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/4h8ss95k
- Slides and videos will be available on the Training Event page and CSASP CS Summer Program page
  - https://www.nersc.gov/users/training/events/introduction-to-nersc-resources-jun2022/
- 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 "e1wR"
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 Cori**
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 8,000 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
NERSC Systems Roadmap

NERSC-7: Edison Multicore CPU
NERSC-8: Cori Manycore CPU
NESAP Launched: transition applications to advanced architectures

NERSC-8: Cori
Multicore CPU
NERSC-9: CPU and GPU nodes
Continued transition of applications and support for complex workflows

NERSC-9: Exa system
2021
2024
2028

NERSC-10: Exa system
NERSC-11: Beyond Moore

2016
2021
2028

Increasingly energy-efficient architectures

2013
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
- 30 PF Peak

Cray XC40 / Aries Dragonfly interconnect
- 28 PB Burst Buffer
- 700 GB/s

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

HPE Cray Slingshot high speed interconnect
- 1.5 TB/s
- 700 GB/s

HPSS Tape Archive
- ~200 PB

Ethernet & IB Fabric
- Science Friendly Security
- Production Monitoring
- Power Efficiency
- LAN

ESnet
- 2 x 100 Gb/s
- SDN

NERSC Systems

NERSC Systems

DTNs, Spin, Gateways
- 5 TB/s
- 35 PB Scratch

LAN
- 100 GB/s
- 5 GB/s

Common File System
- 275 TB
- /home

2 PB
- Burst Buffer
- Scratch

120 PB
- Common File System

NERSC Systems

HPSS Tape Archive
- ~200 PB

Production Monitoring

Power Efficiency

Science Friendly Security

Ethernet & IB Fabric

ESnet
- 2 x 100 Gb/s
- SDN
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/
Training sessions and other NERSC events presentations are archived on youtube, with professional captions.

https://www.youtube.com/c/NERSC Training-HPC
User Slack; User Appointments

Lincoln Bryant  Today at 10:11 AM
Hi folks. I’m trying to figure out what filesystems are attached to
perlmutter and also accessible via Globus. On Cori we used
/global/cscratch1 for our job input/output via the NERSC Cori
globus endpoint. Can anyone point me to the equivalent for
Perlmutter? Maybe I missed something blindingly obvious in the
docs..

Wileam Phan (LBNL)  5 minutes ago
Use the SPSCRATCH environment variable, according to
https://docs.nersc.gov/filesystems/perlmutter-scratch/

Jean Sexton  2 minutes ago
I have not yet seen pscratch accessible from globus, some people
use the CFS project directories, since those are visible from
cori/dtn/globus/perlmutter
https://docs.nersc.gov/systems/perlmutter/#transferring-data-to-
from-perlmutter-scratch

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

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

...
Tree View of NERSC Docs Topics

https://gitlab.com/NERSC/nersc.gitlab.io/-/blob/main/mkdocs.yml
IRIS

- IRIS: NERSC Account Management and Reporting: https://iris.nersc.gov
  - Account info
  - Change password
  - Change contact info
  - SSH Keys, MFA
  - Check usage info
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
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 disk quota
is cori up?
my jobs
Cori GPU Documentation

https://docs-dev.nersc.gov

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

https://www.nersc.gov/users/training/events/

- Using Perlmutter, Jan 2022
- Nvidia HPC SDK, Jan 2022
- Nvidia Performance Tools, Feb 2022
- Codee Training, Apr 2022
- Coding for GPUs with Standard C++, Apr 2022
- Coding for GPUs with Standard Fortran, May 2022
- Programming with SYCL, Mar 2022
- LLVM/OpenMP Ecosystem, May 2022
- OpenMP Offload, Sept 2021
- 3-part OpenACC training series, Apr-Jun, 2020
- 9-part CUDA training series, Jan 2020 - Sept 2021
- GPU for Science, Jul 2020
- Data Analytics in Python on GPUs with NVIDIA RAPIDS, Apr 2020
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@cori.nersc.gov
...
Login connection to host cori01:
Password + OTP:

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 -Y elvis@ cori.nersc.gov
```
el/elvis> module load matlab
e/elvis> matlab

<MATLAB starts up>
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](https://docs.nersc.gov/connect/nx)
  - Works on Window/Mac/Linux.

MFA OTP immediately after password (no spaces)
don't save the password (it changes every login!)
Terminal in Jupyter

You can access Cori from any web browser, via [https://jupyter.nersc.gov](https://jupyter.nersc.gov)
File Systems and Data Management / Transfer
Simplified NERSC File Systems

35 PB (Perlmutter) Flash Scratch
- Lustre >5 TB/s
- temporarily (purge)

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

Performance

Capacity
Global File Systems

<table>
<thead>
<tr>
<th>Global Home</th>
<th>Community File System (CFS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Permanent, relatively small</td>
<td>● Permanent, larger storage</td>
</tr>
<tr>
<td>storage</td>
<td>● Mounted on all platforms</td>
</tr>
<tr>
<td>● Mounted on all platforms</td>
<td>● Medium performance for parallel jobs</td>
</tr>
<tr>
<td>● NOT tuned to perform well for</td>
<td>● Quota can be changed</td>
</tr>
<tr>
<td>parallel jobs</td>
<td>● Snapshot backups (7-day history)</td>
</tr>
<tr>
<td>● Quota cannot be changed</td>
<td>● <strong>Perfect for sharing data within research group</strong></td>
</tr>
<tr>
<td>● Snapshot backups (7-day history)</td>
<td></td>
</tr>
<tr>
<td>● <strong>Perfect for storing data such as source code, shell scripts</strong></td>
<td></td>
</tr>
</tbody>
</table>
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
- (Support is reduced)
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/](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](https://docs.nersc.gov/environment/#nersc-modules-environment)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>module</td>
<td>To list the modules in your environment</td>
</tr>
<tr>
<td>list</td>
<td>To list the modules in your environment</td>
</tr>
<tr>
<td>avail</td>
<td>To list available modules</td>
</tr>
<tr>
<td>avail -S</td>
<td>To see all available modules</td>
</tr>
<tr>
<td>load/unload</td>
<td>To load or unload module</td>
</tr>
<tr>
<td>show/display</td>
<td>To see what a module loads</td>
</tr>
<tr>
<td>whatis</td>
<td>Display the module file information</td>
</tr>
<tr>
<td>swap/switch</td>
<td>To swap two modules</td>
</tr>
<tr>
<td>help</td>
<td>General help: $module help</td>
</tr>
<tr>
<td></td>
<td>Information about a module: $ module help PrgEnv-cray</td>
</tr>
</tbody>
</table>

For example: to swap architecture target from Haswell to KNL

% module swap craype-haswell craype-mic-knl
Default Loaded Modules

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) modules/3.2.11.4</td>
<td>12) xpmem/2.2.27-7.0.3.1_3.10__gada73ac.ari</td>
</tr>
<tr>
<td>2) darshan/3.3.1</td>
<td>13) job/2.2.4-7.0.3.1_3.17__g36b56f4.ari</td>
</tr>
<tr>
<td>3) craype-network-aries</td>
<td>14) dvs/2.12_2.2.224-7.0.3.1_3.14__gc77db2af</td>
</tr>
<tr>
<td>4) intel/19.1.2.254</td>
<td>15) alps/6.6.67-7.0.3.1_3.21__gb91cd181.ari</td>
</tr>
<tr>
<td>5) craype/2.7.10</td>
<td>16) rca/2.2.20-7.0.3.1_3.18__g8e3fb5b.ari</td>
</tr>
<tr>
<td>6) cray-libsci/20.09.1</td>
<td>17) atp/3.14.9</td>
</tr>
<tr>
<td>7) udreg/2.3.2-7.0.3.1_3.16__g5f0d670.ari</td>
<td>18) perftools-base/21.12.0</td>
</tr>
<tr>
<td>8) ugni/6.0.14.0-7.0.3.1_6.4__g8101a58.ari</td>
<td>19) PrgEnv-intel/6.0.10</td>
</tr>
<tr>
<td>9) pmi/5.0.17</td>
<td>20) craype-haswell</td>
</tr>
<tr>
<td>10) dmapp/7.1.1-7.0.3.1_3.21__g93a7e9f.ari</td>
<td>21) cray-mpich/7.7.19</td>
</tr>
<tr>
<td>11) gni-headers/5.0.12.0-7.0.3.1_3.9__gd0d73fe.ari</td>
<td>22) craype-hugepages2M</td>
</tr>
</tbody>
</table>
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
  - Compile on login nodes; Executables run on compute nodes
- Recommend to build separate binaries for each architecture (Cori Haswell, Cori KNL, Perlmutter CPU, Perlmutter GPU) to take advantage of optimizations unique to processor type
  - Haswell binaries do run on KNL
  - All other binaries are not compatible among each other
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
Building Simple Test Program (1)

- To build on Cori Haswell:
  - Using default Intel compiler:
    \[ \text{ftn} \ -o \text{mytest} \text{ mytest\_code.F90} \]
  - Using Cray compiler:
    \[ \text{module swap PrgEnv-intel PrgEnv-cray} \]
    \[ \text{ftn} \ -o \text{mytest} \text{ mytest\_code.F90} \]
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
Building Applications on Perlmutter

- User environment and instructions are still evolving
- Building for Perlmutter CPU similar to Cori
- More info on building for Perlmutter GPU
  - [https://docs.nersc.gov/systems/perlmutter/#compilingbuilding-software](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/](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 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)

---

**Other Compute Nodes allocated to the job**
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

```bash
#!/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
```

- 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
```

- 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

4 MPI tasks per node in this example
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/](https://docs.nersc.gov/jobs/affinity/)
Cori Haswell Compute Nodes

- 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

To obtain processor info:

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

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

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)

```bash
#!/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

- 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 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/
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
Bundle Jobs

Multiple Jobs Sequentially:

```bash
#!/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
```

Multiple Jobs Simultaneously:

```bash
#!/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 largest 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-sequentially](https://docs.nersc.gov/jobs/examples/#multiple-parallel-jobs-sequentially)

- [https://docs.nersc.gov/jobs/examples/#multiple-parallel-jobs-simultaneously](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!

```python
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/](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/](https://docs.nersc.gov/jobs/workflow/gnuparallel/)
NERSC Job Script Generator

https://my.nersc.gov/script_generator.php
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/](https://docs.nersc.gov/jobs/monitoring/)
- On the web
  - [https://my.nersc.gov](https://my.nersc.gov)
    - Cori Queues, Queue backlogs, Queue Wait Times (statistics data)
  - [https://www.nersc.gov/users/live-status/](https://www.nersc.gov/users/live-status/) Queue Look
  - [https://iris.nersc.gov](https://iris.nersc.gov) the “Jobs” tab
## Cori Haswell Queue Policy (as of June 2022)

<table>
<thead>
<tr>
<th>QOS</th>
<th>Max nodes</th>
<th>Max time (hrs)</th>
<th>Submit limit</th>
<th>Run limit</th>
<th>Priority</th>
<th>QOS Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>regular</td>
<td>512/1932²</td>
<td>48</td>
<td>5000</td>
<td>-</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>shared³</td>
<td>0.5</td>
<td>48</td>
<td>10000</td>
<td>-</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>interactive</td>
<td>64⁴</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>debug</td>
<td>64</td>
<td>0.5</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>premium</td>
<td>1772</td>
<td>48</td>
<td>5</td>
<td>-</td>
<td>2</td>
<td>2 -&gt; 4⁵</td>
</tr>
<tr>
<td>flex</td>
<td>64</td>
<td>48</td>
<td>5000</td>
<td>-</td>
<td>6</td>
<td>0.5</td>
</tr>
<tr>
<td>overrun</td>
<td>1772</td>
<td>4</td>
<td>5000</td>
<td>-</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>xfer</td>
<td>1 (login)</td>
<td>48</td>
<td>100</td>
<td>15</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>bigmem</td>
<td>1 (login)</td>
<td>72</td>
<td>100</td>
<td>1</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>realtime</td>
<td>custom</td>
<td>custom</td>
<td>custom</td>
<td>custom</td>
<td>1</td>
<td>custom</td>
</tr>
<tr>
<td>compile</td>
<td>1 (login)</td>
<td>24</td>
<td>5000</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

² See note for 512/1932²
³ See note for shared³
⁴ See note for 64⁴
⁵ See note for 2 -> 4⁵
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
● 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
Running Jobs Considerations

- Running jobs on Perlmutter and instructions are still evolving
  - Running on CPU is similar to running on Cori Haswell
  - More examples for Running on GPU: [https://docs.nersc.gov/systems/perlmutter/running-jobs/](https://docs.nersc.gov/systems/perlmutter/running-jobs/)

- Remember to compile separately for each type of compute nodes

- Running jobs from global homes ($HOME) 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 use shifter for large jobs using shared libraries
Data Analytics Software and Services
Cori’s Data Friendly Features

- 12 32-core Haswell
  500 GB Login Nodes
- 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
- Jupyter Notebook
  Node
- Cray DataWarp:
  Burst Buffer for
  I/O acceleration
- Interactive Queue:
  64 Nodes x 4 Hours
- External Network
  Access to/from
  Compute Nodes
<table>
<thead>
<tr>
<th>Capabilities</th>
<th>Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Transfer + Access</td>
<td>globus online, GridFTP, jupyter, python, django, newt</td>
</tr>
<tr>
<td>Workflows</td>
<td>Parsl, GNUparallel, papermill, FireWorks, TaskFarmer</td>
</tr>
<tr>
<td>Data Management</td>
<td>HDF, netCDF, ROOT, MongoDB, MySQL, PostgreSQL</td>
</tr>
<tr>
<td>Data Analytics</td>
<td>python, Spark, julia, TensorFlow, PyTorch</td>
</tr>
<tr>
<td>Data Visualization</td>
<td>ParaView, VisIt</td>
</tr>
</tbody>
</table>

Production Data Software Stack
Data Analytic Software Services

- Globus Online
- Science Gateways
- Databases
- Shifter
- Python
- Jupyter
- Machine Learning / Deep Learning
- Workflows
- And more …
Globus Online: Move Data

- [https://www.globus.org](https://www.globus.org)  [https://docs.nersc.gov/services/globus/](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 Cori, NERSC Perlmutter, NERSC HPSS, etc.
  - Other Center has endpoints
  - Setup [Globus Connect Personal](https://www.globus.org) to ease transfer between local system (such as laptop) and NERSC systems
Globus File Transfer Example
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://docs.nersc.gov/services/spin/)
    - [https://www.nersc.gov/users/training/spin/](https://www.nersc.gov/users/training/spin/) (SPIN workshop required)
  - Details at: [https://docs.nersc.gov/services/science-gateways/](https://docs.nersc.gov/services/science-gateways/)
● 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/
https://docs.nersc.gov/development/shifter/gpus/
Shifter Accelerates Python Applications

![Graph showing performance of different configurations](image)
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

laptop> docker build -t scanon/myapp:1.1 .
laptop> docker push scanon/myapp:1.1
#!/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
# sbatch ./job.sl
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
- Guide to use Python on Perlmutter:
  - https://docs.nersc.gov/development/languages/python/using-python-perlmutter
- 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
  
  ```bash
  module load python
  conda create -n myenv python=3.9
  conda activate myenv
  conda (or pip) install your_custom_packages
  import antigravity
  conda deactivate myenv
  ```

  https://docs.nersc.gov/development/languages/python/#how-to-run-python-jobs-at-nersc
Options to Run Python Code in Parallel

https://docs.nersc.gov/development/languages/python/parallel-python/#how-to-use-parallelism-in-python

- **Multiprocessing, PyOMP**
  - 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/parallel-python/#using-mpi4py-in-a-shifter-container
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

**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

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

```bash
$ 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.
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 "$@
```

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

<table>
<thead>
<tr>
<th>Shared CPU Node</th>
<th>Exclusive GPU Node</th>
<th>Configurable GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Perlmutter</strong></td>
<td><img src="start.png" alt="start" /></td>
<td><img src="start.png" alt="start" /></td>
</tr>
<tr>
<td><strong>Cori</strong></td>
<td><img src="start.png" alt="start" /></td>
<td><img src="start.png" alt="start" /></td>
</tr>
</tbody>
</table>

**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.

Can request additional access for exclusive Cori CPU and shared Cori GPU nodes
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 --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
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
- using user custom kernels
Hands-on Exercises
Compiling and Running Jobs on Cori

- `% ssh <user>@cori.nersc.gov`
- `% cd $SCRATCH`
- `% git clone https://github.com/NERSC/intro-NERSC-resources.git`
- `% cd intro-NERSC-resources`

Follow
- hello-exercise.README
- matrix-example.README
- xthi-exercise.README

References
- Running Jobs: [https://docs.nersc.gov/jobs/](https://docs.nersc.gov/jobs/)
- Interactive Jobs: [https://docs.nersc.gov/jobs/examples/#interactive](https://docs.nersc.gov/jobs/examples/#interactive)
Using Compute Node Reservations

- Existing NERSC users are added to “ntrain4” project
- Cori node reservations available from 2-3:30 pm today
- User reservations with --reservation=xxx -A ntrain4, where
  - xxx is “intro_hsw” or “intro_knl”
Thank You