	
`avail -S`
	To list available modules
	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`

Default Loaded Modules

```
yunhe@cori03:~> module list
Currently Loaded Modulefiles:
  1) modules/3.2.11.4                                    13)
gni-headers/5.0.12.0-7.0.1.1_6.27__g3b1768f.ari
  2) nsg/1.2.0                                           14) xpmem/2.2.20-7.0.1.1_4.8__g0475745.ari
  3) altd/2.0                                            15) job/2.2.4-7.0.1.1_3.34__g36b56f4.ari
  4) darshan/3.1.7                                       16) dvs/2.12_2.2.156-7.0.1.1_8.6__g5aab709e
  5) intel/19.0.3.199                                    17) alps/6.6.57-7.0.1.1_5.10__g1b735148.ari
  6) craype-network-aries                                18) rca/2.2.20-7.0.1.1_4.42__g8e3fb5b.ari
  7) craype/2.6.2                                        19) atp/2.1.3
  8) cray-libsci/19.06.1                                 20) PrgEnv-intel/6.0.5
  9) udreg/2.3.2-7.0.1.1_3.29__g8175d3d.ari              21) craype-haswell
 10) ugni/6.0.14.0-7.0.1.1_7.32__ge78e5b0.ari            22) cray-mpich/7.7.10
 11) pmi/5.0.14                                          23) craype-hugepages2M
 12) dmapp/7.1.1-7.0.1.1_4.43__g38cf134.ari
```

5) Compiler   8) Cray Scientific Libraries

20) Programing Environment 21) Target architecture Driver 22) 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
- Recommends to build separate binaries for each architecture to take advantage of optimizations unique to processor type

Software Environment

- Available compilers: Intel, GNU, Cray
- Use compiler wrappers to build. It calls native compilers for each compiler (such as ifort, mpiicc, etc.) underneath.
    - Do not use native compilers directly.
    - 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 dynamically by default. Can add “-static” to build statically 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
- Best recommendation to build for KNL target
    - module swap craype-haswell craype-mic-knl
    - The above sets CRAY_CPU_TARGET to mic-knl

Building Simple Test Program (1)

- To build on Cori Haswell:
  - Using default Intel compiler:

    ```
    ftn -o mytest mytest_code.F90
    ```
  - Using Cray compiler:

    ```
    module swap PrgEnv-intel PrgEnv-cray
    ftn -o mytest mytest_code.F90
    ```

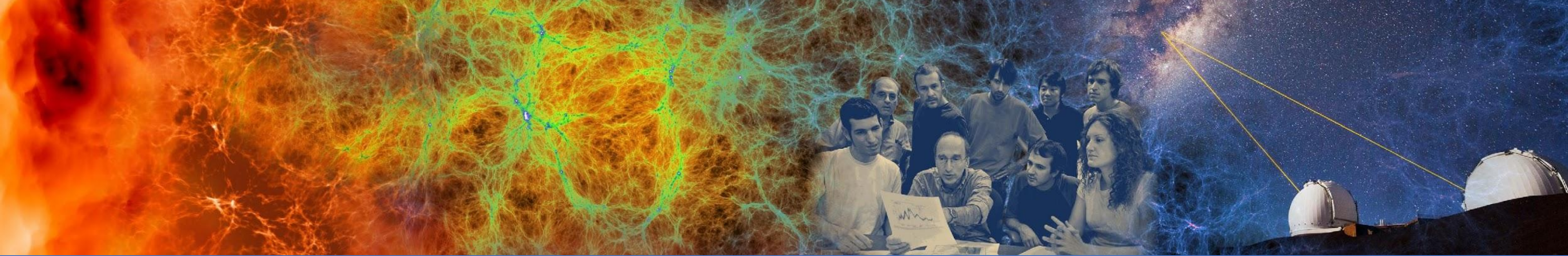
Building Simple Test Program (2)

- To build on Cori KNL
  - Using default Intel compiler

```
module swap craype-haswell craype-mic-knl
cc -o mytest mytest_code.c
```

  - Using Cray compiler

```
module swap PrgEnv-intel PrgEnv-cray
module swap craype-haswell craype-mic-knl
cc -o mytest mytest_code.c
```



Running Jobs

Jobs at NERSC

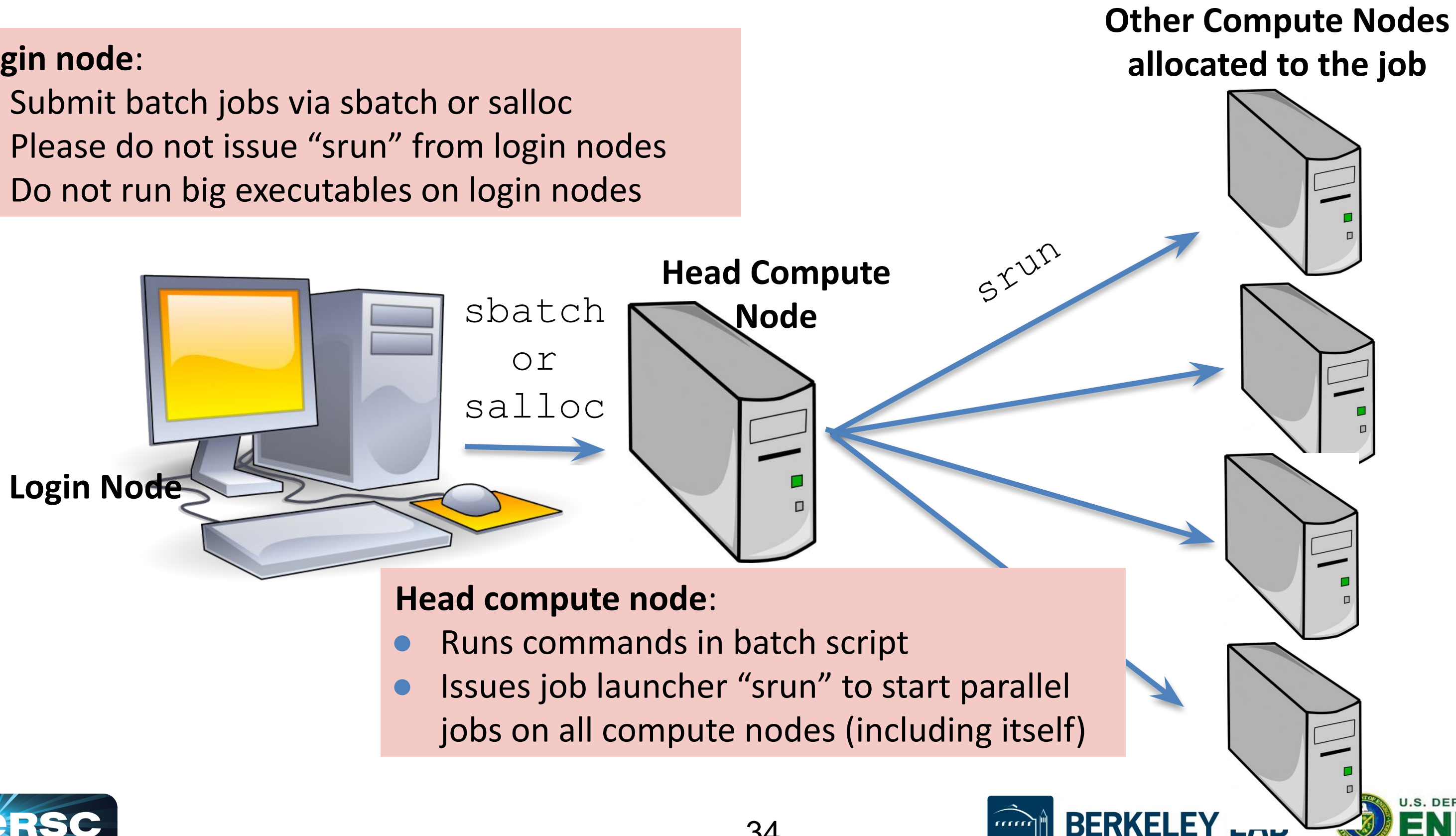
- Most are parallel jobs (10s to 100,000+ cores)
- Also a number of “serial” jobs
  - Typically “pleasantly parallel” simulation or data analysis
- Production runs execute in batch mode
- Our batch scheduler is SLURM
- Typical run times are a few to 10s of hours
  - Limits are necessary because of MTBF and the need to accommodate 7,000 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

Launching Parallel Jobs with Slurm

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



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

My First “Hello World” Program

```
my_batch_script:

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

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

32 MPI tasks per node in this example

- 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

Sample Cori Haswell Batch Script - Hybrid MPI/OpenMP

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

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

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

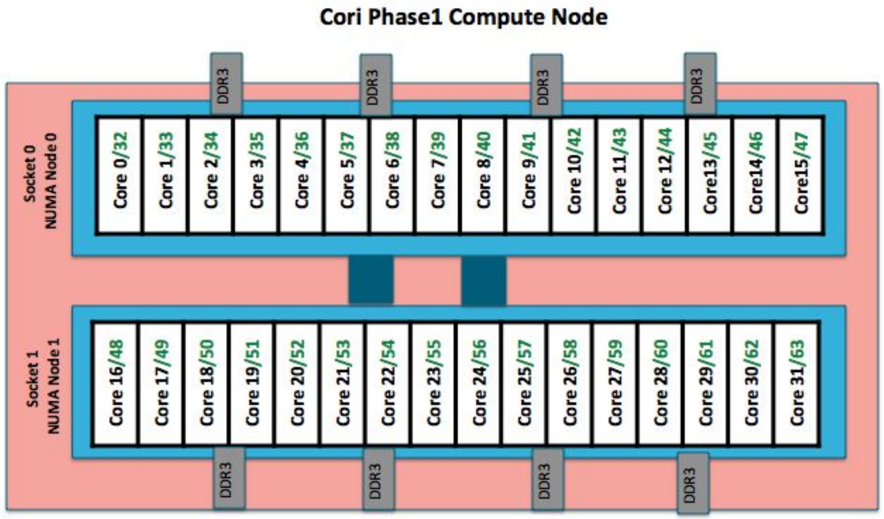
4 MPI tasks per node in this example

- 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 on Haswell, set 64 logical CPUs /4 =16 for "-c"
  - "-c" value should be >= OMP_NUM_THREADS

Process / Thread / Memory Affinity

- Correct process, thread and memory affinity is critical for getting optimal performance on Cori Haswell and KNL
  - 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
- Both -c xx and --cpu-bind=cores are essential, otherwise multiple processes may land on the same core, while other cores are idle, hurting performance badly
- Pay special attention on KNL, usually we waste (or aside for OS) 4 cores on purpose, to allow number of logical cores distributed evenly for each MPI rank
- https://docs.nersc.gov/jobs/affinity/

Cori Haswell Compute Nodes



To obtain processor info:

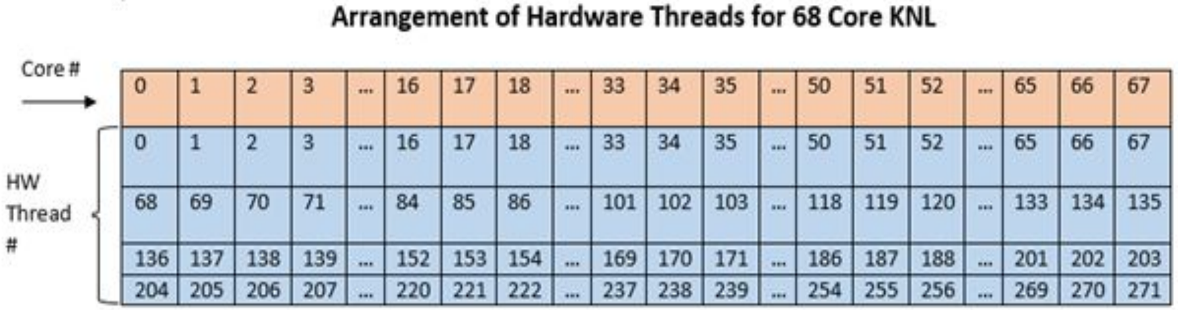
Get on a compute node:
% salloc -N 1 -C …

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

- Each Cori Haswell node has 2 Intel Xeon 16-core Haswell processors
  - **2 NUMA domains (sockets) per node, 16 cores per NUMA domain. 2 hardware threads per 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

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)
- Default mode 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.

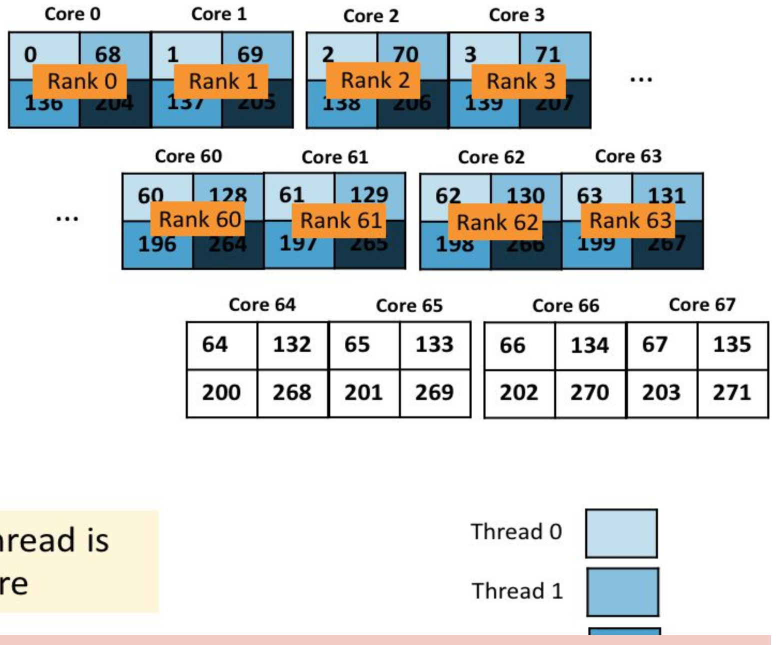
Sample Job Script to Run on KNL Nodes

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
```

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

Process and thread affinity



- Again, specify #_logical_CPUs to be allocated to each MPI task
  - with 64 MPI tasks per node on KNL, set 256 logical CPUs /64 =4 for ”-c”

Use “shared” QOS to Run Serial Jobs

- The “shared” QOS allows multiple executables from different users to share a node
- Each serial job run on a single physical 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
```

- Only available on Cori Haswell, charged by a fraction of a node used
- https://docs.nersc.gov/jobs/best-practices/#serial-jobs

Use salloc to Run Debug and Interactive Jobs

- You can run small parallel jobs interactively on dedicated nodes
- Debug
  - Max 512 nodes, up to 30 min

    ```
    % salloc -N 20 -q debug -C haswell -t 30:00
    ```

- Interactive (highly recommend to use this!!)
  - Instant allocation (get nodes in 6 min or reject)
  - Max walltime 4 hrs, up to 64 nodes total on Cori per project

    ```
    % salloc -N 2 -q interactive -C knl -t 2:00:00
    ```

  - More information (such as how to find out who in your project is using)
    - https://docs.nersc.gov/jobs/examples/#interactive
    - https://docs.nersc.gov/jobs/interactive/

Advanced Running Jobs Options

- Bundle jobs (multiple “srun”s in one script, sequentially or simultaneously)
- Use job dependency features to chain jobs
- Use Job Arrays to manage collections of similar jobs
- Run variable-time jobs and “flex” qos to run longer jobs
- Use workflow tools to manage jobs
- Use Burst Buffer for faster IO
- Use Shifter for jobs with custom user environment
- Use “xfer” for transferring to/from HPSS
- Use “bigmem” for large memory jobs

Bundle Jobs

Multiple Jobs Sequentially:

```
#!/bin/bash
#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
```

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

Multiple Jobs Simultaneously:

```
#!/bin/bash
#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
```

- 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

Dependency Jobs

```
cori% sbatch job1
Submitted batch job 1655447

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 1655447

cori06% cat job2
#!/bin/bash
#SBATCH -q regular
#SBATCH -N 1
#SBATCH -t 1:30:00
#SBATCH -d afterok:1655447
#SBATCH -C haswell
srun -n 16 -c 4 ./a.out

cori06% sbatch job2
```

Job Arrays

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

Use Workflow Management Tools

- These tools can help data-centric science to automate moving data, multi-step processing, and visualization at scales.
- Please do not do below!

```
for i = 1, 10000
     srun -n 1 ./a.out
```

  It is inefficient and overwhelms Slurm scheduler

- Available workflow tools include: GNU parallel, Taskfarmer, Fireworks, Nextflow, Papermill, etc.
- One usage case is to pack large number of serial jobs into one script
- https://docs.nersc.gov/jobs/workflow-tools/

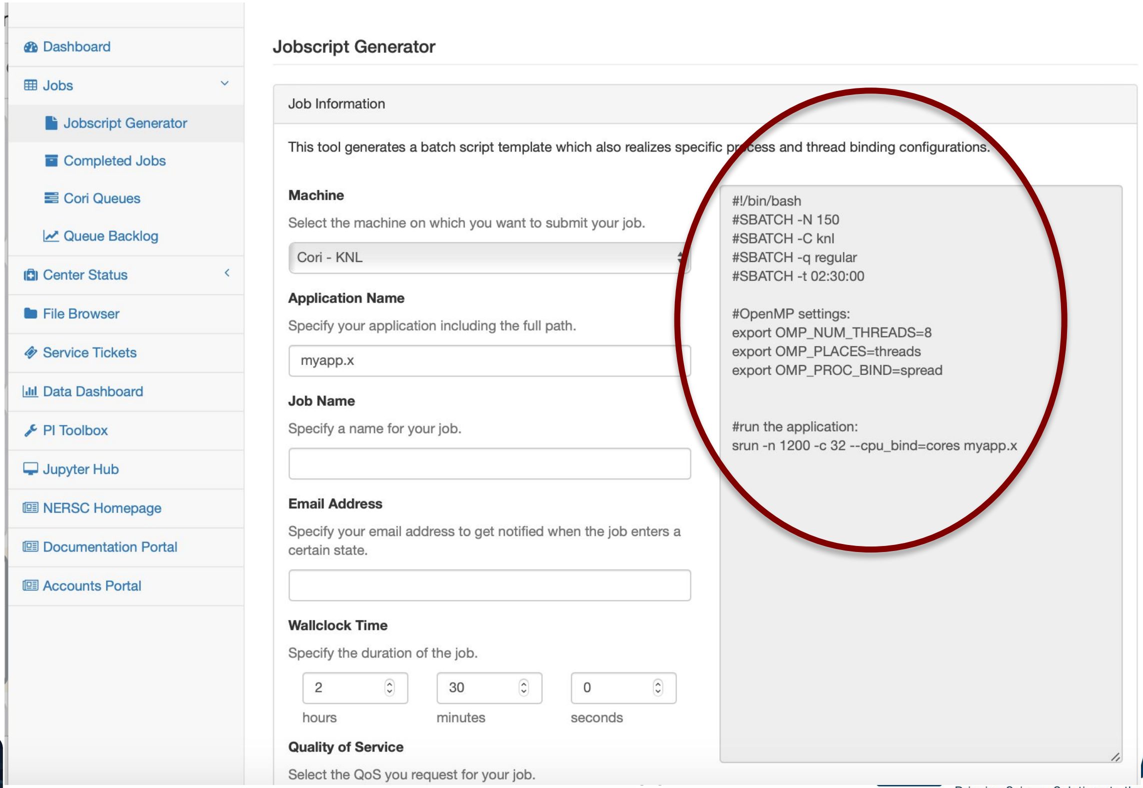
GNU Parallel Is Better Than Shared QOS

```
elvis@cori07:~> module load parallel

elvis@cori07:~> seq 1 5 | parallel -j 2 'echo \
> "Hello world {}!"; sleep 10; date'
Hello world 1!
Thu Jun 11 00:21:00 PDT 2020
Hello world 2!
Thu Jun 11 00:21:00 PDT 2020
Hello world 3!
Thu Jun 11 00:21:10 PDT 2020
Hello world 4!
Thu Jun 11 00:21:10 PDT 2020
Hello world 5!
Thu Jun 11 00:21:20 PDT 2020
elvis@cori07:~>
```

- Packed jobs have massively reduced total queue wait
  - Can also pack single-node tasks into multiple node jobs
- No risk of Slurm overload
- Run combinations of tasks in parallel and sequence
- Easy input substitution
  - If you need it, *much* more power is available
- Superior to task arrays, too
- https://docs.nersc.gov/jobs/workflow/gnuparallel/

NERSC Job Script Generator

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



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.

Email Address

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

Wallclock Time

Specify the duration of the job.

2 hours 30 minutes 0 seconds

Quality of Service

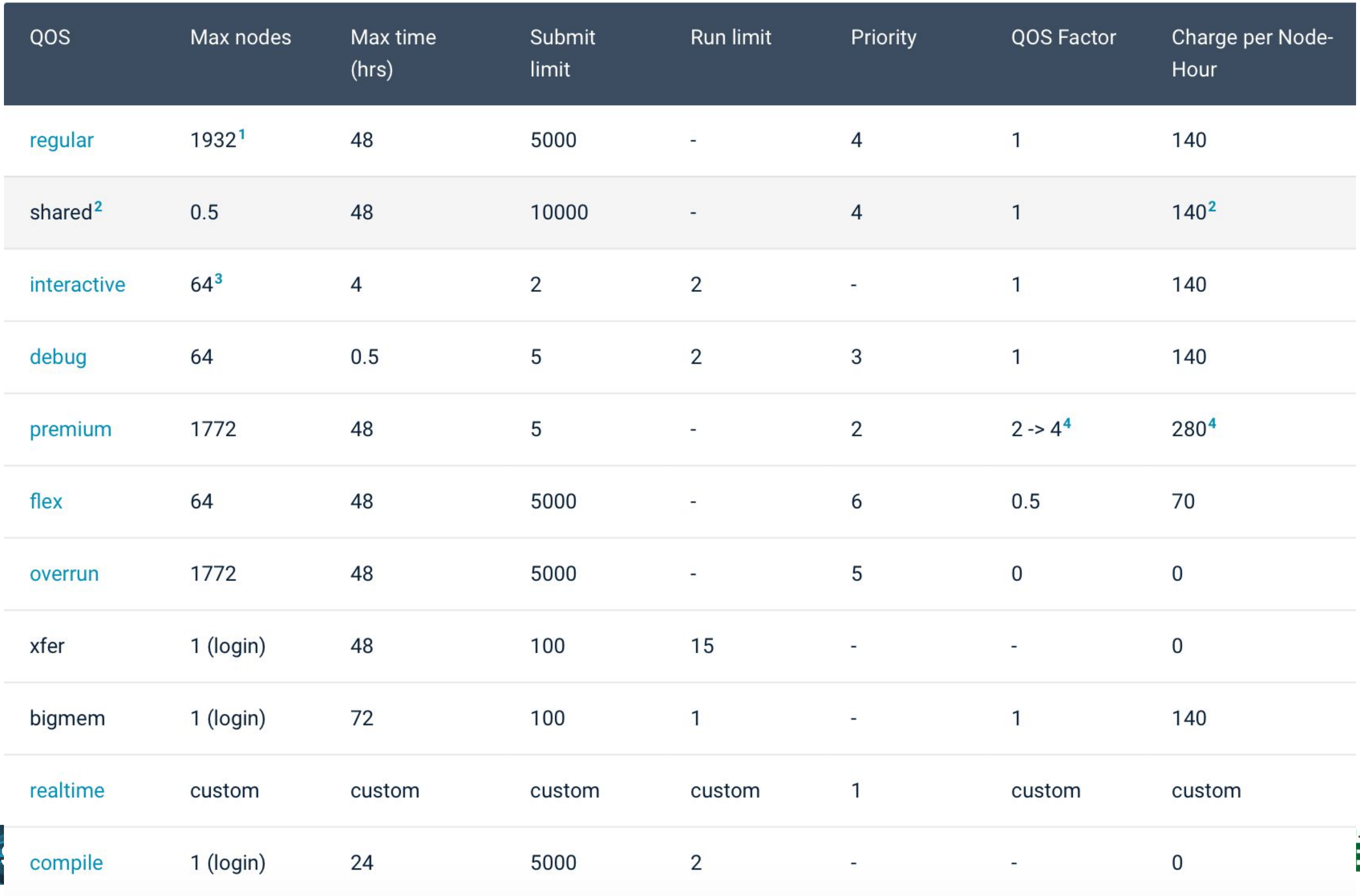
Select the QoS you request for your job.

```
#!/bin/bash
#SBATCH -N 150
#SBATCH -C knl
#SBATCH -q regular
#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 Jobs

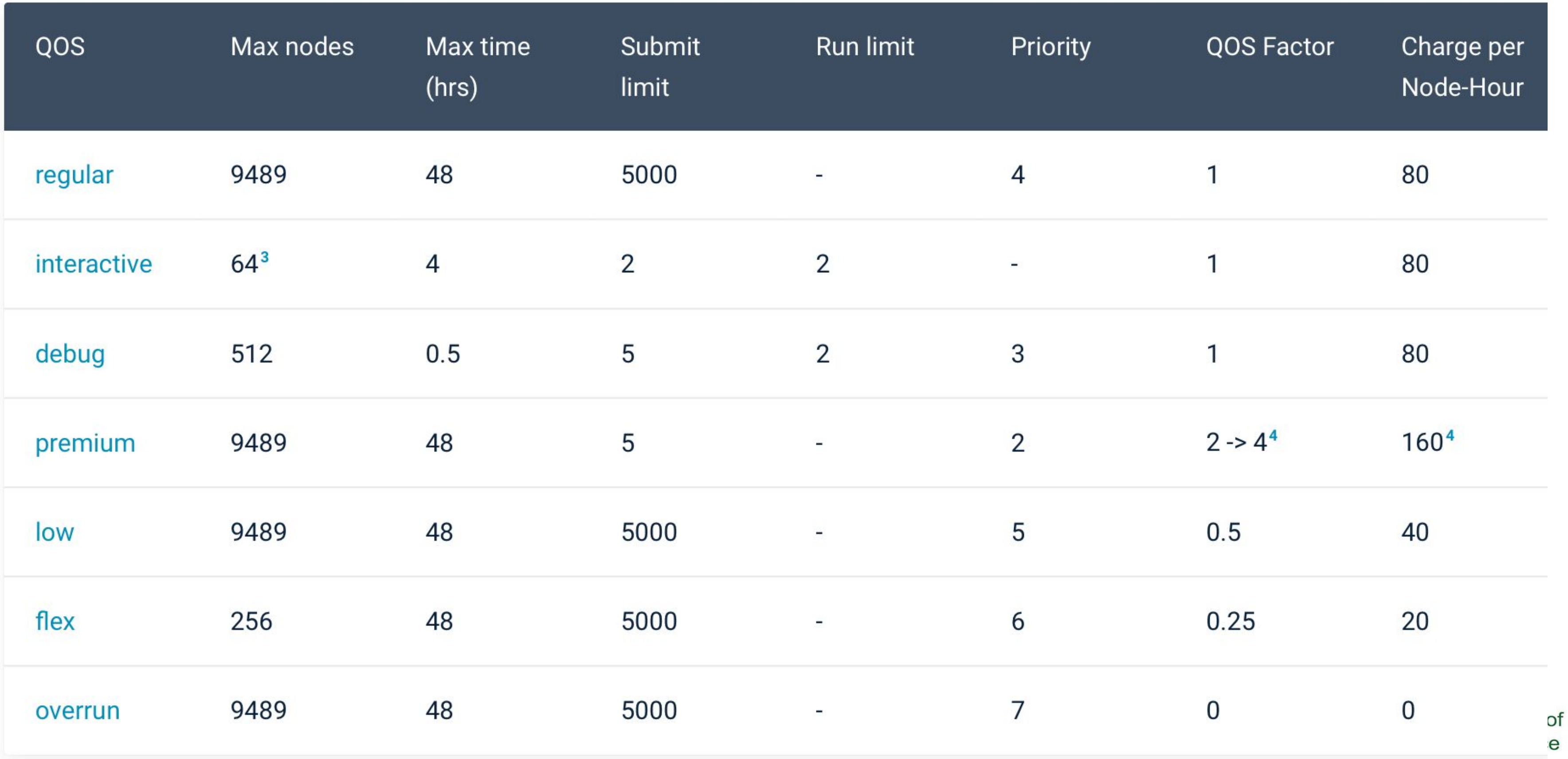
- Jobs are waiting in the queue until resources are available
- Overall job priorities are a combination of QOS, queue wait time, job size, wall time request, etc.
- You can monitor with
  - `squeue`: Slurm native command
  - `sqs`: NERSC custom wrapper script
  - `sacct`: Query Completed and Pending Jobs
  - https://docs.nersc.gov/jobs/monitoring/
- On the web
  - https://my.nersc.gov
    - Cori Queues, Queue backlogs, Queue Wait Times (statistics data)
  - https://www.nersc.gov/users/live-status/ □ Queue Look
  - https://iris.nersc.gov the “Jobs” tab

Cori Haswell Queue Policy (as of June 2021)



QOS	Max nodes	Max time (hrs)	Submit limit	Run limit	Priority	QOS Factor	Charge per Node-Hour
regular	1932[1]	48	5000	-	4	1	140
shared[2]	0.5	48	10000	-	4	1	140[2]
interactive	64[3]	4	2	2	-	1	140
debug	64	0.5	5	2	3	1	140
premium	1772	48	5	-	2	2 -> 4[4]	280[4]
flex	64	48	5000	-	6	0.5	70
overrun	1772	48	5000	-	5	0	0
xfer	1 (login)	48	100	15	-	-	0
bigmem	1 (login)	72	100	1	-	1	140
realtime	custom	custom	custom	custom	1	custom	custom
compile	1 (login)	24	5000	2	-	-	0

Cori KNL Queue Policy (as of June 2021)



QOS	Max nodes	Max time (hrs)	Submit limit	Run limit	Priority	QOS Factor	Charge per Node-Hour
regular	9489	48	5000	-	4	1	80
interactive	64[3]	4	2	2	-	1	80
debug	512	0.5	5	2	3	1	80
premium	9489	48	5	-	2	2 -> 4[4]	160[4]
low	9489	48	5000	-	5	0.5	40
flex	256	48	5000	-	6	0.25	20
overrun	9489	48	5000	-	7	0	0

Tips for Getting Better Throughput

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

Large Jobs Considerations

- sbcast your executables to compute nodes before srun

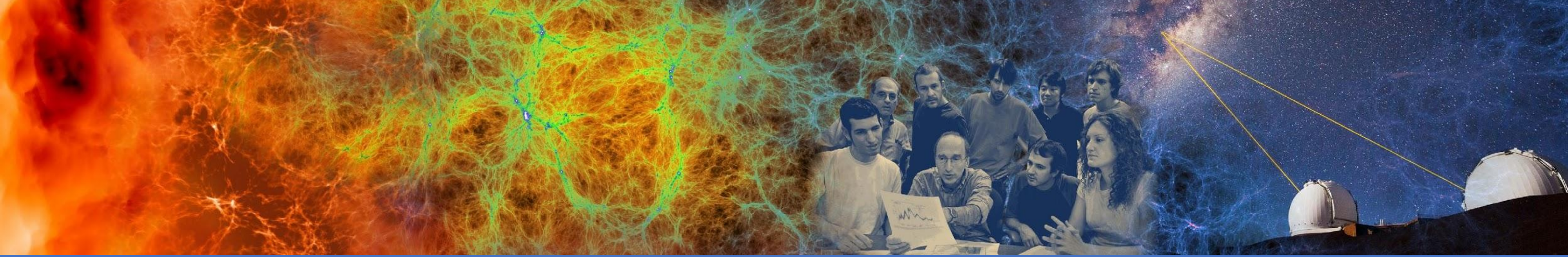
```
sbcast --compress=lz4 /path/to/exe /tmp/exe
srun /tmp/exe
```

  https://docs.nersc.gov/jobs/best-practices/#large-jobs

- Consider to build statically to run large jobs
  - There may be considerable startup delays for running large jobs of dynamic executables
- Consider to use shifter for large jobs using shared libraries
- Consider to use burst buffer for jobs doing large IO

Other Running Jobs Considerations

- Remember to compile separately for each type of compute nodes
- Running jobs from global homes is strongly discouraged
  - IO is not optimized
  - The global homes file system access on compute nodes is much slower than from $SCRATCH
  - It may also cause negative impact for other users interactive response on the system
- Consider to put your project’s shared software in /global/common/software/<project>
  - It is mounted read-only on compute nodes, so has less impact than other GPFS file systems (global homes or community file system)
- Consider to adopt workflow tools for better managing your jobs



Data Analytics Software and Services

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 Data Software Stack

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

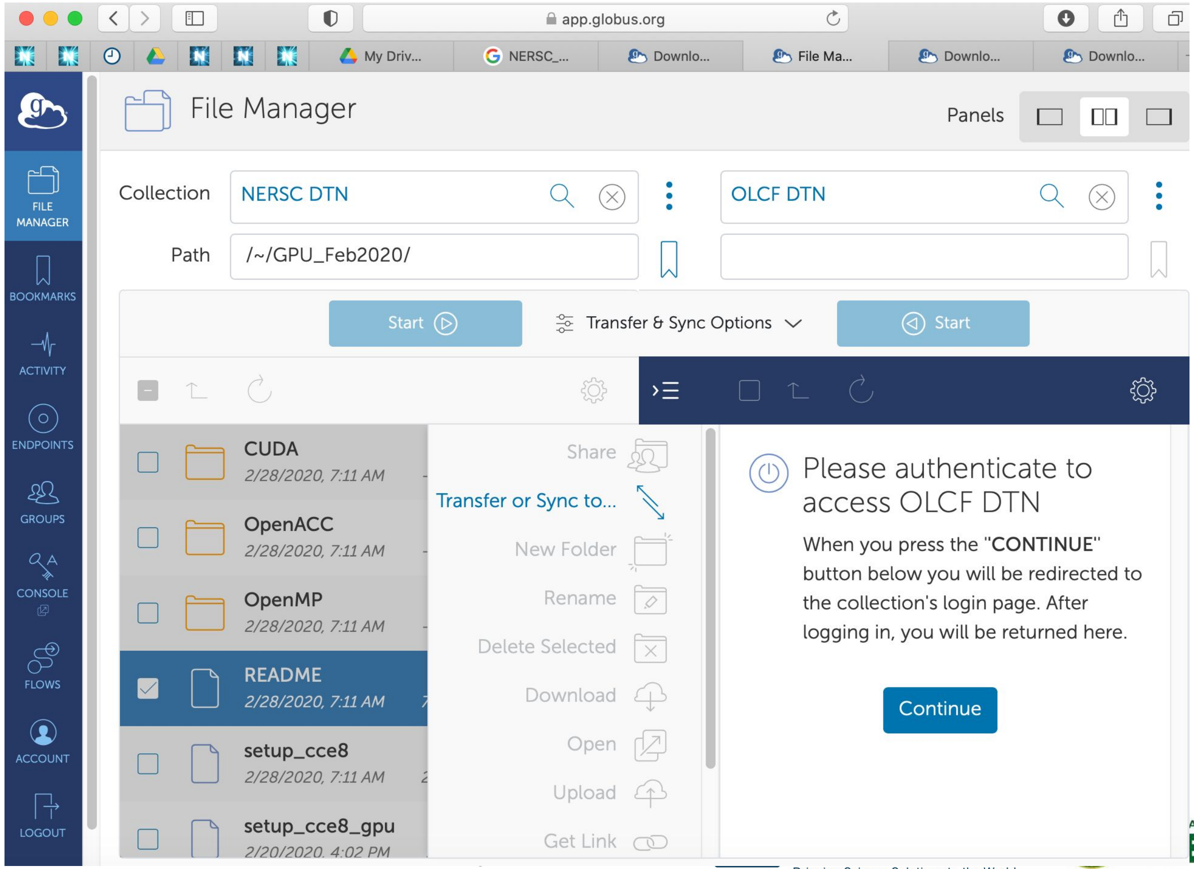
Data Analytic Software Services

- Globus Online
- Science Gateways
- Databases
- Shifter
- Burst Buffer
- Python
- Jupyter
- Machine Learning / Deep Learning
- Workflows
- And more …

Globus Online: Move Data

- https://www.globus.org   https://docs.nersc.gov/services/globus/
- The recommended tool for moving data in&out of NERSC
  - Reliable & easy-to-use web-based service:
    - Automatic retries
    - Email notification of success or failure
  - NERSC managed endpoints for optimized data transfers
    - NERSC DTN (dedicated data transfer system), NERS Cori, NERSC HPSS, etc.
  - Other Center has endpoints
  - Setup Globus Connect Personal to ease transfer between local system (such as laptop) and NERSC systems
  -

Globus File Transfer Example



app.globus.org

File Manager

Collection: NERSC DTN | OLCF DTN

Path: /~/GPU_Feb2020/

CUDA 2/28/2020, 7:11 AM
OpenACC 2/28/2020, 7:11 AM
OpenMP 2/28/2020, 7:11 AM
README 2/28/2020, 7:11 AM
setup_cce8 2/28/2020, 7:11 AM
setup_cce8_gpu 2/20/2020, 4:02 PM

Please authenticate to access OLCF DTN

When you press the "**CONTINUE**" button below you will be redirected to the collection's login page. After logging in, you will be returned here.

Data Transfer General Tips

- Use Globus Online for large, automated or monitored transfers
- cp, scp, or rsync is fine for smaller, one-time transfers (<100 MB)
  - But note that Globus is also fine for small transfers
- Use give-and-take to share files between NERSC users
  - % give -u <receiving_user> <file or directory>
  - % take -u <sending_user> <filename>

Access for External Collaborators

- Web Portals
  - NERSC supports project-level public http access
    - Project specific area can be created:

      /global/cfs/cdirs/<your_project>/www
    - These are available for public access under the URL:

      http://portal.nersc.gov/cfs/<your_project>
  - Each repo has a /project space, can publish as above
- Special Science Gateways can be created. Sophisticated ones can be made with SPIN: https://docs.nersc.gov/services/spin/getting_started/
  - Details at: https://docs.nersc.gov/services/science-gateways/

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:

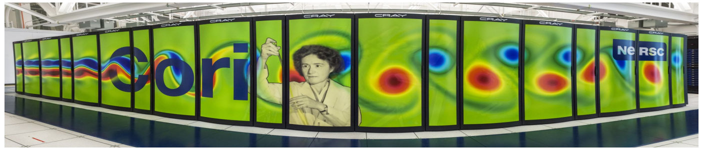
    Un-Structured Data (‘Schema-less’)

    Mid-Size to Large, e.g. 10 GB of Text
- More info and how to request a database: https://docs.nersc.gov/services/databases/

Shifter

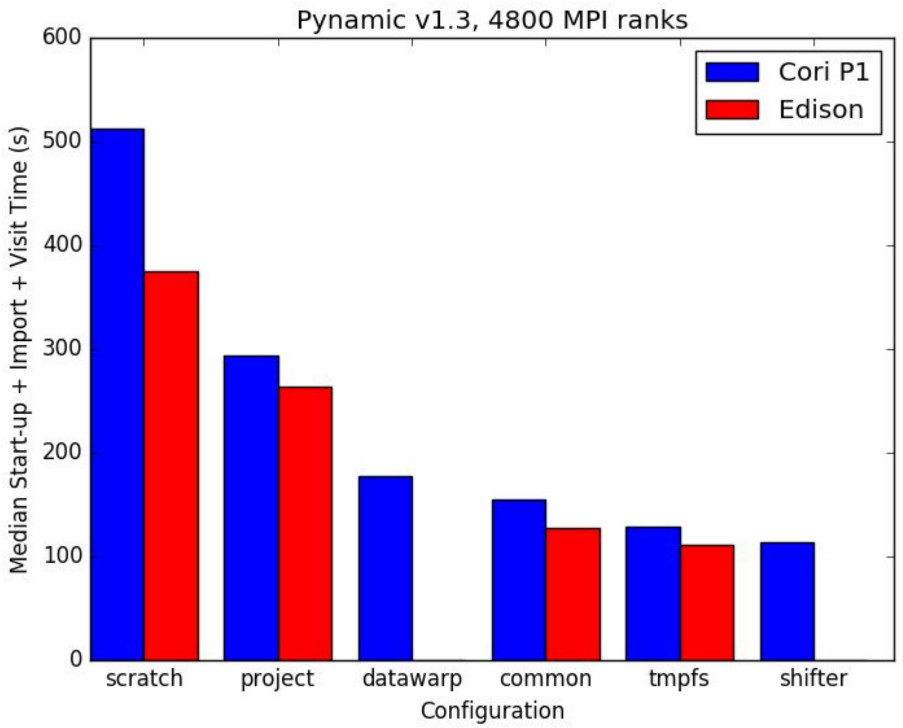
- NERSC R&D effort, in collaboration with Cray, to support Docker Application images
- “Docker-like” functionality on the Cray and HPC Linux clusters. Enables users to run custom environments on HPC systems.
- Addresses security issues in a robust way
- Efficient job-start & Native application performance

docker → SHIFTER →



https://docs.nersc.gov/development/shifter/how-to-use/

Shifter Accelerates Python Applications



Create an Image with Docker

```
FROM ubuntu:14.04
MAINTAINER Shane Canon scanon@lbl.gov
# Update packages and install dependencies
RUN apt-update -y && \
   apt-get install -y build-essential

# Copy in the application
ADD . /myapp
# Build it
RUN cd /myapp && \
    make && make install
```

Dockerfile

```
laptop> docker build -t scanon/myapp:1.1 .
laptop> docker push scanon/myapp:1.1
```

Use the Image with Shifter

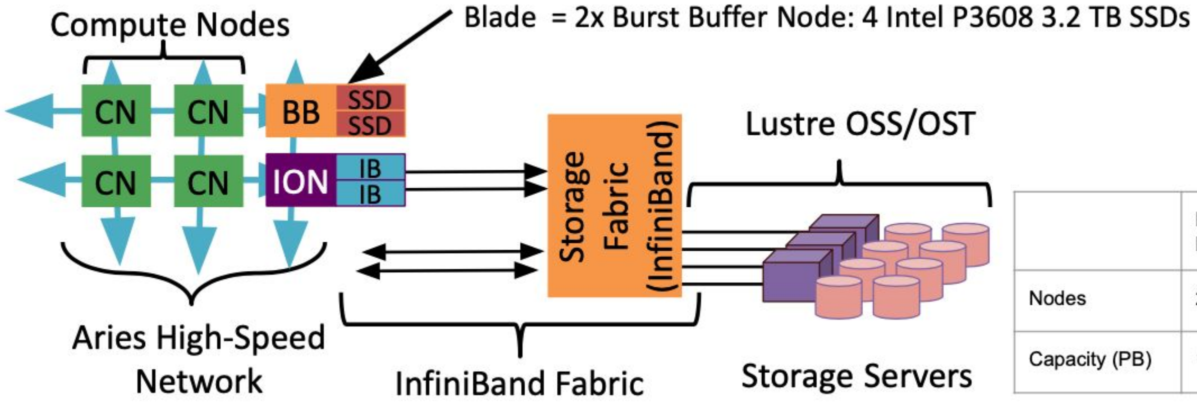
```
#!/bin/bash
#SBATCH -N 16 -t 20
#SBATCH --image=scanon/myapp:1.1

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

Submit script
job.sl

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

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 for multiple jobs to use



	Burst Buffer	Lustre
Nodes	288	248
Capacity (PB)	1.8	28

- https://docs.nersc.gov/jobs/examples/#burst-buffer

Burst Buffer Example

```
#!/bin/bash
#SBATCH -q regular -N 10 -C haswell -t 00:10:00
#DW jobdw capacity=1000GB access_mode=striped type=scratch
#DW stage_in source=$SCRATCH/inputs destination=$DW_JOB_STRIPED/inputs \ type=directory
#DW stage_in source=$SCRATCH/file.dat destination=$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 and striped across multiple BB nodes
- Data ‘stage_in’ before job start and ‘stage_out’ after

Python

- Extremely popular interpreted language, continuing to grow
- Libraries like NumPy, SciPy, scikit-learn commonly used for scientific analysis
- Are used for ML/DL
- Python is fully supported at NERSC - we use Anaconda Python to provide pre-built environments and the ability for users to create their own environments
- Do not use /usr/bin/python, instead:

  ```
  module load python
  ```

  which already includes basic packages: numpy, scipy, mpi4py

Make Your Own Python Conda Environment

- To make a custom env

```
module load python
conda create -n myenv python=3.7
source activate myenv
conda (or pip) install your_custom_package
###import antigravity
source deactivate myenv
```

- To use the custom env later

```
source activate mynev   (# does not change your dot file
setup)
```
or
```
conda activate myenv    (# changes your dot file setup)
<...steps to use this conda env ... >
conda deactivate myenv
```

Options to Run Python Code in Parallel

- Multiprocessing
  - Single node only, process parallelism via a pool of workers
- Dask
  - Single or many nodes, framework to create a group of workers that execute tasks coordinated by a scheduler, nice visualization tools
- mpi4py
  - Single or many nodes, best performance when used together with a container (Docker/Shifter)
  - Do not pip install mpi4py or conda install mpi4py, follow instructions at https://docs.nersc.gov/development/languages/python/mpi4py/#mpi4py-in-your-custom-conda-environment
- https://docs.nersc.gov/development/languages/python/scaling-up/

What is Jupyter?

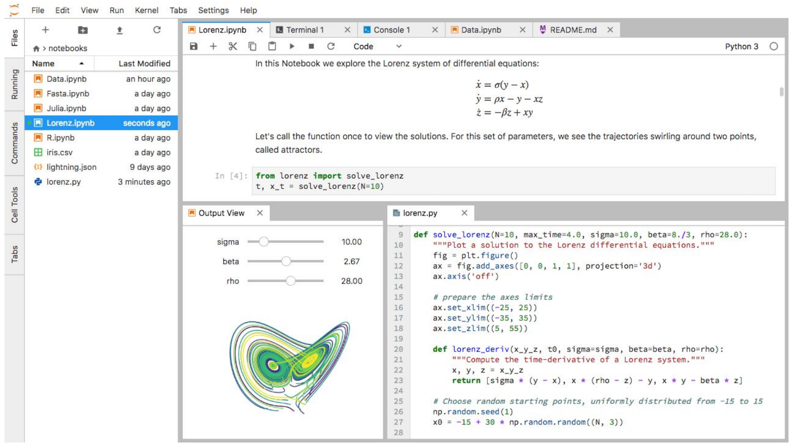
Interactive open-source web application

Allows you to <u>create</u> and <u>share</u> documents, “notebooks,” containing:

Live code
Equations
Visualizations
Narrative text
Interactive widgets

Things you can use Jupyter notebooks for:

Data cleaning and data transformation
Numerical simulation
Statistical modeling
Data visualization
Machine learning
Workflows and analytics frameworks
Training and Tutorials



Name	Last Modified
Data.ipynb	an hour ago
Fasta.ipynb	a day ago
Julia.ipynb	a day ago
Lorenz.ipynb	seconds ago
R.ipynb	a day ago
iris.csv	a day ago
lightning.json	9 days ago
lorenz.py	3 minutes ago

In this Notebook we explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

Let's call the function once to view the solutions. For this set of parameters, we see the trajectories swirling around two points, called attractors.

```
In [4]: from lorenz import solve_lorenz
        t, x_t = solve_lorenz(N=10)
```

sigma 10.00
beta 2.67
rho 28.00

```
def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./3, rho=28.0):
    """Plot a solution to the Lorenz differential equations."""
    fig = plt.figure()
    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
    ax.axis('off')

    # prepare the axes limits
    ax.set_xlim((-25, 25))
    ax.set_ylim((-35, 35))
    ax.set_zlim((5, 55))

    def lorenz_deriv(x_y_z, t0, sigma=sigma, beta=beta, rho=rho):
        """Compute the time-derivative of a Lorenz system."""
        x, y, z = x_y_z
        return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]

    # Choose random starting points, uniformly distributed from -15 to 15
    np.random.seed(1)
    x0 = -15 + 30 * np.random.random((N, 3))
```

Your Own Custom Jupyter Kernel

Most common Jupyter question:

"How do I take a conda environment and use it from Jupyter?"

Several ways to accomplish this, here's the easy one.

```
$ module load python
$ conda create -n myenv python=3.7
$ source activate myenv
(myenv) $ conda install ipykernel <other-packages>...
(myenv) $ python -m ipykernel install --user --name myenv-jupyter
```

Point your browser to jupyter.nersc.gov.
(You may need to restart your notebook server via control panel).
Kernel "myenv-jupyter" should be present in the kernel list.

Additional Customization

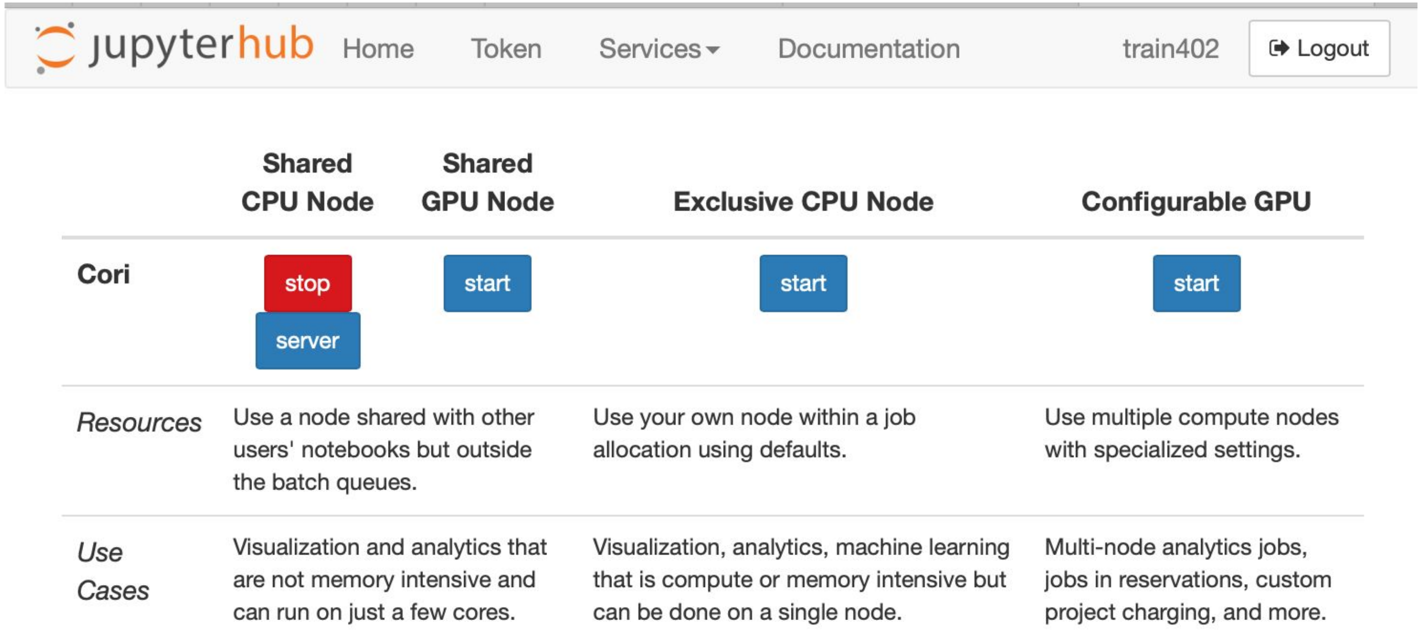
```
{
 "argv": [
  "/global/homes/y/yunhe/jupyter-helper.sh",
  "-f",
  "{connection_file}"
 ],
 "display_name": "myenv-jupyter2",
 "language": "python",
}
```

The helper script is the most flexible approach for NERSC users since it easily enables modules.

Meanwhile, in jupyter-helper.sh:

```
#!/bin/bash
export SOMETHING=123
module load texlive
exec python -m ipykernel "$@”
```

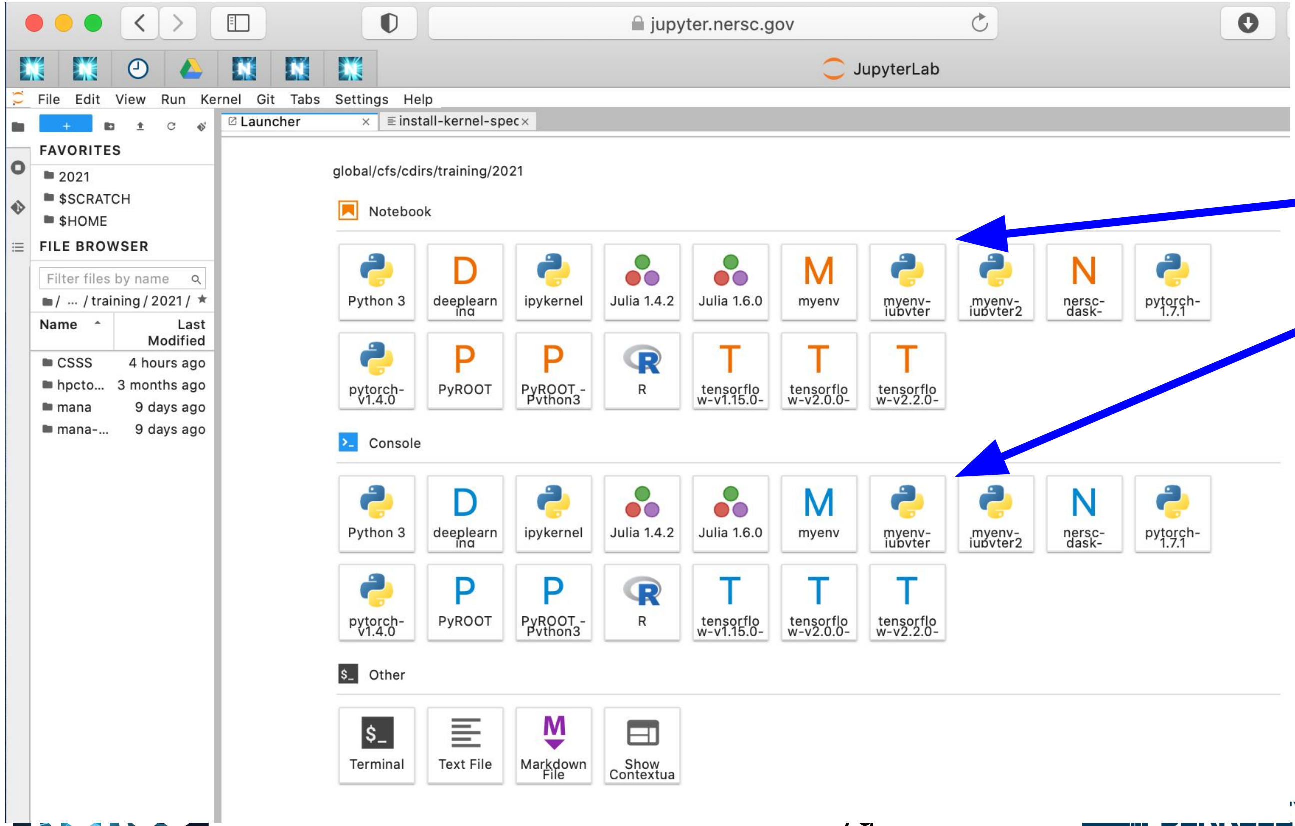
Available Notebook Servers



	Shared CPU Node	Shared GPU Node	Exclusive CPU Node	Configurable GPU
Cori	stop server	start	start	start
Resources	Use a node shared with other users' notebooks but outside the batch queues.		Use your own node within a job allocation using defaults.	Use multiple compute nodes with specialized settings.
Use Cases	Visualization and analytics that are not memory intensive and can run on just a few cores.		Visualization, analytics, machine learning that is compute or memory intensive but can be done on a single node.	Multi-node analytics jobs, jobs in reservations, custom project charging, and more.

Need to request access for exclusive CPU, and GPU nodes

Available Jupyter Kernels



Your own custom kernels

And many NERSC provided kernels: Python, Julia, ML/DL packages etc.

NERSC Deep Learning Software Stack Overview

General strategy:

- Provide functional, performant installations of the most popular frameworks and libraries
- Enable flexibility for users to customize and deploy their own solutions

Frameworks:

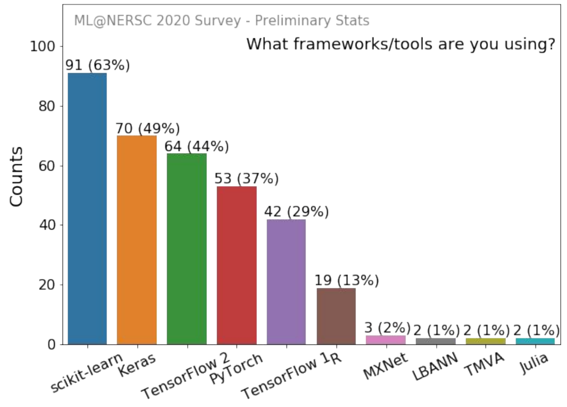
TensorFlow K Keras PyTorch

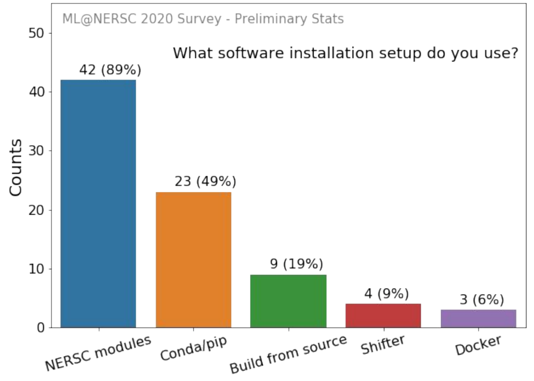
Distributed training libraries:

- Horovod
- PyTorch distributed
- Cray Plugin

Productive tools and services:

- Jupyter, Shifter





How to Use NERSC DL Software Stack

We have modules you can load which contain python and DL libraries:

```
module load tensorflow/intel-2.1.0-py37
module load pytorch/v1.5.0
```

Check which software versions are available with:

```
module avail tensorflow
```

You can install your own packages on top to customize:

```
pip install --user MY-PACKAGE
```

Or you can create your conda environments from scratch:

```
conda create -n my-env MY-PACKAGES
```

More on how to customize your setup can be found in the docs (TensorFlow, PyTorch).

We also have pre-installed Jupyter kernels.

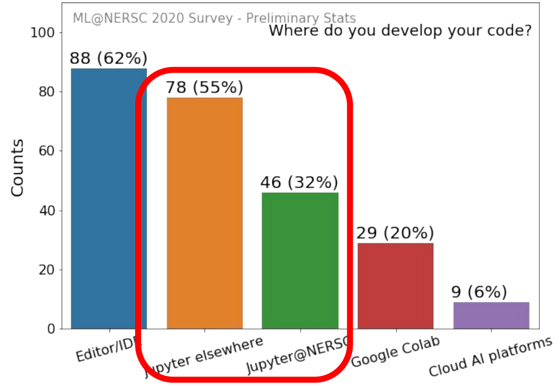
Jupyter for Deep Learning

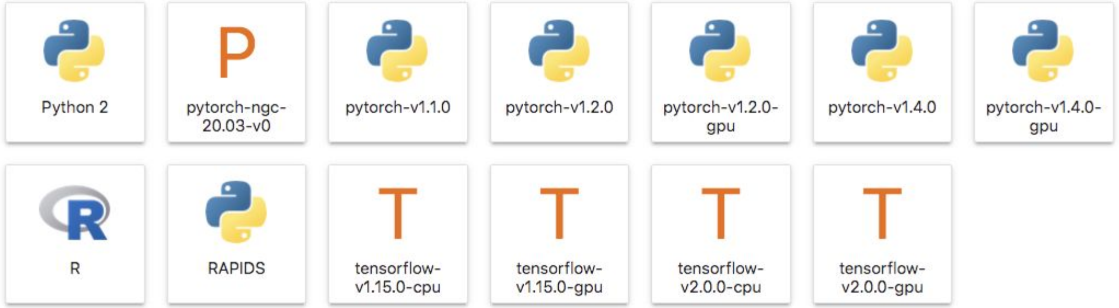
JupyterHub service provides a rich, interactive notebook ecosystem on Cori

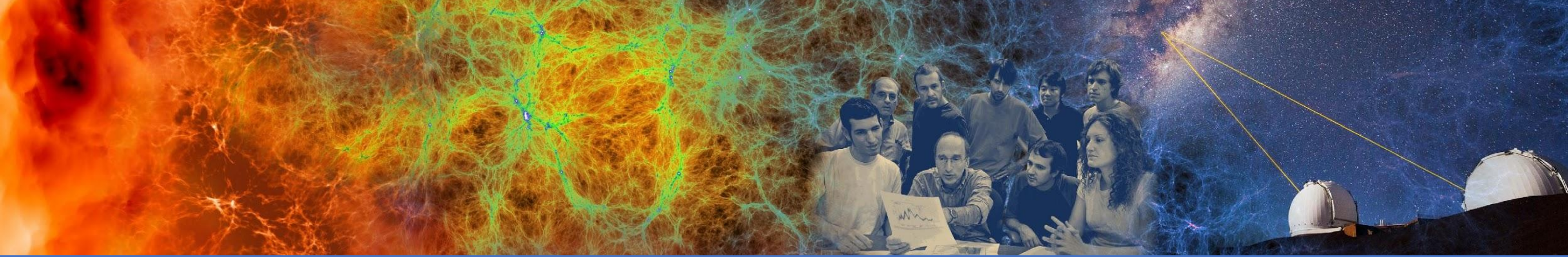
- Very popular service with hundreds of users
- A favorite way for users to develop ML code

Users can run their deep learning workloads

- on Cori CPU and Cori GPU
- using our pre-installed DL software kernels
- [using their own custom kernels]()



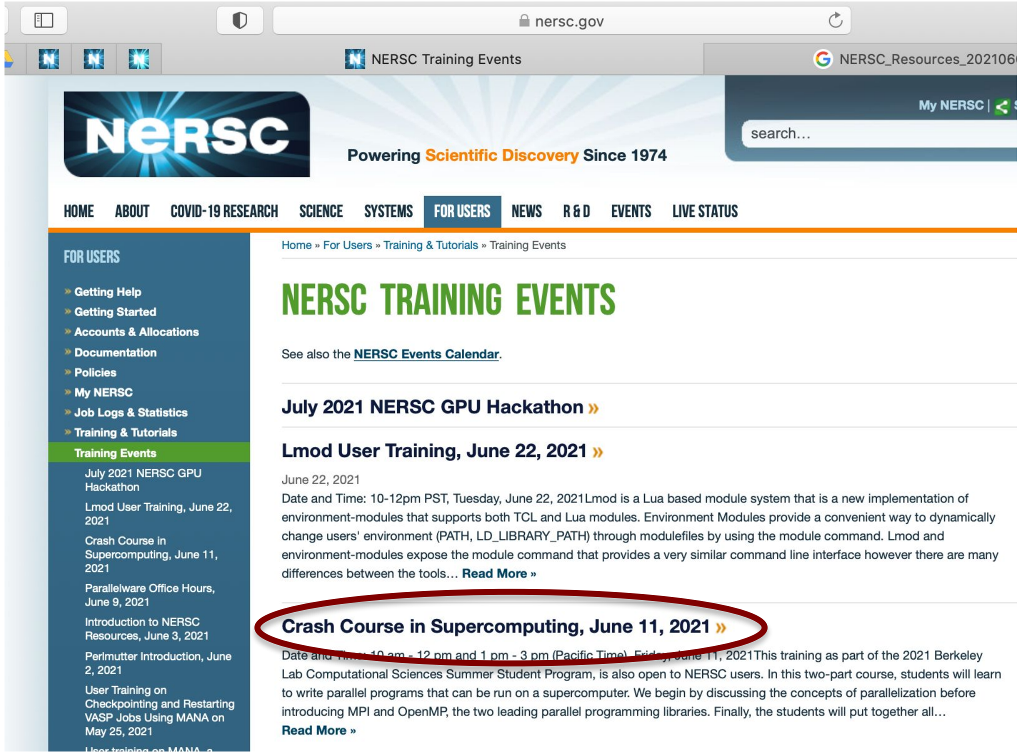




NERSC Online Resources

Online Resources: Classic NERSC Page

- https://www.nersc.gov
- Science, News, Publications
- Contact Us
- Live Status (MOTD): https://www.nersc.gov/live-status/motd/
- Training Events: https://www.nersc.gov/users/training/events/
- YouTube channel: NERSC
- NERSC users Slack channel
  - https://www.nersc.gov/users/NUG/nersc-users-slack/



nersc.gov
NERSC Training Events
NERSC_Resources_202106
NERSC
Powering Scientific Discovery Since 1974
My NERSC |
search...
HOME
ABOUT
COVID-19 RESEARCH
SCIENCE
SYSTEMS
FOR USERS
NEWS
R & D
EVENTS
LIVE STATUS
FOR USERS
» Getting Help
» Getting Started
» Accounts & Allocations
» Documentation
» Policies
» My NERSC
» Job Logs & Statistics
» Training & Tutorials
Training Events
July 2021 NERSC GPU Hackathon
Lmod User Training, June 22, 2021
Crash Course in Supercomputing, June 11, 2021
Parallelware Office Hours, June 9, 2021
Introduction to NERSC Resources, June 3, 2021
Perlmutter Introduction, June 2, 2021
User Training on Checkpointing and Restarting VASP Jobs Using MANA on May 25, 2021

Home » For Users » Training & Tutorials » Training Events

NERSC TRAINING EVENTS

See also the NERSC Events Calendar.

July 2021 NERSC GPU Hackathon »

Lmod User Training, June 22, 2021 »

June 22, 2021

Date and Time: 10-12pm PST, Tuesday, June 22, 2021Lmod is a Lua based module system that is a new implementation of environment-modules that supports both TCL and Lua modules. Environment Modules provide a convenient way to dynamically change users' environment (PATH, LD_LIBRARY_PATH) through modulefiles by using the module command. Lmod and environment-modules expose the module command that provides a very similar command line interface however there are many differences between the tools… **Read More »**

Crash Course in Supercomputing, June 11, 2021 »

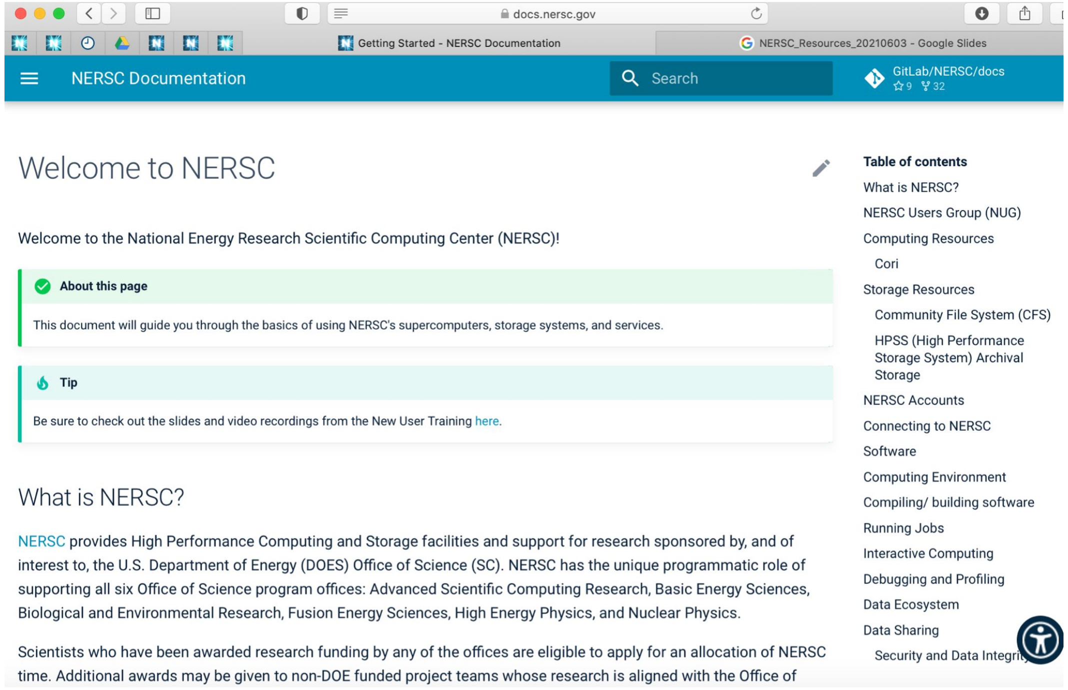
Date and Time: 10 am - 12 pm and 1 pm - 3 pm (Pacific Time), Friday, June 11, 2021This training as part of the 2021 Berkeley Lab Computational Sciences Summer Student Program, is also open to NERSC users. In this two-part course, students will learn to write parallel programs that can be run on a supercomputer. We begin by discussing the concepts of parallelization before introducing MPI and OpenMP, the two leading parallel programming libraries. Finally, the students will put together all…

Read More »

Online Resources: NERSC Docs

Technical Documentations
https://docs.nersc.gov

- Accounts
- IRIS
- Connecting
- Programming
- Running Jobs
- Applications
- Storage Systems
- Analytics
- Performance
- ...


docs.nersc.gov

Getting Started - NERSC Documentation | NERSC_Resources_20210603 - Google Slides

NERSC Documentation | Search | GitLab/NERSC/docs ☆9 32

Welcome to NERSC

Welcome to the National Energy Research Scientific Computing Center (NERSC)!

About this page

This document will guide you through the basics of using NERSC's supercomputers, storage systems, and services.

Tip

Be sure to check out the slides and video recordings from the New User Training here.

What is NERSC?

NERSC provides High Performance Computing and Storage facilities and support for research sponsored by, and of interest to, the U.S. Department of Energy (DOES) Office of Science (SC). NERSC has the unique programmatic role of supporting all six Office of Science program offices: Advanced Scientific Computing Research, Basic Energy Sciences, Biological and Environmental Research, Fusion Energy Sciences, High Energy Physics, and Nuclear Physics.

Scientists who have been awarded research funding by any of the offices are eligible to apply for an allocation of NERSC time. Additional awards may be given to non-DOE funded project teams whose research is aligned with the Office of

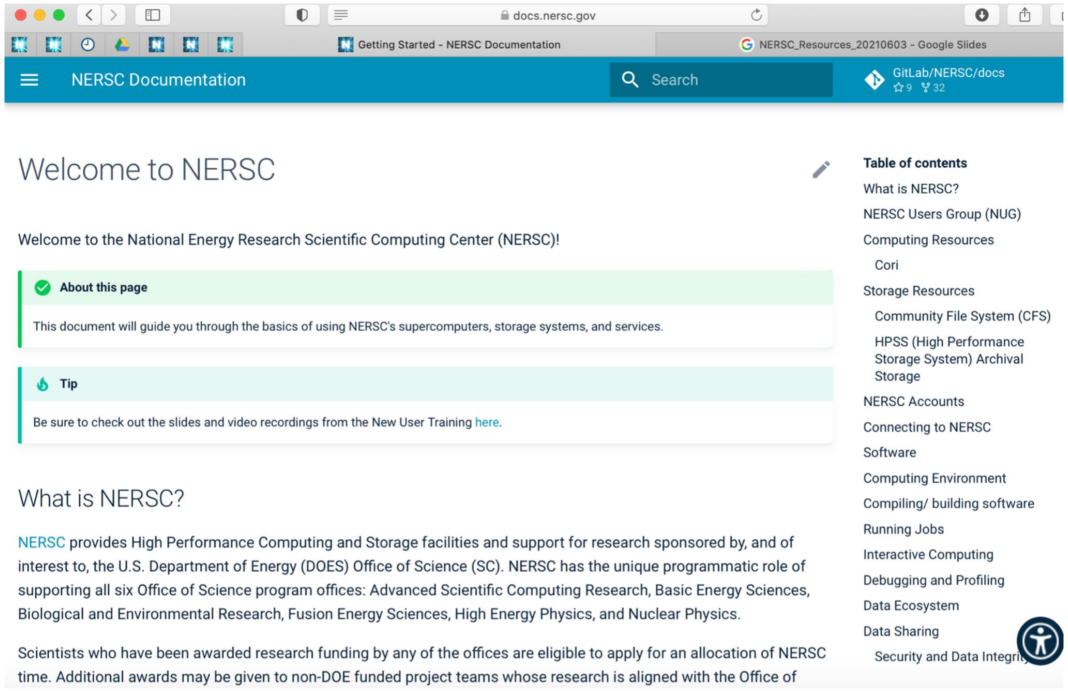
Table of contents
- What is NERSC?
- NERSC Users Group (NUG)
- Computing Resources
  - Cori
- Storage Resources
  - Community File System (CFS)
  - HPSS (High Performance Storage System) Archival Storage
- NERSC Accounts
- Connecting to NERSC
- Software
- Computing Environment
- Compiling/ building software
- Running Jobs
- Interactive Computing
- Debugging and Profiling
- Data Ecosystem
- Data Sharing
  - Security and Data Integrit

https://docs.nersc.gov/getting-started/

Online Resources: NERSC Docs

Technical Documentations
https://docs.nersc.gov

- Getting Started
https://docs.nersc.gov/getting-started/
- IRIS
- Systems
- Connecting
- Environment
- Development
- Running Jobs
- Applications
- Analytics
- Machine Learning
- Performance


docs.nersc.gov
Getting Started - NERSC Documentation
NERSC_Resources_20210603 - Google Slides
NERSC Documentation
Search
GitLab/NERSC/docs
☆9 ⑂32

Welcome to NERSC

Welcome to the National Energy Research Scientific Computing Center (NERSC)!

About this page

This document will guide you through the basics of using NERSC's supercomputers, storage systems, and services.

Tip

Be sure to check out the slides and video recordings from the New User Training here.

What is NERSC?

NERSC provides High Performance Computing and Storage facilities and support for research sponsored by, and of interest to, the U.S. Department of Energy (DOES) Office of Science (SC). NERSC has the unique programmatic role of supporting all six Office of Science program offices: Advanced Scientific Computing Research, Basic Energy Sciences, Biological and Environmental Research, Fusion Energy Sciences, High Energy Physics, and Nuclear Physics.

Scientists who have been awarded research funding by any of the offices are eligible to apply for an allocation of NERSC time. Additional awards may be given to non-DOE funded project teams whose research is aligned with the Office of

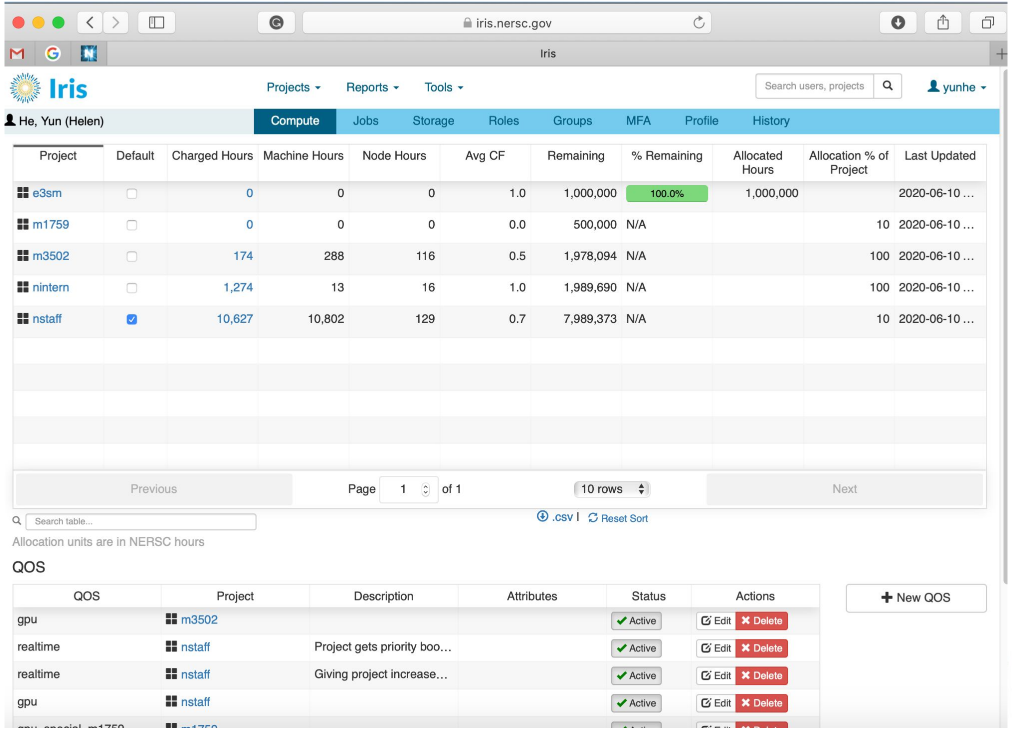
Table of contents

What is NERSC?
NERSC Users Group (NUG)
Computing Resources
Cori
Storage Resources
Community File System (CFS)
HPSS (High Performance Storage System) Archival Storage
NERSC Accounts
Connecting to NERSC
Software
Computing Environment
Compiling/ building software
Running Jobs
Interactive Computing
Debugging and Profiling
Data Ecosystem
Data Sharing
Security and Data Integrity

BERKELEY LAB
Bringing Science Solutions to the World
U.S. DEPARTMENT OF ENERGY
Office of Science

Online Resources: IRIS

- IRIS: NERSC Account Management and Reporting: https://iris.nersc.gov
  - Change password
  - Change contact info
  - SSH Keys, MFA
  - Check usage info



Project	Default	Charged Hours	Machine Hours	Node Hours	Avg CF	Remaining	% Remaining	Allocated Hours	Allocation % of Project	Last Updated
e3sm	☐	0	0	0	1.0	1,000,000	100.0%	1,000,000		2020-06-10 …
m1759	☐	0	0	0	0.0	500,000	N/A		10	2020-06-10 …
m3502	☐	174	288	116	0.5	1,978,094	N/A		100	2020-06-10 …
nintern	☐	1,274	13	16	1.0	1,989,690	N/A		100	2020-06-10 …
nstaff	☑	10,627	10,802	129	0.7	7,989,373	N/A		10	2020-06-10 …

Previous Page 1 of 1 10 rows Next

Search table... .csv | Reset Sort

Allocation units are in NERSC hours

QOS

QOS	Project	Description	Attributes	Status	Actions
gpu	m3502			✓ Active	Edit Delete
realtime	nstaff	Project gets priority boo…		✓ Active	Edit Delete
realtime	nstaff	Giving project increase…		✓ Active	Edit Delete
gpu	nstaff			✓ Active	Edit Delete

+ New QOS

Online Resources: Help Portal

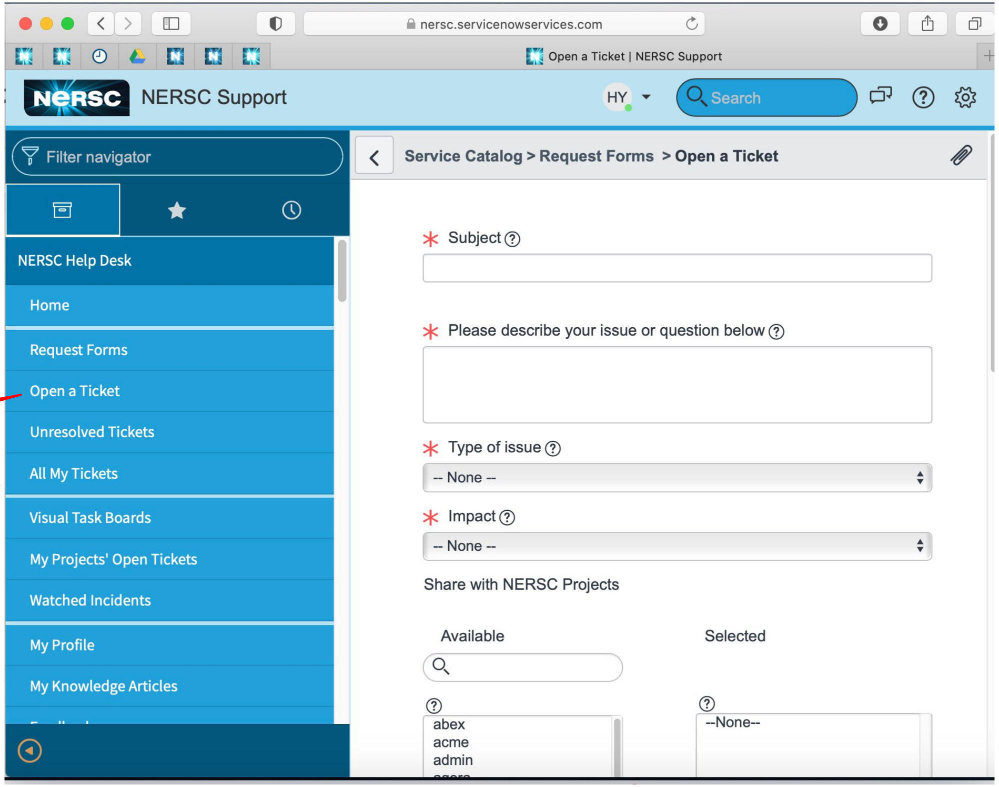
https://help.nersc.gov

- Submit tickets (ask questions)
- Request forms:
  - Quota Increase
  - Reservations
- Allocation (ERCAP) Requests

Open a ticket

All my tickets

My project's open tickets


nersc.servicenowservices.com
Open a Ticket | NERSC Support
NERSC Support

Service Catalog > Request Forms > Open a Ticket

* Subject

* Please describe your issue or question below

* Type of issue
-- None --

* Impact
-- None --

Share with NERSC Projects

Available: abex, acme, admin

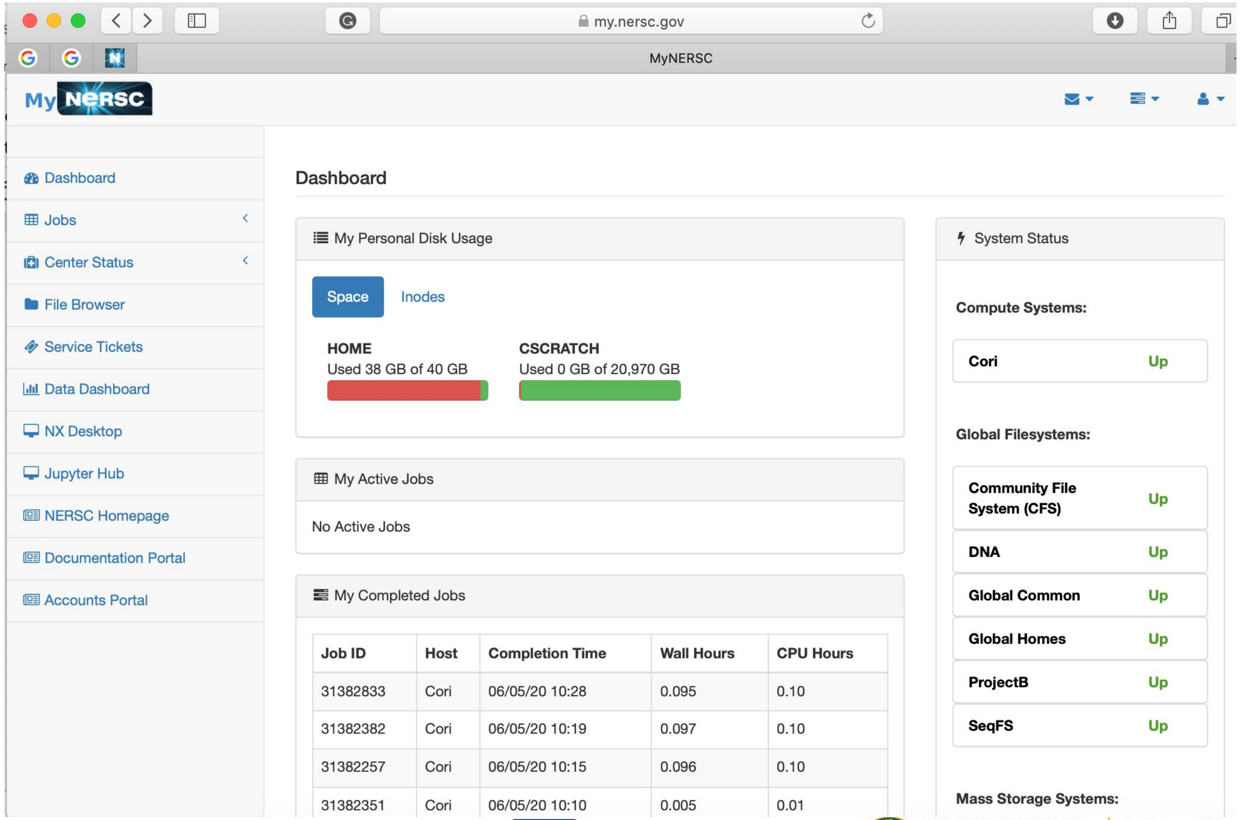
Selected: --None--

NERSC · BERKELEY LAB Bringing Science Solutions to the World · U.S. DEPARTMENT OF ENERGY Office of Science

Online Resources: MyNERSC

https://my.nersc.gov

- Dashboard
- Jobs
- Center Status
- File Browser
- Service Tickets
- Data Dashboard
- Jupyter Hub
- Links to other useful pages



Dashboard

My Personal Disk Usage

Space Inodes

HOME
Used 38 GB of 40 GB

CSCRATCH
Used 0 GB of 20,970 GB

My Active Jobs

No Active Jobs

My Completed Jobs

Job ID	Host	Completion Time	Wall Hours	CPU Hours
31382833	Cori	06/05/20 10:28	0.095	0.10
31382382	Cori	06/05/20 10:19	0.097	0.10
31382257	Cori	06/05/20 10:15	0.096	0.10
31382351	Cori	06/05/20 10:10	0.005	0.01

System Status

Compute Systems:

Cori	Up

Global Filesystems:

Community File System (CFS)	Up
DNA	Up
Global Common	Up
Global Homes	Up
ProjectB	Up
SeqFS	Up

Mass Storage Systems:

https://my.nersc.gov Leads You to All Sites

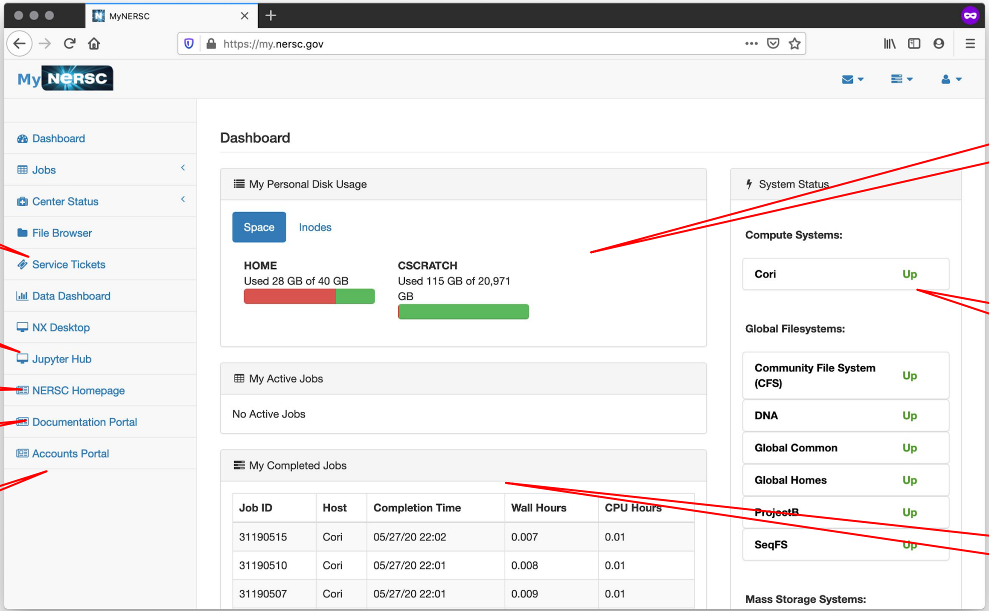
help.nersc.gov

jupyter.nersc.gov

www.nersc.gov

docs.nersc.gov

iris.nersc.gov



Dashboard

My Personal Disk Usage

Space | Inodes

HOME
Used 28 GB of 40 GB

CSCRATCH
Used 115 GB of 20,971 GB

My Active Jobs

No Active Jobs

My Completed Jobs

Job ID	Host	Completion Time	Wall Hours	CPU Hours
31190515	Cori	05/27/20 22:02	0.007	0.01
31190510	Cori	05/27/20 22:01	0.008	0.01
31190507	Cori	05/27/20 22:01	0.009	0.01

System Status

Compute Systems:

Cori	Up

Global Filesystems:

Community File System (CFS)	Up
DNA	Up
Global Common	Up
Global Homes	Up
ProjectB	Up
SeqFS	Up

Mass Storage Systems:

my disk quota

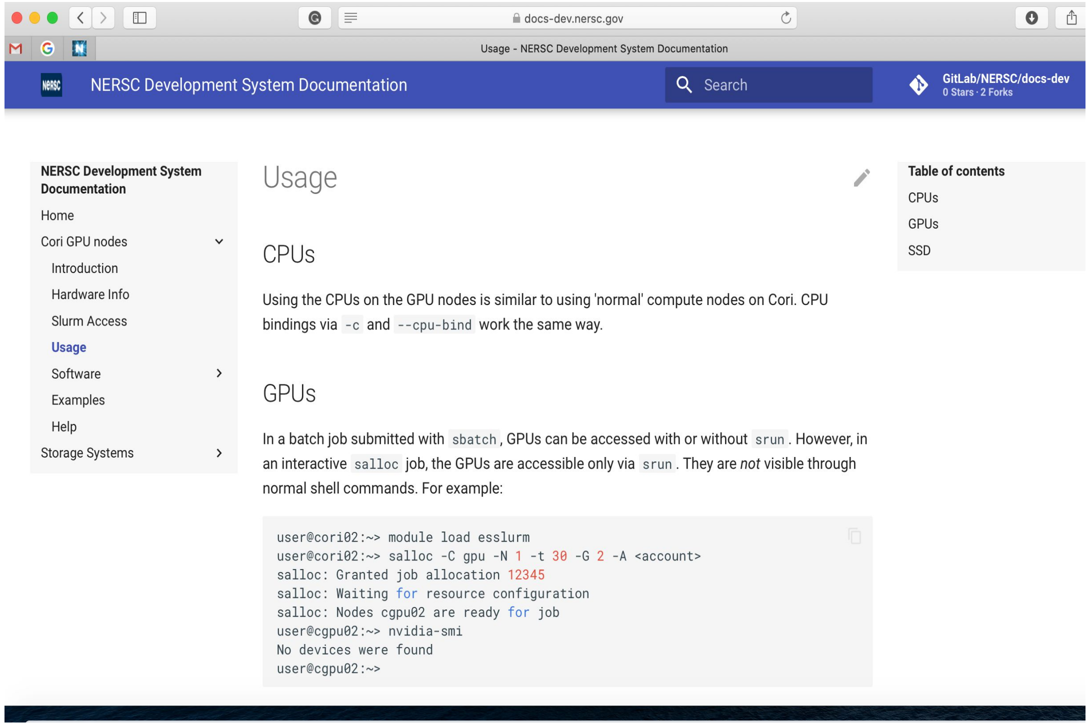
is cori up?

my jobs

Online Resources: Cori GPU Documentation

https://docs-dev.nersc.gov

- GPU nodes
  - Hardware info
  - Slurm access
  - Usage
  - Software
    - Compilers
    - Math libraries
    - Python
    - Shifter
    - Profiling
  - Examples



Usage

CPUs

Using the CPUs on the GPU nodes is similar to using 'normal' compute nodes on Cori. CPU bindings via `-c` and `--cpu-bind` work the same way.

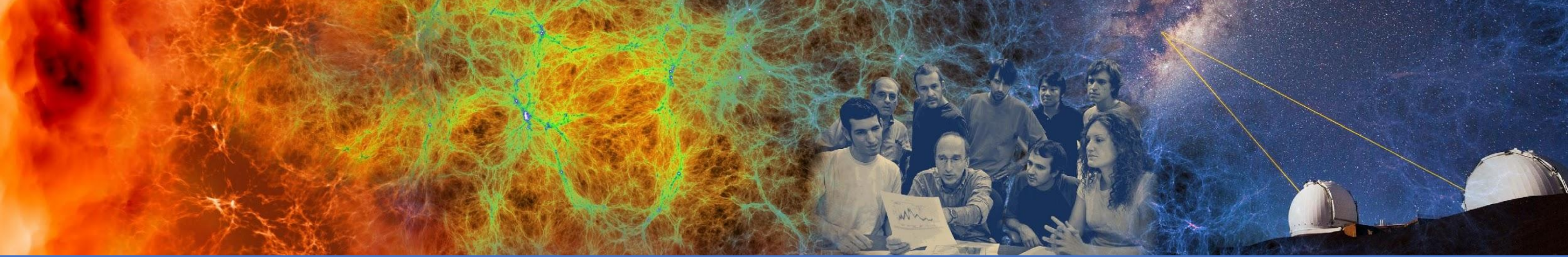
GPUs

In a batch job submitted with `sbatch`, GPUs can be accessed with or without `srun`. However, in an interactive `salloc` job, the GPUs are accessible only via `srun`. They are *not* visible through normal shell commands. For example:

```
user@cori02:~> module load esslurm
user@cori02:~> salloc -C gpu -N 1 -t 30 -G 2 -A <account>
salloc: Granted job allocation 12345
salloc: Waiting for resource configuration
salloc: Nodes cgpu02 are ready for job
user@cgpu02:~> nvidia-smi
No devices were found
user@cgpu02:~>
```

Acknowledgement

- Used / adapted some slides and materials from the NERSC New user training (June 16, 2020)
  - https://www.nersc.gov/users/training/events/new-user-training-june-16-2020/



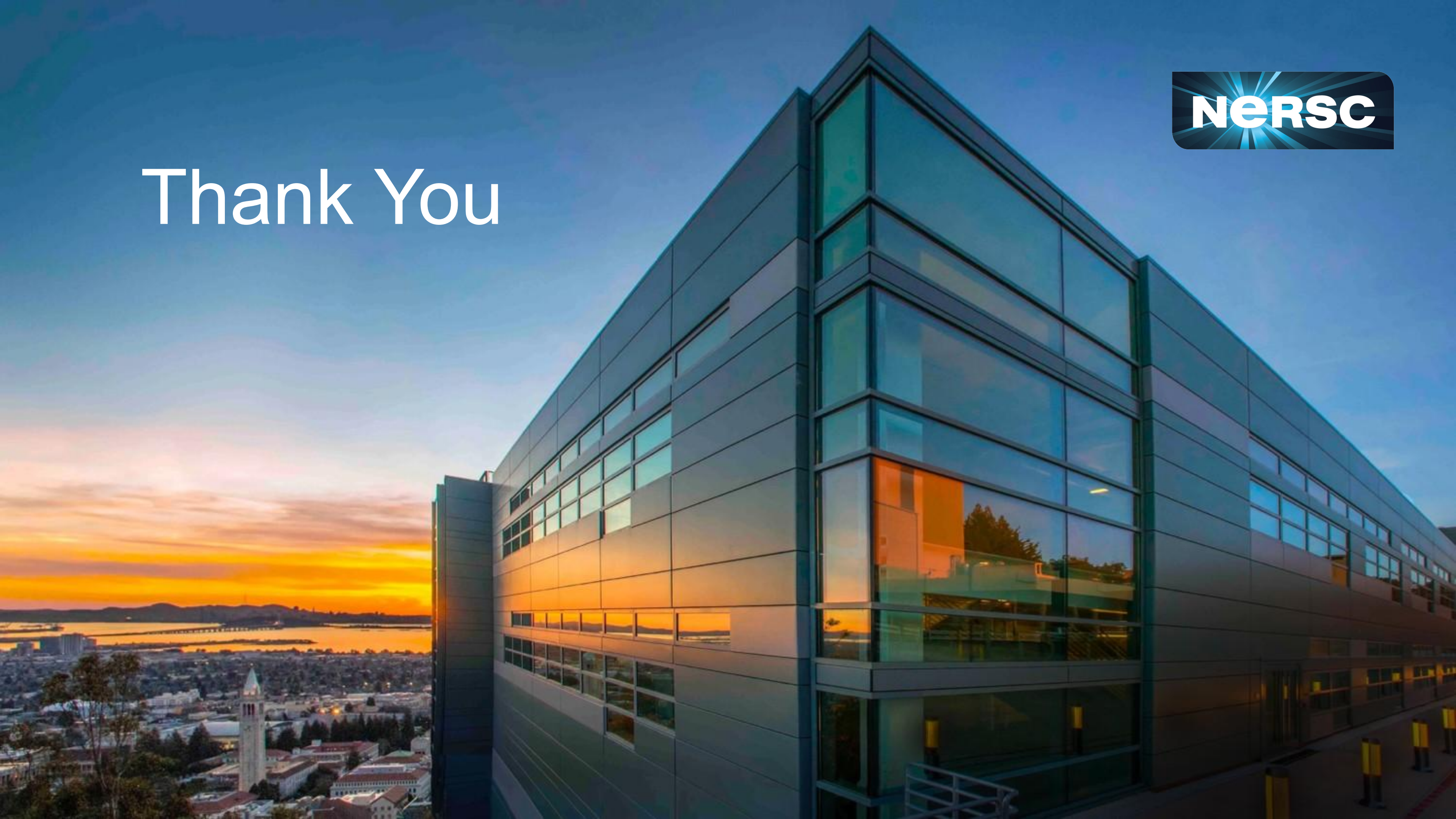
Hands-on Exercises

Hands-on Exercises

- % cd $SCRATCH
- % cp -r /global/cfs/cdirs/training/2021/CSSS .
  - Notice the space and the last dot in the above command
- % cd CSSS
- Follow:
  - hello-exercise.README
  - matrix-example.README
  - xthi-exercise.README
- References
  - Running Jobs: https://docs.nersc.gov/jobs/
  - Interactive Jobs: https://docs.nersc.gov/jobs/examples/#interactive

Using Compute Node Reservations

- Existing NERSC users are added to “nintern” project
- Cori node reservations available from 2-3:30 pm today
- User reservations with --reservation=xxx -A yyy, where
  - xxx is “intro_haswell” or “intro_knl”
  - yyy is “nintern” (existing users) or “ntrain” (trainxxx users)



Thank You