Introduction to NERSC Resources

LBNL CS Summer Program
June 8, 2023

Helen He
NERSC User Engagement Group

Some Logistics

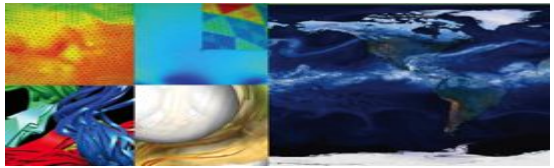
- Users are muted upon joining Zoom (can unmute to speak)
- Please change your name in Zoom session
  - to: first_name last_name
  - Click “Participants”, then “More” next to your name to rename
- Click the CC button to toggle captions and View Full Transcript
- GDoc is used for Q&A (instead of Zoom chat)
  - https://tinyurl.com/mtva7dar
- Slides and videos will be available on the Training Event page and CSA Summer Program page
  - https://www.nersc.gov/users/training/events/introduction-to-nersc-resources-jun2023/
  - https://cs.lbl.gov/careers/summer-student-and-faculty-program/2023-csa-summer-program/summer-program/
- Apply for a training account if no NERSC account or MFA not setup yet
  - https://iris.nersc.gov/train, and use the 4-letter code "aO7N"

Outline

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

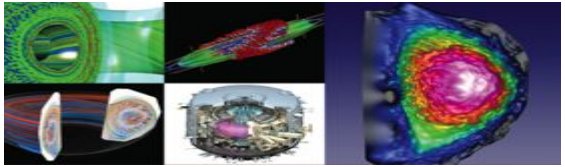
NERSC and Systems Overview

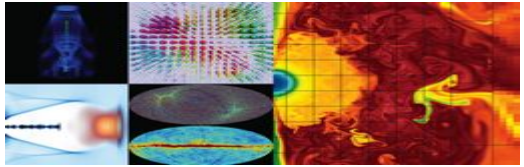
NERSC is the Mission HPC Computing Center for the DOE Office of Science



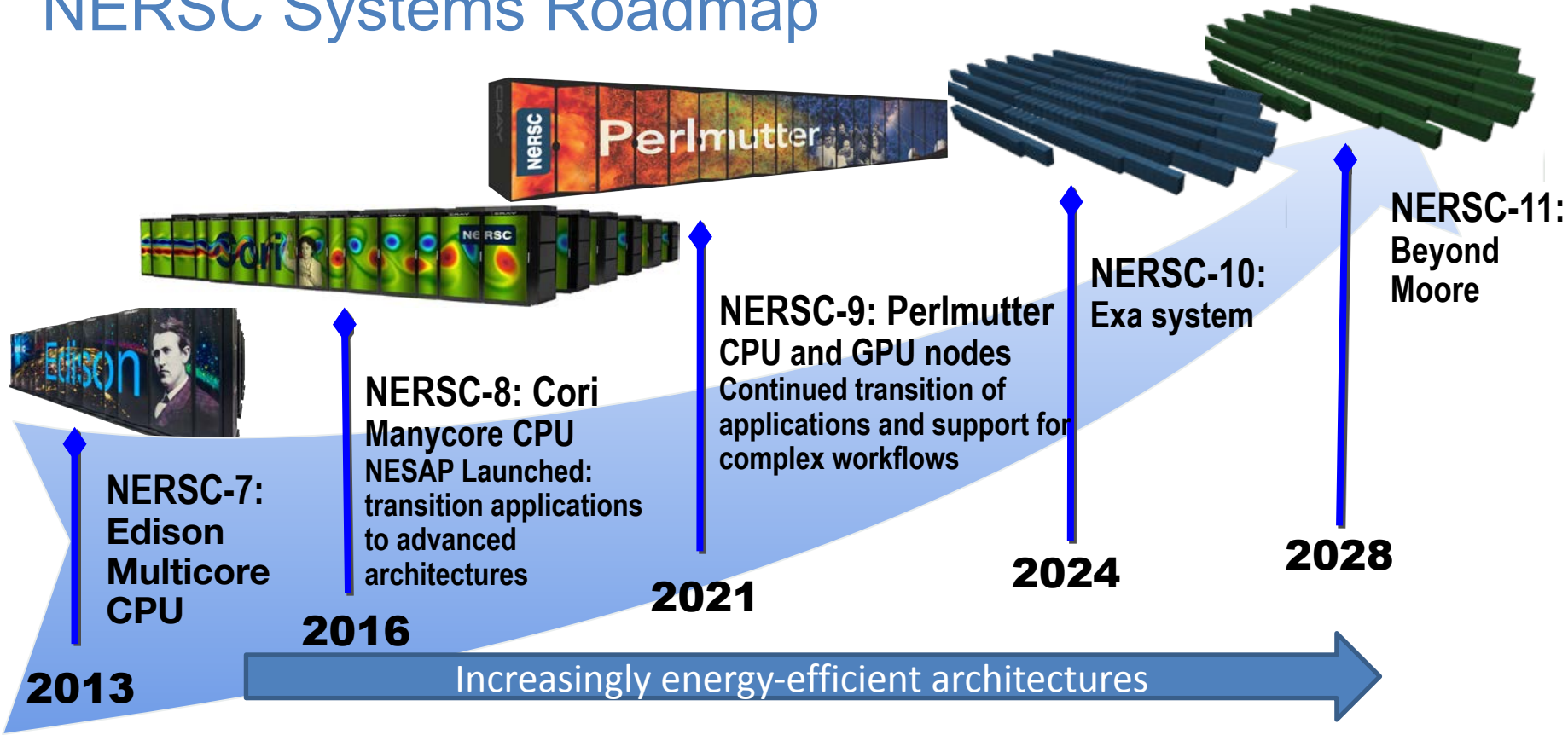
- NERSC deploys advanced HPC and data systems for the broad Office of Science community
- NERSC staff provide advanced application and system performance expertise to users
- Approximately 9,000 users and 900 projects
- Over 2,000 publications cite using NERSC resources per year
- Founded in 1974, focused 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

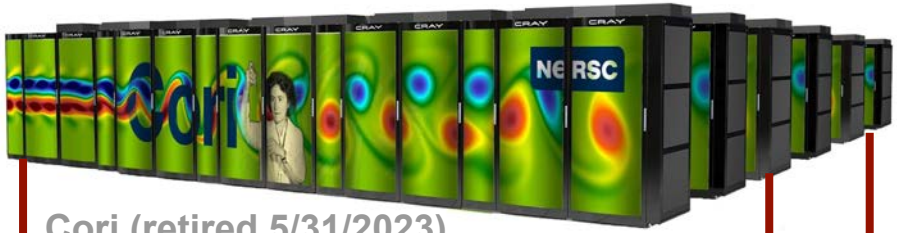




NERSC Systems Roadmap



NERSC Systems



Perlmutter

1,536 NVIDIA A100 accelerated nodes
4 A100 GPUs & 1 AMD 'Milan' CPU per node
384 TB (CPU) + 240 TB (GPU) memory
HPE Cray Slingshot high speed interconnect
World's 7th most powerful supercomputer
140 PF Peak
Pre-production system

5 TB/s

35 PB Scratch

Cori (retired 5/31/2023)

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
30 PF Peak

1.5 TB/s

2 PB Burst Buffer

700 GB/s

28 PB Scratch

50 GB/s

HPSS Tape Archive ~200 PB

DTNs, Spin, Gateways

ESnet ENERGY SCIENCES NETWORK

2 x 100 Gb/s SDN

Ethernet & IB Fabric

Science Friendly Security
Production Monitoring
Power Efficiency

LAN

100 GB/s

120 PB Common File System

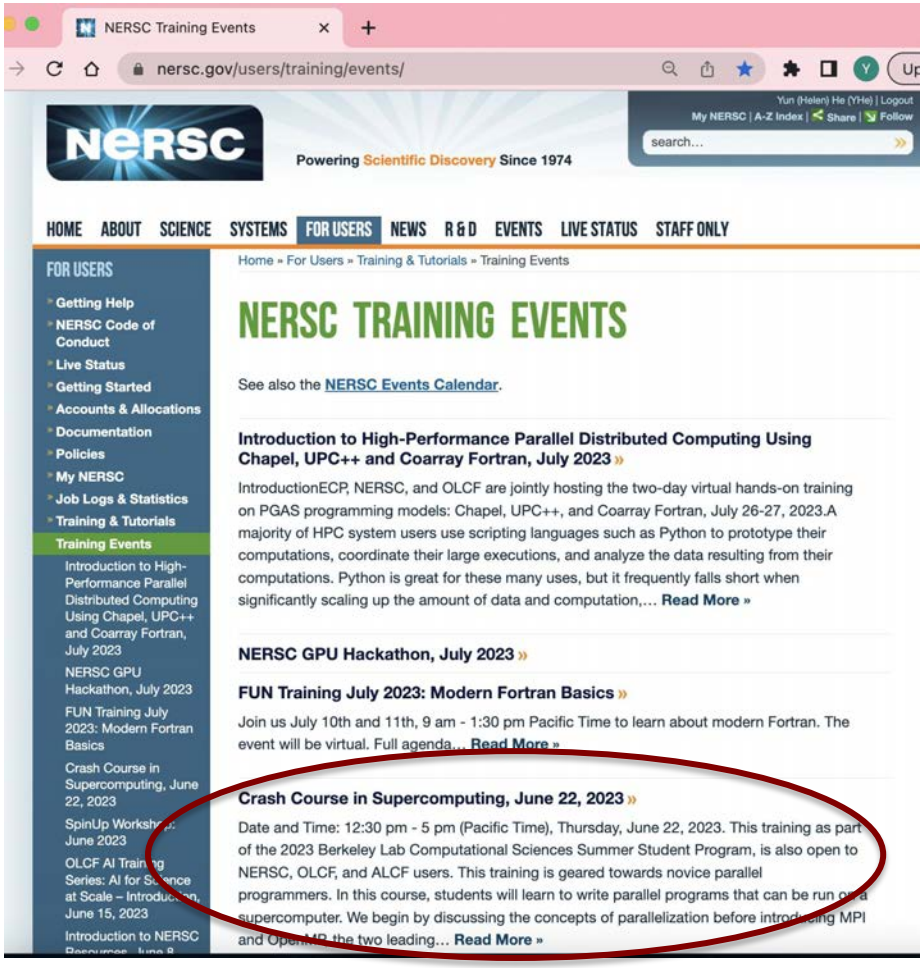
5 GB/s

275 TB /home

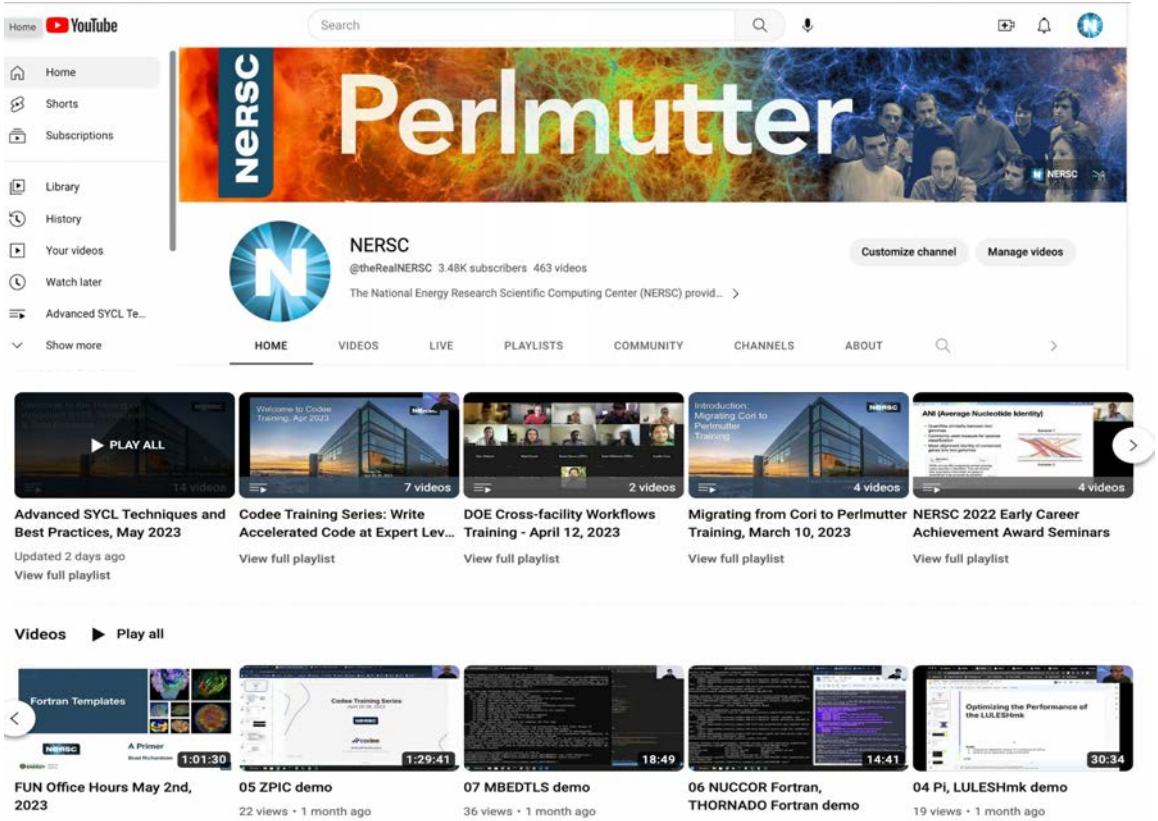
NERSC Online Resources

Classic NERSC Page

- https://www.nersc.gov
- Science, News, Publications
- Contact Us
- Live Status (MOTD) https://www.nersc.gov/live-status/motd/
- NUG (and Slack)
- Training Events https://www.nersc.gov/users/training/events/
  - New Users, Using Systems, GPUs, Programming Models, Performance Tools, Applications, Data Analytics, ML/DL, Workflows, and Services, …



NERSC YouTube Channel



https://www.youtube.com/c/NERSCTraining-HPC

Training sessions and other NERSC events presentations are archived on youtube, with professional captions

User Slack; User Appointments

https://www.nersc.gov/users/NUG/

https://docs.nersc.gov/getting-started/#appointments-with-nersc-user-support-staff

general This is a venue to discuss NERSC happenings with f... 1,546

Thursday, June 1st

Erik Palmer
Did you know ...

With Cori retired, job scripts with the constraints, `haswell` or `knl`, will no longer run.
To run on Perlmutter, if your Cori script has,

```
--constraint=haswell
```

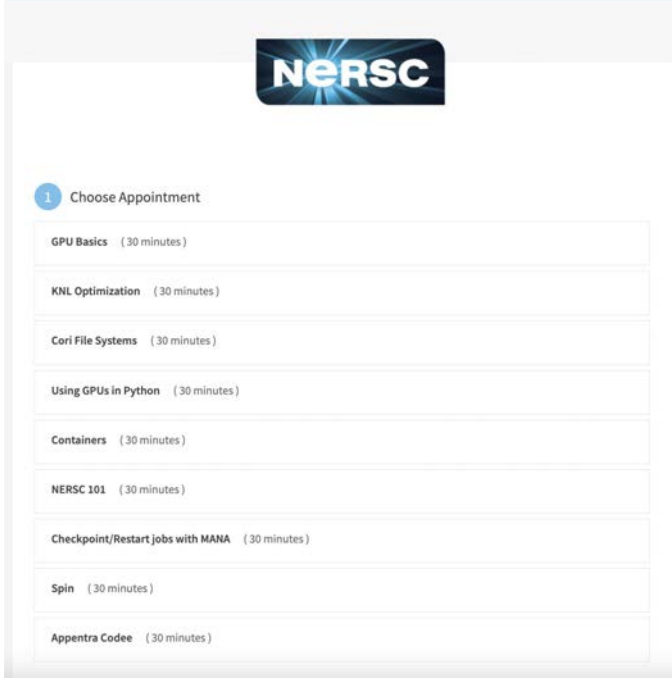
or
Show more
Posted in # tips-and-tricks | Jun 1st | View message

Zhe Feng 10:31 AM
Hi everyone, I have a quick question about file permissions on NERSC. For a given file/directory, we can only set 1 group owner right? A user must be belonging to that group to have access to the data (if the permission is set to group read only, not global read).
3 replies Last reply 2 days ago

Helen He 1:34 PM
@here Bring to your attention a NERSC training event next week:

- **Introduction to NERSC Resources Training, June 8**

NERSC is offering a training entitled "Introduction to NERSC Resources" on June 8. **This training**, offered through the 2023 Berkeley Lab Computing Sciences Summer Student program and open to NERSC users, **is aimed at novice users of NERSC resources**. Topics covered include: systems



NERSC Docs

Technical Documentations

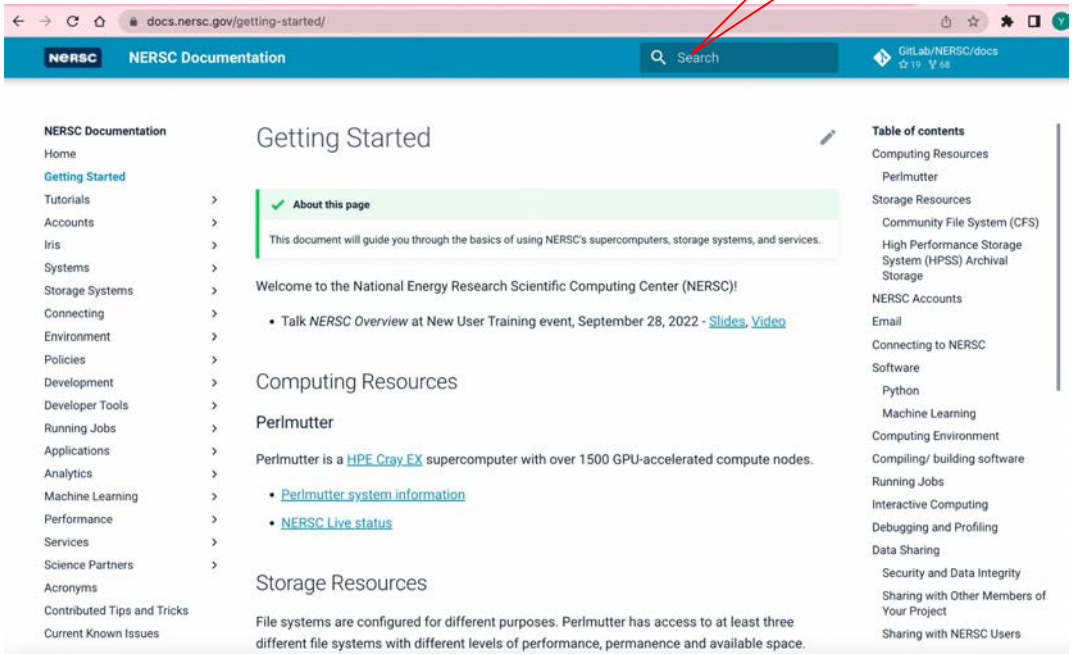
https://docs.nersc.gov

- Getting Started

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

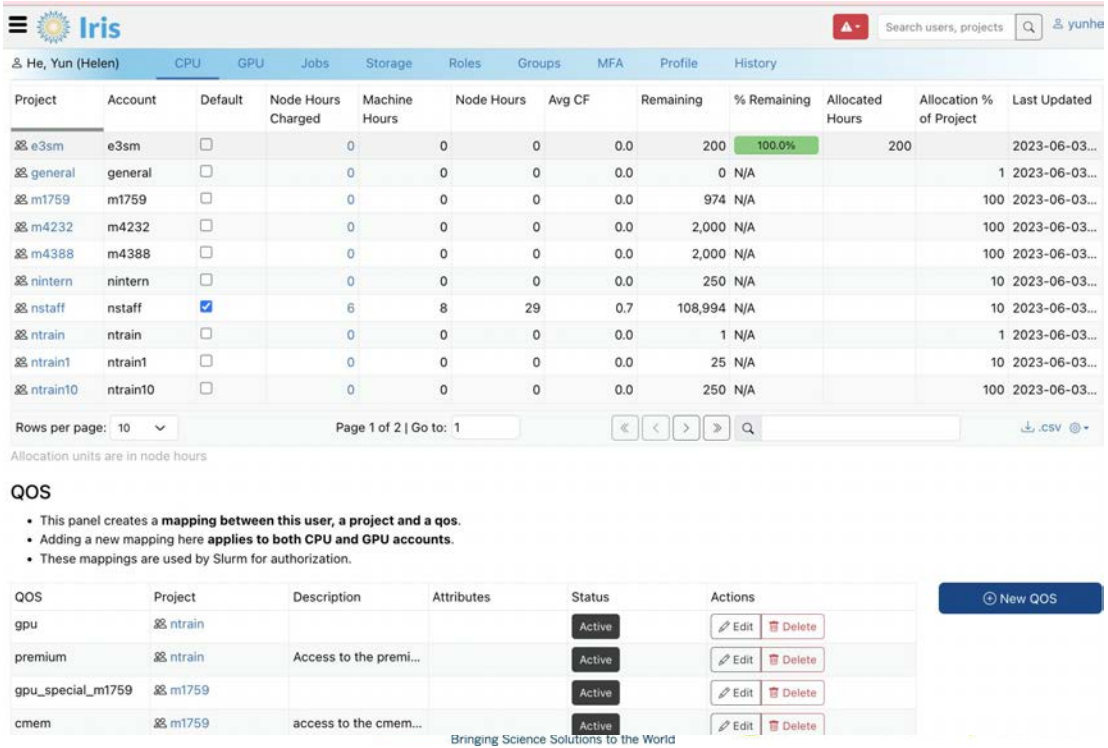
Home
Getting Started
Tutorials
Accounts
Iris
Systems
Storage Systems
Connecting
Environment
Policies
Development
Developer Tools
Running Jobs
Applications
Analytics
Machine Learning
Performance
Services
Science Partners
Acronyms
Contributed Tips and Tricks
Current Known Issues

search box



IRIS

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



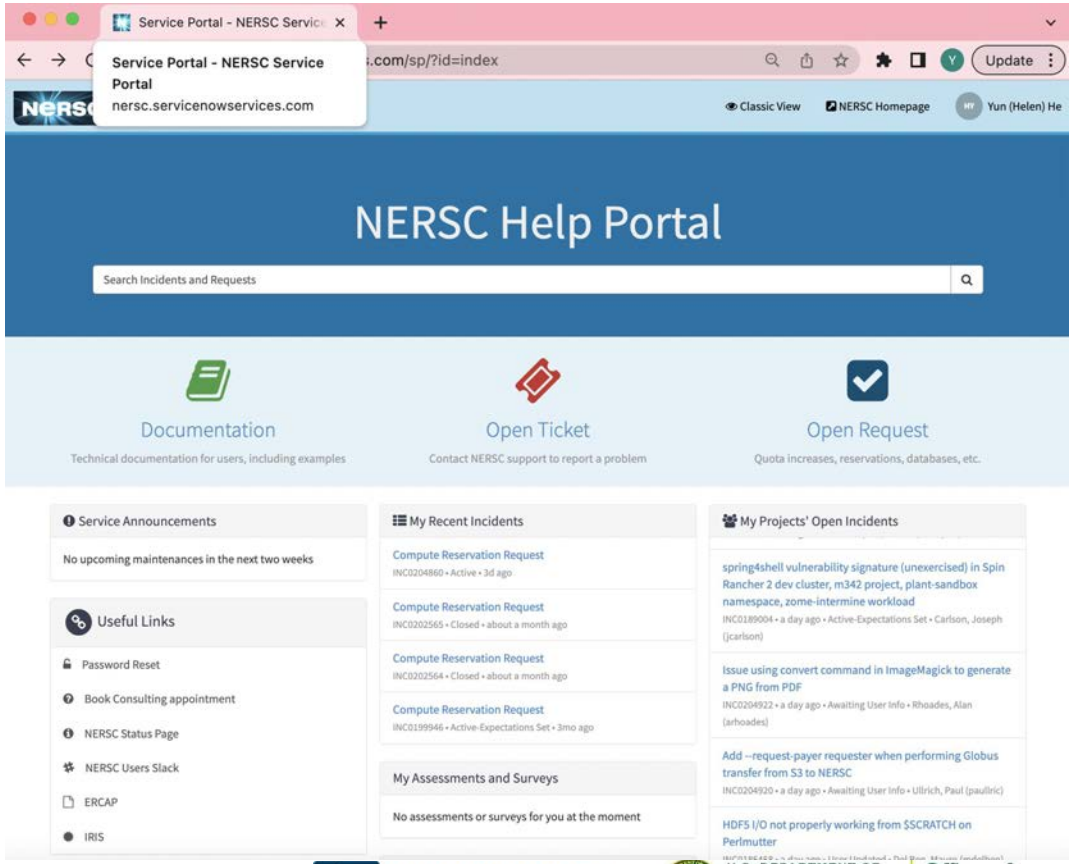
Project	Account	Default	Node Hours Charged	Machine Hours	Node Hours	Avg CF	Remaining	% Remaining	Allocated Hours	Allocation % of Project	Last Updated
e3sm	e3sm	☐	0	0	0	0.0	200	100.0%	200		2023-06-03...
general	general	☐	0	0	0	0.0	0	N/A		1	2023-06-03...
m1759	m1759	☐	0	0	0	0.0	974	N/A		100	2023-06-03...
m4232	m4232	☐	0	0	0	0.0	2,000	N/A		100	2023-06-03...
m4388	m4388	☐	0	0	0	0.0	2,000	N/A		100	2023-06-03...
nintern	nintern	☐	0	0	0	0.0	250	N/A		10	2023-06-03...
nstaff	nstaff	☑	6	8	29	0.7	108,994	N/A		10	2023-06-03...
ntrain	ntrain	☐	0	0	0	0.0	1	N/A		1	2023-06-03...
ntrain1	ntrain1	☐	0	0	0	0.0	25	N/A		10	2023-06-03...
ntrain10	ntrain10	☐	0	0	0	0.0	250	N/A		100	2023-06-03...

QOS	Project	Description	Attributes	Status	Actions
gpu	ntrain			Active	Edit Delete
premium	ntrain	Access to the premi...		Active	Edit Delete
gpu_special_m1759	m1759			Active	Edit Delete
cmem	m1759	access to the cmem...		Active	Edit Delete

Help Portal

https://help.nersc.gov

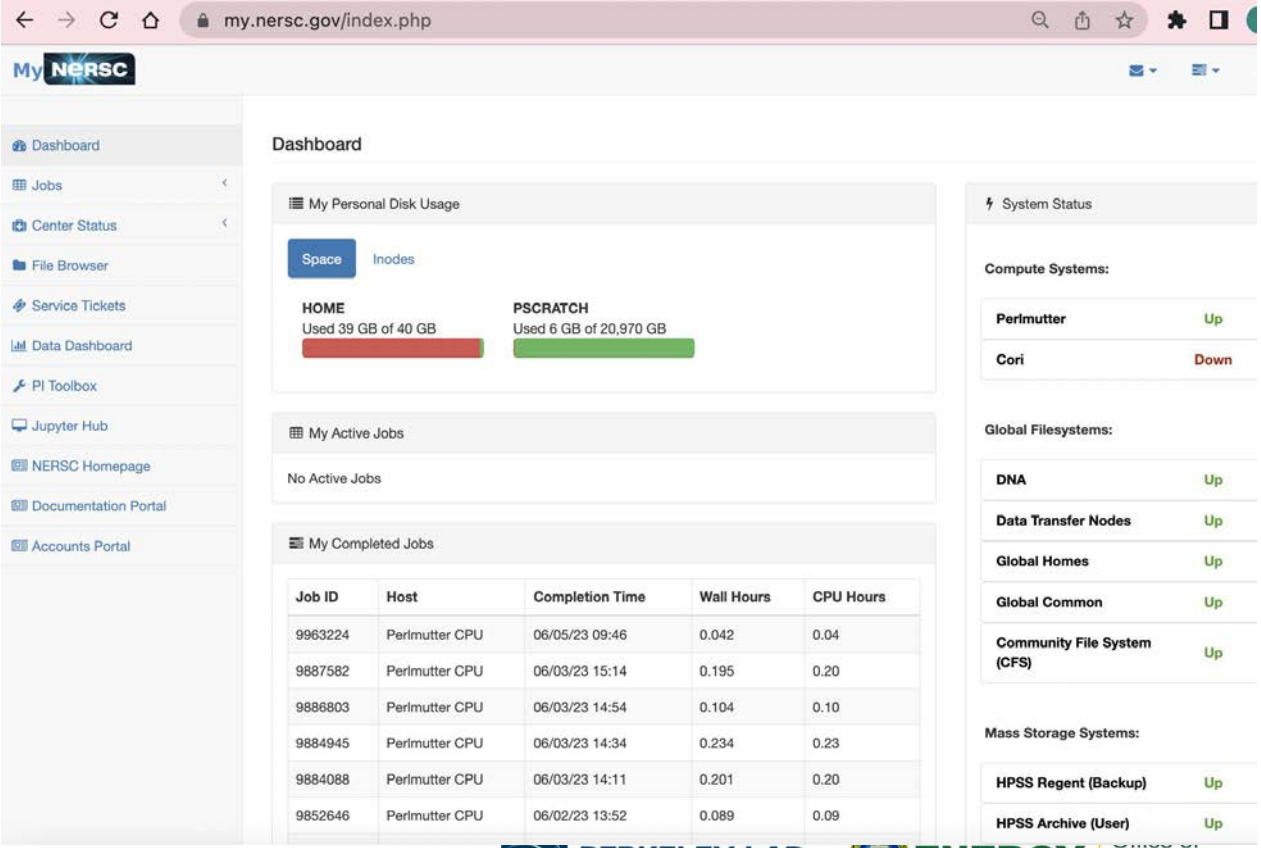
- Submit tickets (ask questions)
- All my tickets
- All my projects tickets
- Request forms:
  - Quota Increase
  - Reservations, …
- Book consulting appo
- NERSC user Slack
- Allocation (ERCAP) requests
- Iris



MyNERSC

https://my.nersc.gov

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



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

help.nersc.gov

jupyter.nersc.gov

www.nersc.gov

docs.nersc.gov

iris.nersc.gov

my.nersc.gov/index.php

My NERSC

Dashboard
Jobs
Center Status
File Browser
Service Tickets
Data Dashboard
PI Toolbox
Jupyter Hub
NERSC Homepage
Documentation Portal
Accounts Portal

Dashboard

My Personal Disk Usage

Space | Inodes

HOME
Used 39 GB of 40 GB

PSCRATCH
Used 6 GB of 20,970 GB

My Active Jobs

No Active Jobs

My Completed Jobs

Job ID	Host	Completion Time	Wall Hours	CPU Hours
9963224	Perlmutter CPU	06/05/23 09:46	0.042	0.04
9887582	Perlmutter CPU	06/03/23 15:14	0.195	0.20
9886803	Perlmutter CPU	06/03/23 14:54	0.104	0.10
9884945	Perlmutter CPU	06/03/23 14:34	0.234	0.23
9884088	Perlmutter CPU	06/03/23 14:11	0.201	0.20
9852646	Perlmutter CPU	06/02/23 13:52	0.089	0.09

System Status

Compute Systems:

Perlmutter	Up
Cori	Down

Global Filesystems:

DNA	Up
Data Transfer Nodes	Up
Global Homes	Up
Global Common	Up
Community File System (CFS)	Up

Mass Storage Systems:

HPSS Regent (Backup)	Up
HPSS Archive (User)	Up

my disk quota

is Perlmutter up?

my jobs

Perlmutter Documentation

https://docs.nersc.gov/systems/perlmutter

docs.nersc.gov/systems/perlmutter/

Calendar | HPC | NERSC | LBL | Google | Imported | Other Bookmarks

NERSC Using Perlmutter

Search

GitLab/NERSC/docs

- Iris
- Systems
  - Perlmutter
    - System
    - Interconnect
    - Running Jobs
    - Finding and using software
    - Timeline
  - Cori
  - Cori Large Memory
  - Data Transfer Nodes
- Storage Systems
- Connecting
- Environment
- Policies
- Development
- Developer Tools
- Running Jobs
- Applications
- Analytics
- Machine Learning
- Performance
  - Perlmutter Readiness

- Static compilation isn't officially supported by NERSC, but we have outlined some instructions under the static compilation section in the compiler wrappers documentation page.
- MPI/mpi4py users may notice a `mlx5` error that stems from spawning forks within an MPI rank, which is considered undefined/unsupported behavior.
- `PrgEnv-gnu` users when using a GPU enabled code (gcc and nvcc) you might have to load a compatible gcc version for the respective `cudatoolkit` installation. Please see our gcc compatibility section for additional details.
- Users may notice MKL-based CPU code runs more slowly. Please try `module load fast-mkl-amd`.

Preparing for Perlmutter

Please check the Transitioning Applications to Perlmutter webpage for a wealth of useful information on how to transition your applications for Perlmutter.

Compiling/Building Software

You can find information below on how to compile your code on Perlmutter:

Programming Environment & Cray Wrappers

Table of contents

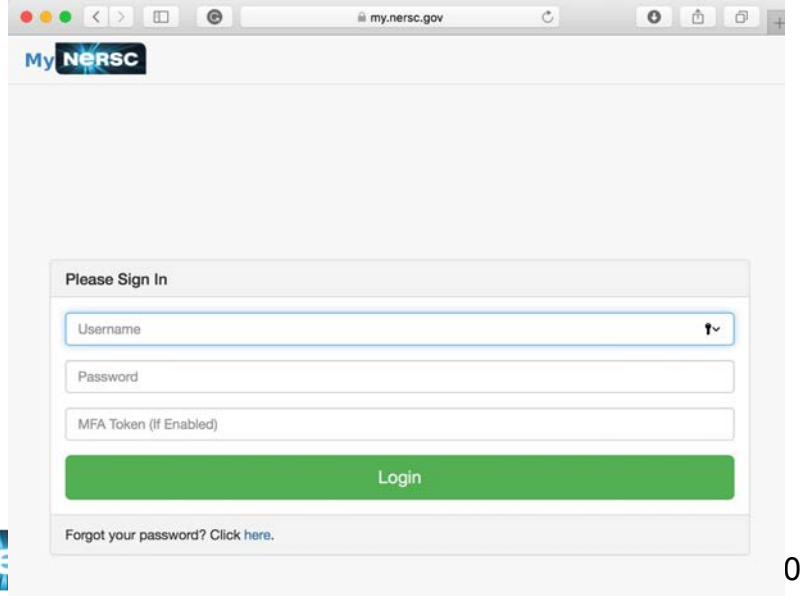
Connecting to NERSC

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 elvis@perlmutter.nersc.gov
…
Password + OTP:
elvis@perlmutter:login32:~>
```



You will login to one of the login nodes (40 on Perlmutter).

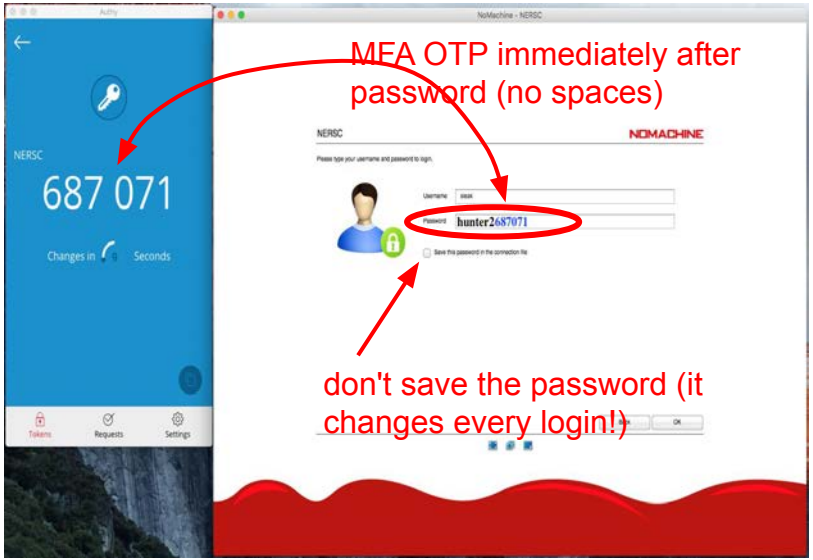
To allow X-forwarding to access visualization programs, use the "-Y" flag:

localhost% ssh -Y elvis@perlmutter.nersc.gov

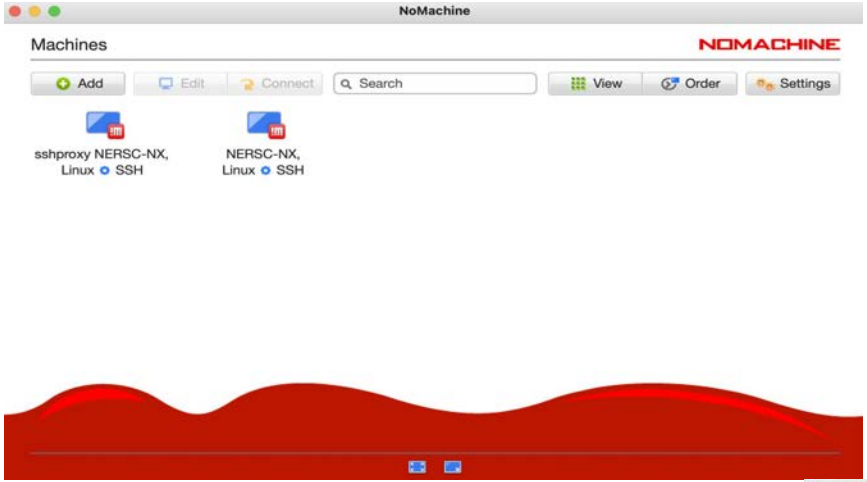
```
e/elvis> module load matlab
e/elvis> matlab
    <MATLAB starts up>
```

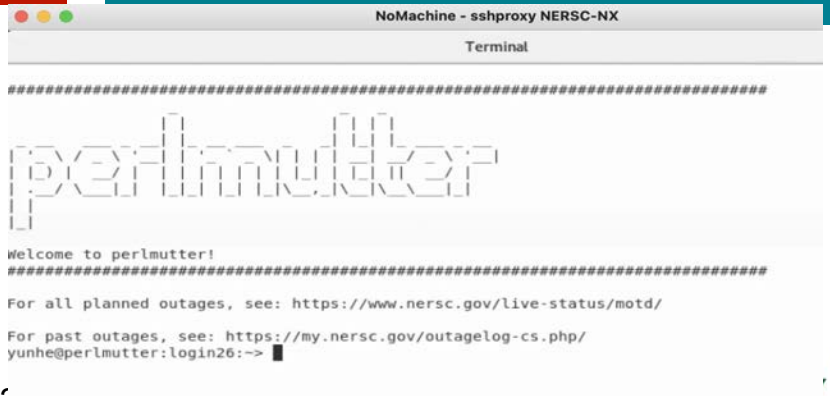
Connecting to NERSC: NX

- 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



NoMachine

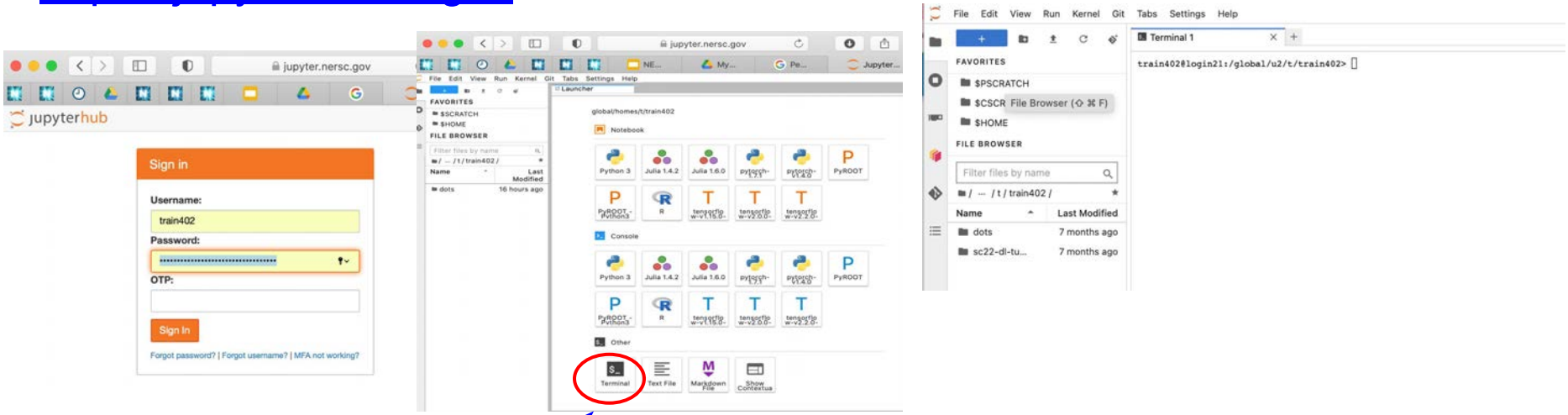




- Could also setup with sshproxy so only need to authenticate once per day

Terminal in Jupyter

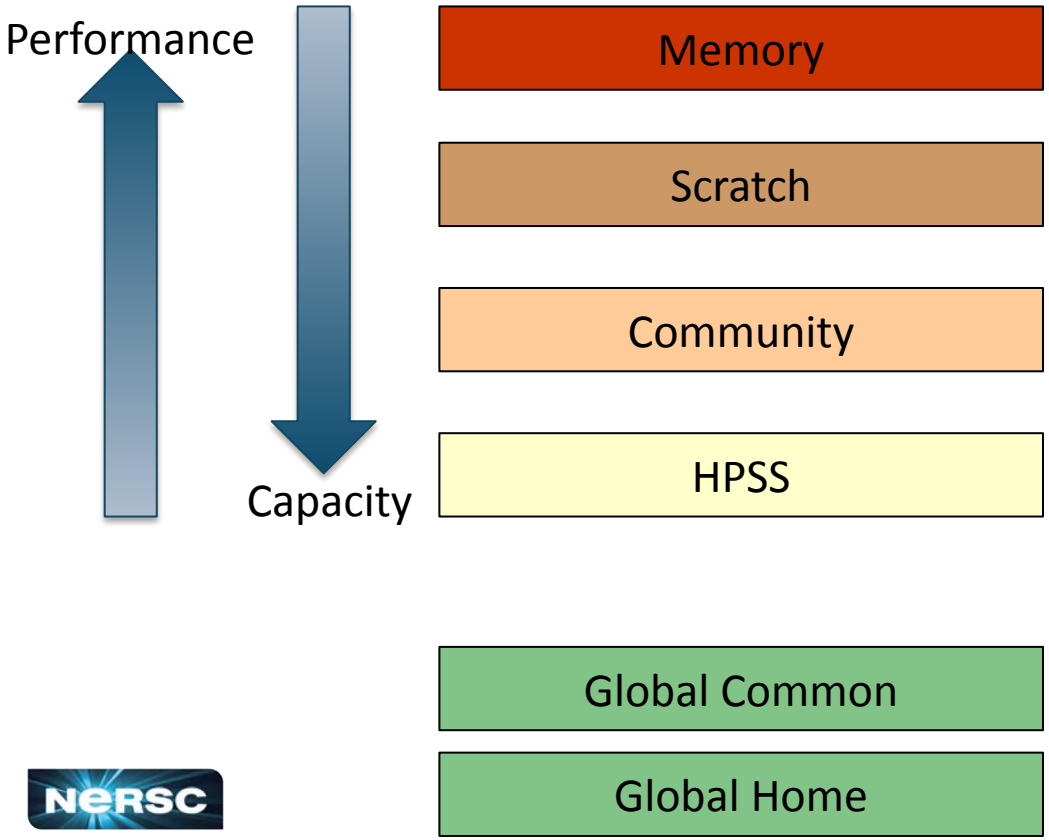
You can access Perlmutter from any web browser, via https://jupyter.nersc.gov



Terminal

File Systems and Data Management / Transfer

Simplified NERSC File Systems



35 PB (Perlmutter) Flash Scratch
- Lustre >5 TB/s
- temporarily (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 (8 weeks)
- **Perfect for staging data and performing computations**

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

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

`module`	
`list`	To list the modules in your environment
`spider <name>`	To list available modules with <name> as substring, and how to load
`load/unload ..`	To load or unload module
`swap .. ..`	To swap modules
`show/display ..`	To see what a module loads, what env a module sets
`whatis ..`	Display the module file information
`help ..`	General help: `$module help` Information about a module: `$ module help PrgEnv-cray`

Default Modules Loaded at Login (GPU Environment)

Modules Loaded by Default:

1) craype-x86-milan
2) libfabric/1.15.0.0
3) craype-network-ofi
4) perftools-base/22.06.0
5) xpmem/2.4.4-2.3_12.2__gff0e1d9.shasta
6) gcc/11.2.0
7) craype/2.7.16
8) cray-dsmml/0.2.2
9) cray-mpich/8.1.17
10) cray-libsci/21.08.1.2
11) PrgEnv-gnu/8.3.3
12) xalt/2.10.2
13) darshan/3.4.0
14) Nsight-Compute/2022.1.1
15) Nsight-Systems/2022.2.1
16) cudatoolkit/11.7
17) craype-accel-nvidia80
18) gpu/1.0

- CPU Architecture
- Default Programming Environment, Compiler, MPI, Scientific Libraries
- GPU Architecture, CUDA-Aware MPI, GPU Profilers

- CUDA-aware MPI is enabled by default
- Modules cudatoolkit, craype-accel-nvidia80, and gpu are loaded by default.
- gpu module also sets MPICH_GPU_SUPPORT_ENABLED to 1.

Default Modules for CPU-only Code

For CPU-only code we recommend:

```
module load cpu
```

1) craype-x86-milan
2) libfabric/1.15.0.0
3) craype-network-ofi
4) perftools-base/22.06.0
5) xpmem/2.4.4-2.3_12.2__gff0e1d9.shasta
6) gcc/11.2.0
7) craype/2.7.16
8) cray-dsmml/0.2.2
9) cray-mpich/8.1.17
10) cray-libsci/21.08.1.2
11) PrgEnv-gnu/8.3.3
12) xalt/2.10.2
13) darshan/3.4.0
14) cpu/1.0

- CPU Architecture
- Default Programming Environment, Compiler, MPI and Scientific Libraries
- Configured for CPU-only MPI

Software Environment

- Available compilers: GNU, Nvidia, CCE, (and Intel, in progress)
- It calls native compilers for each compiler (such as gfortran, gcc, g++, 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.

Building Sample Program on CPU

- module load cpu
- Using default GNU compiler

  ftn -o mytest mytest.f90 (MPI code)

  cc -fopenmp -o mytest mytest_hybrid.c (hybrid MPI/OpenMP code)
- Using Nvidia compiler

  module load PrgEnv-nvidia

  cc -o mytest mytest_code.c (MPI code)

  cc -mp -o mytest_hybrid mytest_hybrid.c (MPI/OpenMP hybrid code)

Perlmutter Supports Every GPU Programming Model

	Fortran/ C/C++	CUDA	OpenACC 2.x	OpenMP 5.x	CUDA Fortran	Kokkos / Raja	MPI	HIP	DPC++ / SYCL
NVIDIA									
CCE									
GNU									
LLVM									
Intel									

Vendor Supported

NERSC Supported in progress

Building CUDA Program on GPU

- module load gpu
- Using default GNU compiler

  CC -o mytest mytest.cpp
- using Nvidia compiler

  module load PrgEnv-nvidia

  CC -cuda -o mytest mytest.cpp

Building OpenMP Offload Program on GPU

- module load gpu
- using Nvidia compiler

```
module load PrgEnv-nvidia
ftn -mp=gpu -o mytest mytest.f90
cc -mp=gpu -o mytest mytest.c
CC -mp=gpu -o mytest mytest.cc
```

- Using CCE compiler

```
module load PrgEnv-cray
ftn -O3 -h omp -h noacc -o mytest mytest.f90
cc -Ofast -fopenmp -o mytest mytest.c
CC -Ofast -fopenmp -o mytest mytest.cc
```

Building Applications on Perlmutter

- More info on building for Perlmutter GPU
  - https://docs.nersc.gov/systems/perlmutter/#compilingbuilding-software
- More info on porting and optimizing for GPU on Perlmutter Readiness page
  - https://docs.nersc.gov/performance/readiness/
  - Basic GPU concepts and programming considerations, programming models, running jobs, machine learning applications, libraries, profiling tools, IO, case studies, …

Running Jobs

Jobs at NERSC

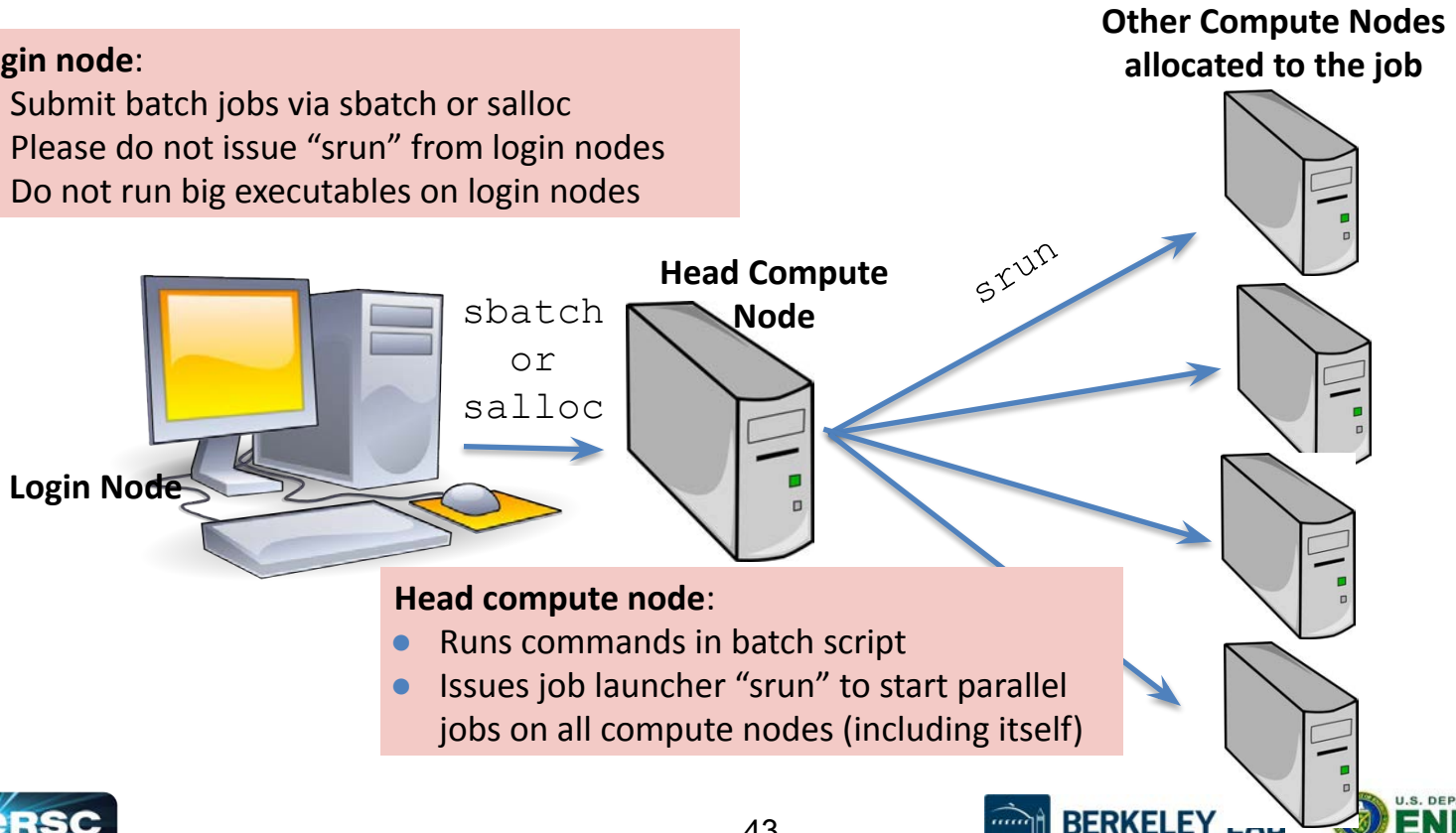
- Most are parallel jobs (10s to 100,000+ cores)
- Also a number of “serial” jobs
  - Typically “pleasantly parallel” simulation or data analysis
- Production runs execute in batch mode
- Our batch scheduler is SLURM
- Typical run times are a few to 10s of hours
  - Limits are necessary because of MTBF and the need to accommodate 9,000 users’ jobs

Login Nodes and Compute Nodes

- Login nodes (external)
  - Edit files, compile codes, submit batch jobs, etc.
  - Run short, serial utilities and applications
- Compute nodes
  - Execute your application
  - Dedicated resources for your job
  - Perlmutter has CPU and GPU compute nodes

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 cpu
##SBATCH -L SCRATCH
##SBATCH -J myjob
srun -n 64 ./helloWorld
```

To run via batch queue

% sbatch my_batch_script

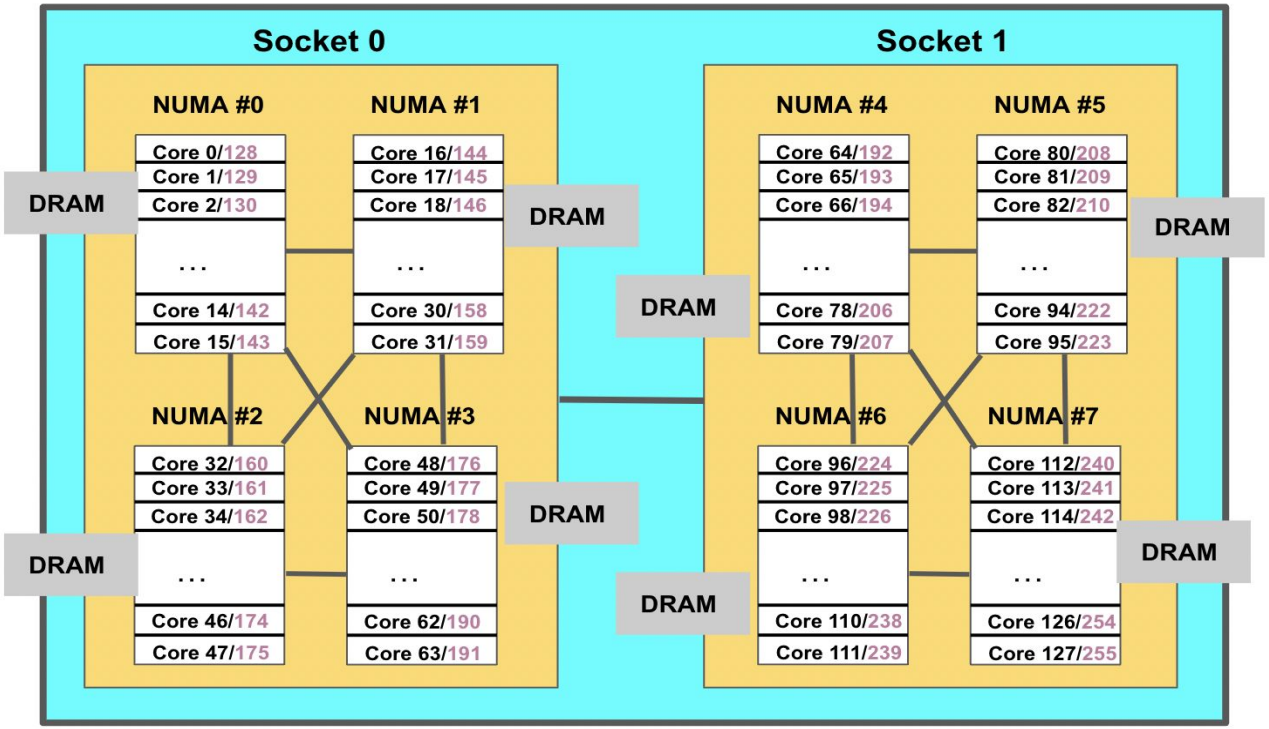
To run via interactive batch

% salloc -N 2 -q interactive -C cpu -t 10:00

<wait_for_session_prompt. Land on a compute node>

% srun -n 64 ./helloWorld

Perlmutter CPU Compute Node



- 2 sockets 4 NUMA domains/socket (8/node)
- 128 physical cores
- 256 logical cores
- Memory access on remote NUMA domains are slower

To obtain processor info:
Get on a compute node:
% salloc -N 1 -C …

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

Sample Perlmutter CPU Batch Script - MPI

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

srun -n 1280 -c 8 --cpu_bind=cores ./mycode.exe
```

32 MPI tasks per node in this example

- There are 256 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 Perlmutter CPU Batch Script - Hybrid MPI/OpenMP

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

export OMP_NUM_THREADS=8
export OMP_PROC_BIND=spread
export OMP_PLACES=threads

srun -n 320 -c 32 --cpu-bind=cores ./mycode.exe
```

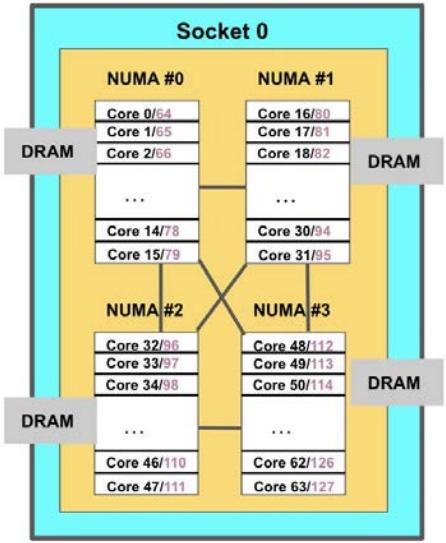
8 MPI tasks per node in this example

- 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 8 MPI tasks per nodel, set 256 logical CPUs / 8 =32 for ”-c”
  - “-c” value should be >= OMP_NUM_THREADS

CPU and GPU Compute Nodes Affinity

	Perlmutter CPU	CPU on Perlmutter GPU
Physical cores	128	64
Logical CPUs per physical core	2	2
Logical CPUs per node	256	128
NUMA domains	8	4
-c value for srun	2* floor(128/tpn)	2*floor(64/tpn)

CPU on Perlmutter GPU



tpn = Number of MPI tasks per node

Process / Thread / Memory Affinity

- Correct process, thread and memory affinity is critical for getting optimal performance on Perlmutter CPU and GPU
  - 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
- https://docs.nersc.gov/jobs/affinity/

Use salloc to Run Debug and Interactive Jobs

- You can run small parallel jobs interactively on dedicated nodes
- Debug
  - Max 8 nodes, up to 30 min

    % salloc -N 20 -q debug -C cpu -t 30:00
- Interactive (highly recommend to use this!!)
  - Instant allocation (get nodes in 6 min or reject)
  - Max 4 nodes, walltime 4 hrs

    % salloc -N 2 -q interactive -C cpu -t 2:00:00
  - More information
    - https://docs.nersc.gov/jobs/examples/#interactive
    - https://docs.nersc.gov/jobs/interactive/

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 128 (Perlmutter CPU) jobs from different users depending on their memory requirements

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

- Charged by a fraction of a node used
- https://docs.nersc.gov/jobs/examples/#shared
- Also available on Perlmutter GPU

Bundle Jobs

Multiple Jobs Sequentially:

```
#!/bin/bash
#SBATCH --qos=debug
#SBATCH --nodes=4
#SBATCH --time=10:00
#SBATCH --licenses=cfs,scratch
#SBATCH --constraint=cpu

# each srun uses 4 nodes
srun -n 128 -c 8 --cpu_bind=cores ./a.out
srun -n 64 -c 16 --cpu_bind=cores ./b.out
srun -n 32 -c 32 --cpu_bind=cores ./c.out
```

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

Multiple Jobs Simultaneously:

```
#!/bin/bash
#SBATCH --qos=debug
#SBATCH --nodes=8
#SBATCH --time=30:00
#SBATCH --licenses=scratch
#SBATCH --constraint=cpu

# 3 sruns combined use 8 nodes
srun -N 2 -n 176 -c 2 --cpu_bind=cores ./a.out &
srun -N 4 -n 432 -c 2 --cpu_bind=cores ./b.out &
srun -N 2 -n 160 -c 2 --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

```
perlmutter% sbatch job1
Submitted batch job 1655447

perlmutter% sbatch --dependency=afterok:165547 job2
or
perlmutter% sbatch --dependency=afterany:165547 job2
```

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

```
perlmutter% sbatch job1
submitted batch job 1655447

perlmutter% cat job2
#!/bin/bash
#SBATCH -q regular
#SBATCH -N 1
#SBATCH -t 1:30:00
#SBATCH -d afterok:1655447
#SBATCH -C cpu
srun -n 64 -c 4 –cpu-bind=cores ./a.out

perlmutter% 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 cpu

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

```
perlmutter% module load parallel

perlmutter% seq 1 5 | parallel -j 2 'echo
"Hello world {}!"; sleep 10; date'
Hello world 1!
Wed 07 Jun 2023 10:22:11 PM PDT
Hello world 2!
Wed 07 Jun 2023 10:22:11 PM PDT
Hello world 3!
Wed 07 Jun 2023 10:22:21 PM PDT
Hello world 4!
Wed 07 Jun 2023 10:22:21 PM PDT
Hello world 5!
Wed 07 Jun 2023 10:22:31 PM PDT
```

- 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/

Sample GPU Job Script

```
#!/bin/bash
#SBATCH --account=mxxx
#SBATCH --qos=regular
#SBATCH --nodes=2
#SBATCH --time=60
#SBATCH --constraint=gpu
#SBATCH --job-name=myjob
#SBATCH --ntasks-per-node=64
#SBATCH --cpus-per-task=2
#SBATCH --gpus-per-node=4

export OMP_NUM_THREADS=1
srun -n 128 --cpu-bind=cores --gpu-bind=closest <executable>
```

```
c = 2*floor(64/tpn)

Where:
    tpn = ntasks-per-node
```

- By default all processes will have access to all GPUs.
- A round robin assignment does not guarantee affinity.
- To guarantee that closest GPU is assigned: -gpus-bind=closest
- To bind ranks to individual cores: -cpu-bind=cores

1 Node, 4 Tasks, 4 GPUs

1 GPU visible to each task¶

```
#!/bin/bash
#SBATCH -A ntrain3
#SBATCH -C gpu
#SBATCH -q regular
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH --ntasks-per-node=4
#SBATCH -c 32
#SBATCH --gpus-per-task=1
export SLURM_CPU_BIND="cores"
srun ./gpus_for_tasks
```

Default for –gpus-per-task=1 is 1 task only see 1 GPU

4 GPUs visible to each task¶

```
#!/bin/bash
#SBATCH -A ntrain3
#SBATCH -C gpu
#SBATCH -q debug
#SBATCH -t 10:00
#SBATCH -N 1
#SBATCH --ntasks-per-node=4
#SBATCH -c 32
#SBATCH --gpus-per-task=1
#SBATCH --gpu-bind=none
export SLURM_CPU_BIND="cores"
srun ./gpus_for_tasks
```

Default for –gpus-per-task=1 and –gpu-bind-none is each task sees all GPU

Perlmutter CPU Queue Policy (as of June 2023)

QOS	Max nodes	Max time (hrs)	Submit limit	Run limit	Priority	QOS Factor
regular	-	12	5000	-	medium	1
interactive	4	4	2	2	high	1
jupyter	4	6	1	1	high	1
debug	8	0.5	5	2	medium	1
shared[3]	0.5	12	5000	-	medium	1
preempt	128	24 (preemptible after two hours)	5000	-	medium	0.5
overrun	-	12	5000	-	very low	0
realtime	custom	custom	custom	custom	very high	1

Perlmutter GPU Queue Policy (as of June 2023)

QOS	Max nodes	Max time (hrs)	Submit limit	Run limit	Priority	QOS Factor
regular	-	12	5000	-	medium	1
interactive	4	4	2	2	high	1
jupyter	4	6	1	1	high	1
debug	8	0.5	5	2	medium	1
shared[3]	0.5	12	5000	-	medium	1
preempt	128	24 (preemptible after two hours)	5000	-	medium	0.25
overrun	-	12	5000	-	very low	0
realtime	custom	custom	custom	custom	very high	1

NERSC

U.S. DEPARTMENT OF ENERGY | Office of Science

NERSC Job Script Generator

my.nersc.gov/script_generator.php

My NERSC

- Dashboard
- Jobs
  - Jobscript Generator
  - Completed Jobs
  - Perlmutter Queues
  - Queue Backlog
- Center Status
- File Browser
- Service Tickets
- Data Dashboard
- PI Toolbox
- Jupyter Hub
- NERSC Homepage
- Documentation Portal
- Accounts Portal

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.

Perlmutter - CPU

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.

Quality of Service

Select the QoS you request for your job.

regular

Wallclock Time

```
#!/bin/bash
#SBATCH -N 4
#SBATCH -C cpu
#SBATCH -q regular
#SBATCH -t 01:30:00

#OpenMP settings:
export OMP_NUM_THREADS=4
export OMP_PLACES=threads
export OMP_PROC_BIND=spread


#run the application:
srun -n 32 -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://www.nersc.gov/users/live-status/ □ Queue Look
  - https://iris.nersc.gov the “Jobs” tab

Data Analytics Software and Services

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, MATLAB, TensorFlow, K, PyTorch
Data Visualization	VisIt, ParaView

Data Analytic Software Services

- Globus Online
- Science Gateways
- Databases
- Shifter / Podman
- 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), NERSC Perlmutter, NERSC HPSS, etc.
  - Other Center has endpoints, such as OLCF DTN
  - Setup Globus Connect Personal to ease transfer between local system (such as laptop) and NERSC systems

Globus File Transfer Example

app.globus.org

My Driv... NERSC_... Downlo... File Ma... Downlo... Downlo...

File Manager

Panels

FILE MANAGER
BOOKMARKS
ACTIVITY
ENDPOINTS
GROUPS
CONSOLE
FLOWS
ACCOUNT
LOGOUT

Collection NERSC DTN

OLCF DTN

Path /~/GPU_Feb2020/

Start

Transfer & Sync Options

Start

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

Share

Transfer or Sync to...

New Folder

Rename

Delete Selected

Download

Open

Upload

Get Link

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.

Continue

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:

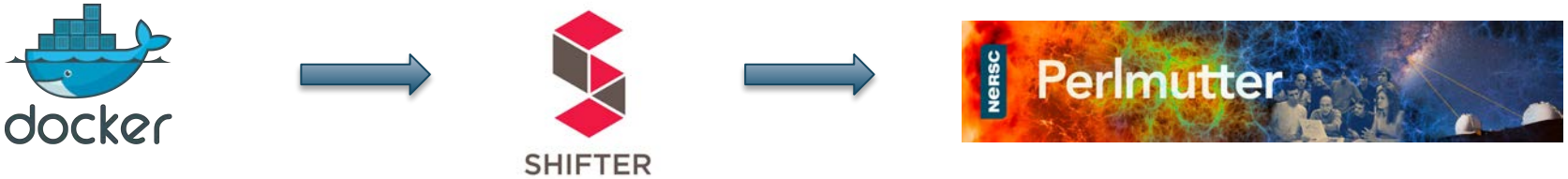
      http://portal.nersc.gov/cfs/<your_project>
  - Each repo has a /project space, can publish as above
- Special Science Gateways can be created.
  - Sophisticated ones can be made with SPIN

    https://docs.nersc.gov/services/spin/

    https://www.nersc.gov/users/training/spin/ (SPIN workshop required)
  - Details at: https://docs.nersc.gov/services/science-gateways/

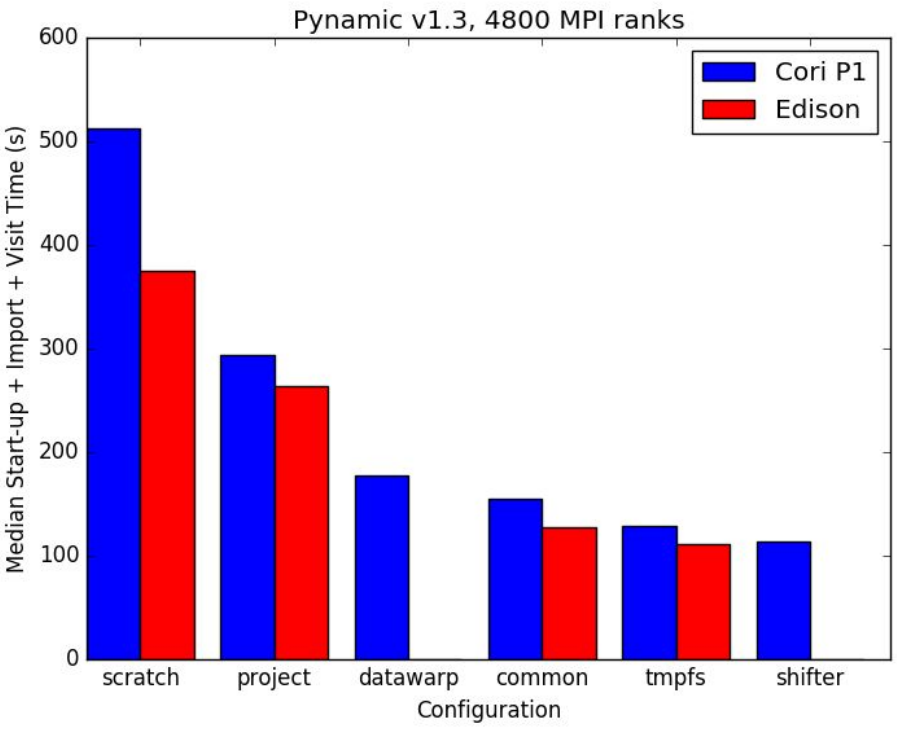
Shifter

- 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



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

Shifter Accelerates Python Applications



- Shifter is especially helpful for python applications
- A large number of shared libraries needed on compute nodes before execution

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
```

Try this: Podman

podman

- Podman (Pod manager) is an Open Container Initiative compliant container framework under active development by Red Hat
- Free and open source
- Usable anywhere (including your laptop), not just NERSC
- Can provide *rootless containers,* which give users the ability to run as root within their image while still maintaining security
- **Will allow users to build images on Perlmutter login nodes**
- Performance in most cases should be similar to what is currently possible with Shifter (i.e. **it's fast!**)
- https://docs.nersc.gov/development/podman-hpc/overview/
-

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

Python

- Avoid running "`conda init`" which will hardcode conda initialization in your shell startup file ($HOME/.bashrc)
- Do not use /usr/bin/python, instead:

  module load python

  which already includes basic packages: numpy, scipy, mpi4py
- Guide to use Python on Perlmutter:
  - https://docs.nersc.gov/development/languages/python/using-python-perlmutter

Other options for using Python at NERSC

Create a custom conda environment:

```
perlmutter> module load python
perlmutter> conda create --name myenv --yes python=3.10
perlmutter> conda activate myenv
(myenv) perlmutter> python
Python 3.10.4 (main, Mar 31 2022, 08:41:55) [GCC 7.5.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Use Python inside a Shifter container:

```
perlmutter> shifter --image=docker:library/python:latest python
Python 3.10.7 (main, Sep 13 2022, 14:31:33) [GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

https://docs.nersc.gov/development/languages/python/nersc-python/

Building and using mpi4py

- mpi4py provides a Python interface to MPI
- mpi4py is available via **`module load python`**
- This mpi4py is CUDA-aware (can communicate GPU objects)
- To build your own CUDA-aware mpi4py, follow this recipe:

```
perlmutter> module load PrgEnv-gnu cudatoolkit python
perlmutter> conda create -n cudaaware python=3.9 -y
perlmutter> conda activate cudaaware
perlmutter> MPICC="cc -target-accel=nvidia80 -shared" pip install
--force-reinstall --no-cache-dir --no-binary=mpi4py mpi4py
```

- Be aware that with any CUDA-aware mpi4py, you must have **`cudatoolkit`** loaded, even for code that does not use the GPU

Getting started with GPUs in Python

- NumPy and SciPy do not utilize GPUs out of the box

- There are many Python GPU frameworks out there:
  - “drop in” replacements for numpy, scipy, pandas, scikit-learn, etc
    - **CuPy**, **RAPIDS**
  - “machine learning” libraries that also support general GPU computing
    - **PyTorch**, **TensorFlow, JAX**
  - “I want to write my own GPU kernels”
    - **Numba, PyOpenCL, PyCUDA, CUDA Python**
  - multi-node / distributed memory:
    - **mpi4py+X, dask, cuNumeric**

Getting started with GPUs in Python (CuPy)

```
> module load python
> conda create -y --name cupy-demo python=3.9 numpy scipy
> conda activate cupy-demo
> pip install cupy-cuda11X
> python
>>> import cupy as cp
>>> print(cp.array([1, 2, 3]))
[1 2 3]
```

Note: cudatoolkit module is loaded by default
Current default version is cudatoolkit/11.7

Check your package documentation to see cudatoolkit compatibility requirements

See documentation at https://docs.nersc.gov/development/languages/python/using-python-perlmutter/
or open a ticket at https://help.nersc.gov/

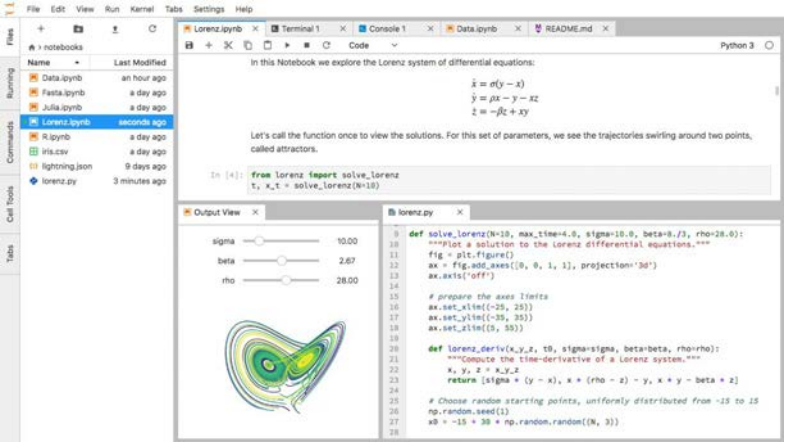
What is Jupyter?

Interactive open-source web application

Allows you to create and share documents, "notebooks," containing:

Live code
Equations
Visualizations
Narrative text
Interactive widgets

https://docs.nersc.gov/services/jupyter/



Things you can use Jupyter notebooks for:

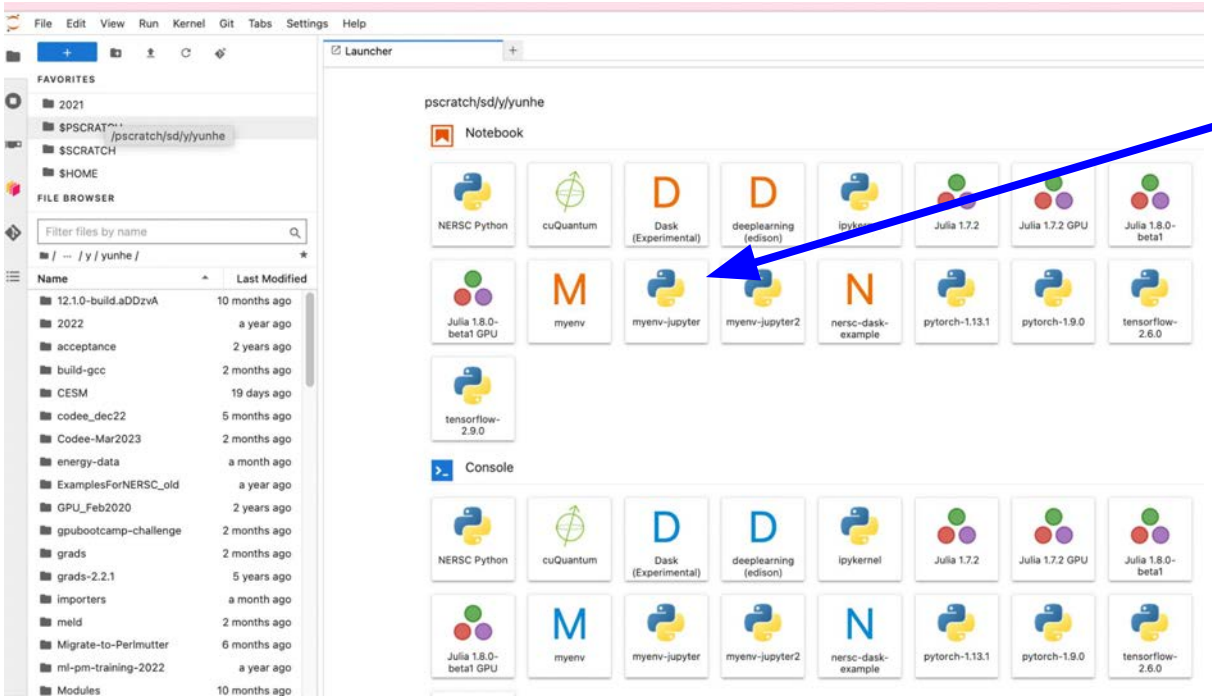
Data cleaning and data transformation
Numerical simulation
Statistical modeling
Data visualization
Machine learning
Workflows and analytics frameworks
Training and Tutorials

Available Notebook Servers

jupyterhub Home Token Services Documentation train402

	Shared CPU Node	Exclusive CPU Node	Exclusive GPU Node	Configurable Job
Perlmutter	start	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.

Available Jupyter Kernels



Your own custom kernels

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

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=3.9 ipykernel <more-packages-to-install>
$ conda activate myenv
(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

edit:
$HOME/.local/share/jupyter/kernels/myenv-jupyter/kernel.json

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

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

Meanwhile, in jupyter-helper.sh:

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

NERSC Deep Learning Software Stack Overview

https://docs.nersc.gov/machinelearning/

Frameworks:

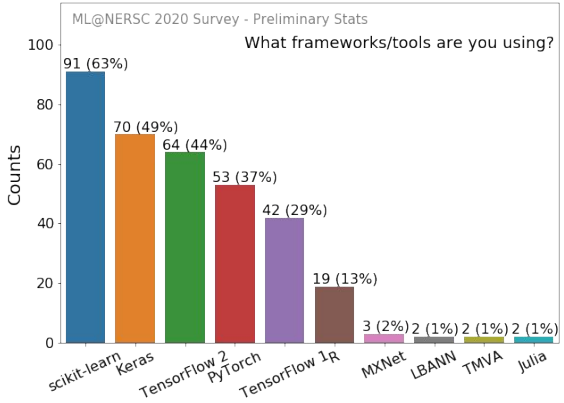
TensorFlow Keras PyTorch

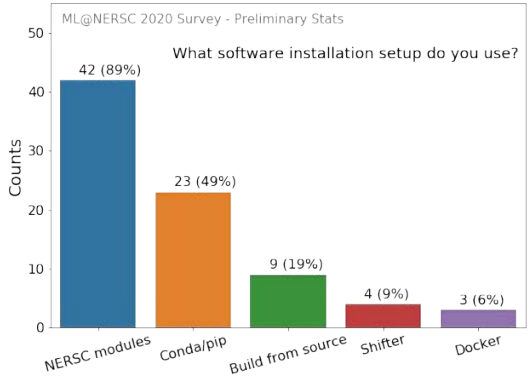
Distributed training libraries:

- 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/<version>
  - module load pytorch/<version>
- You can install your own packages on top to customize
  - pip install --force-reinstall --no-cache-dir --user MY-PACKAGE
- Or you can create your conda environments from scratch
  - conda create -n my-env MY-PACKAGES
- We also have pre-installed Jupyter kernels

Containerized DL: using Shifter on Perlmutter

To see images currently available:

```
shifterimg images | grep pytorch
```

To pull desired docker images onto Perlmutter:

```
shifterimg pull <dockerhub_image_tag>
```

To use interactively:

```
shifter --module gpu --image=nvcr.io/nvidia/pytorch:22.05-py3
```

Use Slurm image shifter options for best performance in batch jobs:

```
#SBATCH --image=nersc/pytorch:ngc-22.05_v1
srun shifter python my_python_script.py
```

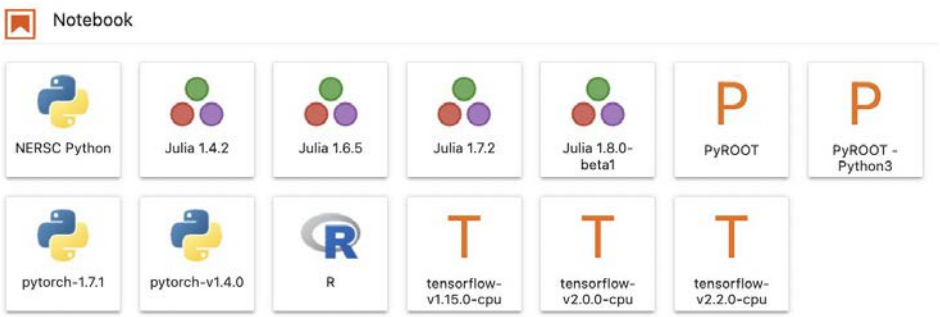
Jupyter for Deep Learning

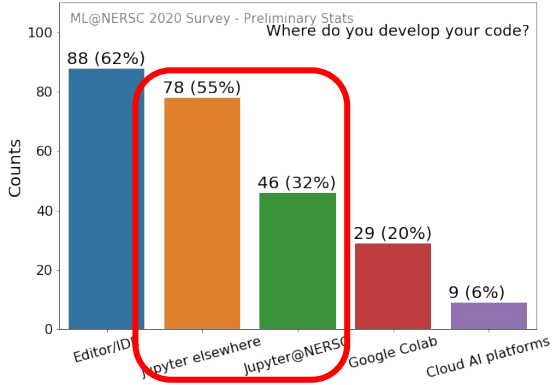
JupyterHub service provides a rich, interactive notebook ecosystem

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

Users can run their deep learning workloads

- Using our pre-installed DL software kernels on dedicated Perlmutter GPU nodes
- Using user custom kernels





Hands-on Exercises

Compiling and Running Jobs on Perlmutter

- % ssh <user>@perlmutter.nersc.gov (or ssh <user>@saul.nersc.gov)
- % cd $SCRATCH
- % git clone https://github.com/NERSC/intro-NERSC-resources.git
- % cd intro-NERSC-resources

CPU Examples:

- 01-hello: build and run basic MPI program on CPU

- 02-matrix: build and run a hybrid MPI/OpenMP matrix multiply code on CPU

- 03-xthi: a hybrid MPI/OpenMP code, mainly on CPU affinity settings

GPU Examples:

- 04-pi_targ: build and run an OpenMP target offload program on GPU

- 05-gpus_for_tasks: build and run a CUDA code on GPU, and gpu affinity settings

Using Compute Node Reservations

- Existing NERSC users are added to “ntrain3” project
- Perlmutter node reservations available from 2-3:30 pm today
- User reservations with --reservation=xxx -A ntrain3, where
  - xxx is “intro_cpu” or “intro_gpu”

Thank You