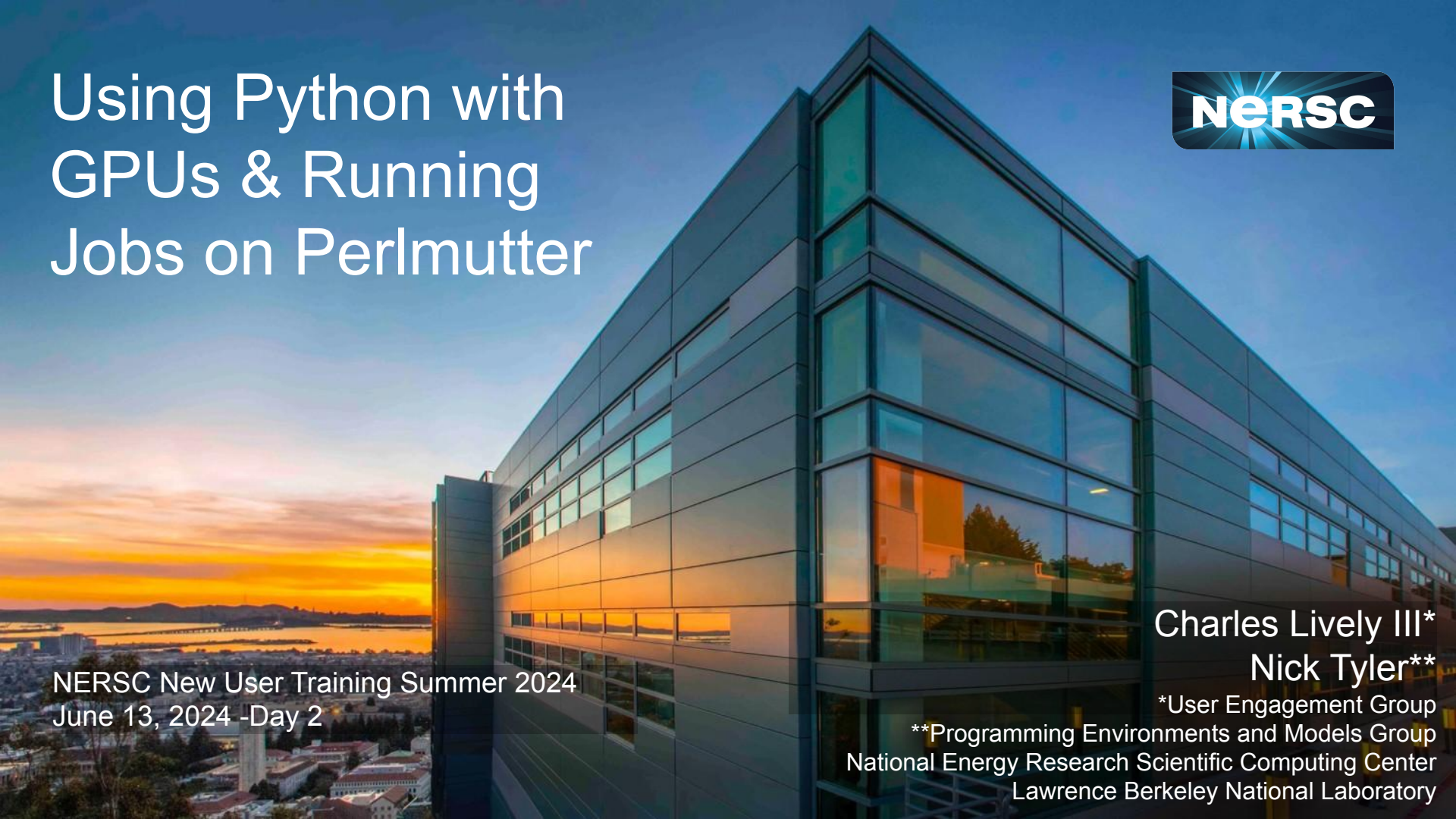
Using Python with GPUs & Running Jobs on Perlmutter



NERSC New User Training Summer 2024
June 13, 2024 -Day 2

Charles Lively III*
Nick Tyler**
*User Engagement Group
**Programming Environments and Models Group
National Energy Research Scientific Computing Center
Lawrence Berkeley National Laboratory

Using Python on GPUs

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, CUDA Python**
  - multi-gpu / multi-node / distributed memory:
    - **mpi4py+X, dask, cuNumeric**
- Many of these GPU libraries have adopted the [CUDA Array Interface] which makes it easier to pass array-like objects stored in GPU memory between the libraries
- There is also effort in the community to standardize around a common [Python array API]


```
numpy:          mean(a, axis=None, dtype=None, out=None, keepdims=<no value>)
dask.array:     mean(a, axis=None, dtype=None, out=None, keepdims=<no value>)
cupy:           mean(a, axis=None, dtype=None, out=None, keepdims=False)
jax.numpy:      mean(a, axis=None, dtype=None, out=None, keepdims=False)
mxnet.np:       mean(a, axis=None, dtype=None, out=None, keepdims=False)
sparse:         s.mean(axis=None, keepdims=False, dtype=None, out=None)
torch:          mean(input, dim, keepdim=False, out=None)
tensorflow:     reduce_mean(input_tensor, axis=None, keepdims=None, name=None,
                            reduction_indices=None, keep_dims=None)
```

cudatoolkit dependency via module

```
> module load conda
```

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

```
> conda create --name cupy-demo python=3.11 numpy scipy
> conda activate cupy-demo
> pip install cupy-cuda11X
> python
>>> import cupy as cp
>>> print(cp.array([1, 2, 3]))
[1 2 3]
```

Check your package documentation to see cudatoolkit compatibility requirements

See documentation at https://docs.nersc.gov/development/languages/python/using-python-perlmutter/

cudatoolkit dependency via conda

```
> module load conda
> module unload cudatoolkit
> conda create --name cupy-demo python=3.11 numpy scipy
> conda activate cupy-demo
> conda install -c conda-forge cupy
> python
>>> import cupy as cp
>>> print(cp.array([1, 2, 3]))
[1 2 3]
```

cupy conda-forge package will pull cudatoolkit dependencies into conda env

cupy conda-forge package will pull cudatoolkit dependencies into conda env

See documentation at https://docs.nersc.gov/development/languages/python/using-python-perlmutter/

Is my code a good fit for a GPU?

CPUs → low latency
GPUs → high throughput

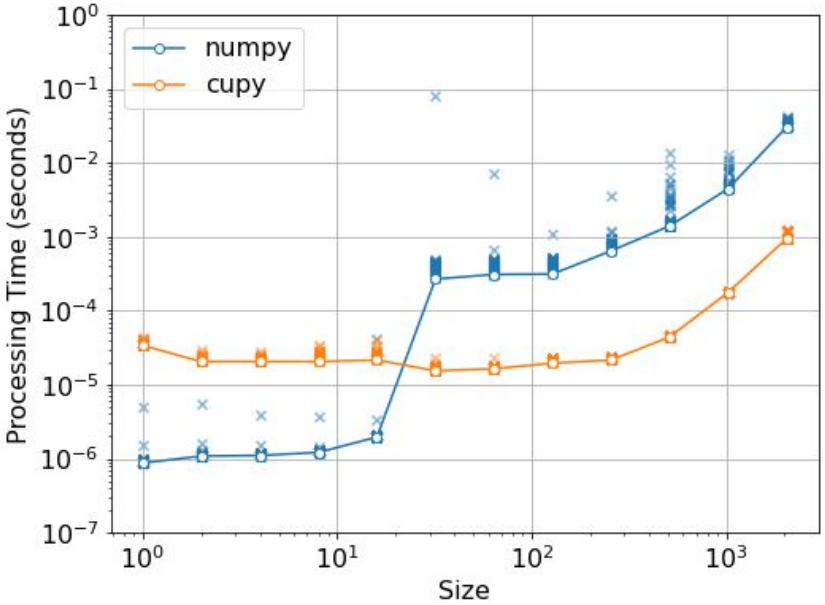
GPUs are likely a good fit if the following are true for your application:

- Performs computation using large arrays, matrices, or images
- Dataset can fit in GPU memory
  - (40GB for Perlmutter's A100 GPUs)
- IO is not a bottleneck

For more help choosing a GPU-accelerated Python framework:

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

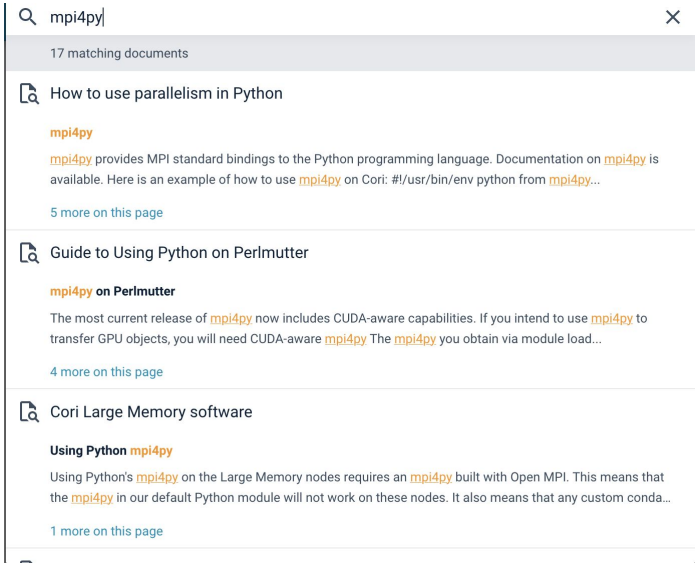
```
a = xp.random.rand(size, size)
b = xp.random.rand(size, size)
def f(a, b):
    return xp.dot(a, b)
```



Best Practices & Where to get Python information

- Utilize Conda

- Check out Python in NERSC docs:
  - Python at NERSC
  - Python on Perlmutter
  - Jupyter at NERSC
  - Try the search bar at docs.nersc.gov, it's pretty good!
- Can't find the answer? Submit a ticket at help.nersc.gov



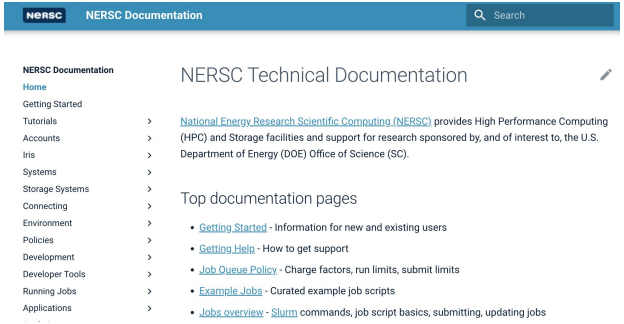
Summary

- Welcome to NERSC!
- We are here to help you use Python productively on Perlmutter
- If you have questions, please check our docs.nersc.gov or file a ticket at help.nersc.gov

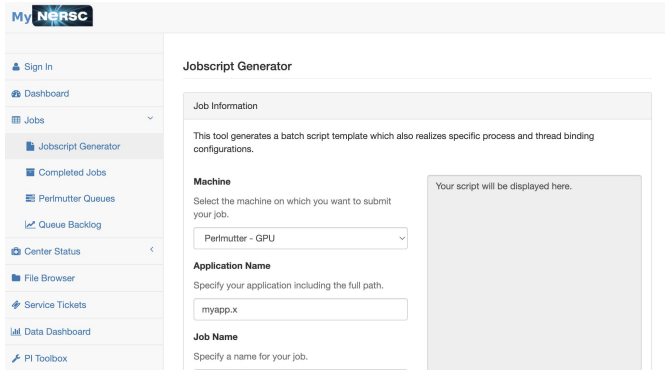
Pipeline

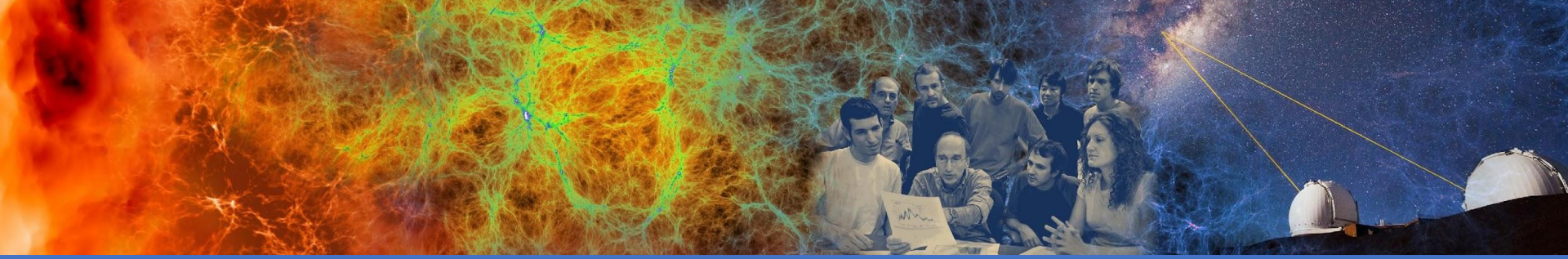
- If you're brand new to HPC, Welcome!
  - What is a job?
  - How to run your code as a job?
- If you're just new to NERSC, Also Welcome!
  - Get to more advanced topics later
    - Running a job in container
    - Workflows
- Docs and Script Generator
-

https://docs.nersc.gov



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





Basic Job Submission

What is a Job? How do I get one?

- When you connect to Perlmutter you are on a login node
  - This includes Jupyter sessions
- Login nodes are **NOT** meant for large computing tasks!
  - They are shared by all users
  - Be kind to your fellow user
  - We only have 40 login nodes
- So where does my computation go?
  - On a compute node!
  - Perlmutter has 4864 compute nodes
    - 1792 GPU nodes, 3072 CPU nodes

What is a Job? How do I get one?

- There are two ways to access a compute node
  - Interactive job
    - Directly connect to the compute node
    - Through a command line interface
    - Have a jupyter notebook on a compute node
  - Batch job
    - Place the work you want to do in a script
    - Submit the script to a queue
    - Wait for the work to be done

How are jobs managed?

- Perlmutter uses Slurm workload manager
  - Slurm is an open source tool that performs job scheduling
- Slurm takes care of three key responsibilities
  - Allocating computer resources to jobs
  - Executes and monitors all jobs
  - Managing priorities of the jobs
- Even if you're familiar with Slurm it is configured differently per site

How do I get a job from Slurm?

- Interactive
  - `salloc` - Slurm allocation
    - Gets an allocation on a node or set of nodes
  - At NERSC this defaults to running your login shell on a node in the allocation

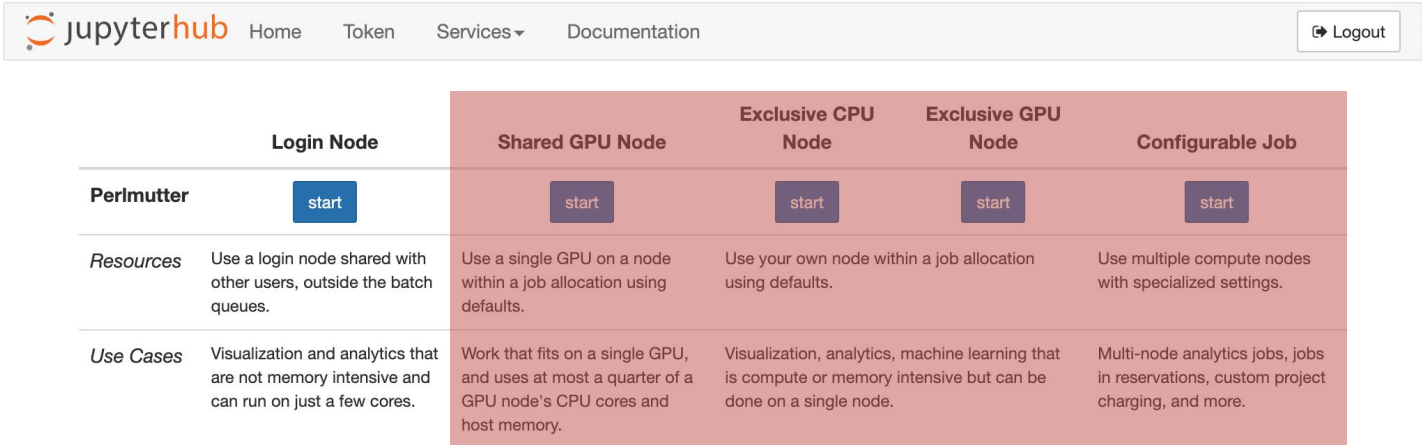
```
clively@nid001465:/global/u1/c/clively> salloc -A m4388 -N 1 -t 10:00 -C gpu
salloc: Pending job allocation 25915811
salloc: job 25915811 queued and waiting for resources
salloc: job 25915811 has been allocated resources
salloc: Granted job allocation 25915811
salloc: Waiting for resource configuration
salloc: Nodes nid004053 are ready for job
clively@nid004053:/global/u1/c/clively>
```

What did I ask Slurm to do?

- `salloc -A m0000 -N 1 -t 10:00 -C gpu`
- `salloc`
  - Give me some compute nodes to use
- `-A m0000 | --account=m0000`
  - Charge to this NERSC account (usually starts with `m`)
- `-N 1 | --nodes=1`
  - Get 1 compute node to work on
- `-t 10:00 | --time=10:00`
  - Give me that node for 10 minutes
- `-C gpu | --constraint=gpu`
  - The type of node you want, either `cpu` or `gpu`

How do I get a job from Slurm?

- Interactive allocations in Jupyter
  - These options can get you on a compute node
  - Come tomorrow to learn more about Jupyter!



When do I use an interactive job?

- Use interactive jobs to test and debug code
  - Also good option for profiling code
- Jobs in the interactive queue have limits
  - `-q interactive | --qos=interactive`
    - 1-4 nodes && 4 hours max walltime
  - `-q shared_interactive | --qos=shared_interactive`
    - ½ node max && 4 hours max walltime
      - 2 GPUs, 32 cores, 64 threads, ~120GB ram
      - 64 cores, 128 threads, ~250GB ram

I need more time and nodes!

- Use a batch job
  - Submits the work you want to do into a queue
  - Lets Slurm schedule your work
    - Allows Slurm to give your job more time
    - Allows Slurm to schedule more compute nodes

```
clively@nid004053:/global/u1/c/clively> sbatch job_script.sh
Submitted batch job 25916143
clively@nid004053:/global/u1/c/clively>
```

How do I submit a batch job?

- `sbatch` - Slurm Batch
  - Submit a batch script to Slurm
  - `sbatch job_script.sh`
  - Slurm gives you back a job id

```
clively@nid004053:/global/u1/c/clively> sbatch job_script.sh
Submitted batch job 25916143
clively@nid004053:/global/u1/c/clively>
```

What does script.sh look like?

```
#!/bin/bash
#SBATCH -A m0000
#SBATCH -q regular
#SBATCH -N 4
#SBATCH -t 8:00:00
#SBATCH -C cpu
#SBATCH -J science
#SBATCH -o %x_%j.out
#SBATCH -e %x_%j.err


srun -n $SLURM_NNODES hostname
```

- Similar options to `salloc`
- Add the special `#SBATCH` comment
- Slurm reads options from script
- Ask for 4 nodes for 8 hours
  - `-J science | --job-name=science`
  - Organize slurm outputs
    - `%x` - job name
    - `%j` - job id

What does script.sh look like?

```
#!/bin/bash
#SBATCH -A m0000
#SBATCH -q regular
#SBATCH -N 4
#SBATCH -t 8:00:00
#SBATCH -C cpu
#SBATCH -J science
#SBATCH -o %x_%j.out
#SBATCH -e %x_%j.err


srun -n $SLURM_NNODES hostname
```

- Slurm adds environment variables to your job
  - Use the `$SLURM_NNODES` to get number of nodes requested
- Slurm run - `srun`
  - Run parallel jobs
    - Use this instead of `mpirun`
- This will run one `hostname` per node

Helpful Slurm environment variables

```
SLURM_JOB_NUM_NODES # -N/--nodes=
SLURM_NTASKS_PER_NODE # --ntasks-per-node=
SLURM_CPUS_ON_NODE # Set by Slurm
SLURM_GPUS_ON_NODE # Set by Slurm

Total CPUs: $((SLURM_JOB_NUM_NODES * SLURM_CPUS_ON_NODE))

Total Tasks: $((SLURM_JOB_NUM_NODES * SLURM_NTASKS_PER_NODE))

CPUs per Task: $(((SLURM_JOB_NUM_NODES * SLURM_CPUS_ON_NODE) / SLURM_NTASKS))

Total GPUs: $((SLURM_JOB_NUM_NODES * SLURM_GPUS_ON_NODE))

GPUs per Task: $(((SLURM_JOB_NUM_NODES * SLURM_GPUS_ON_NODE) / SLURM_NTASKS))
```

What does the -q option do?

- Different queues with different limits
- `-q qebug | --qos=debug`
  - 1-8 nodes && 30 minute max walltime
  - Test your script
  - Scaling before running larger jobs
- `regular` and `shared`
  - Where science gets done!
  - 24 hour max walltime, 5000 max job submissions
  - `-q regular | --qos=regular`
  - `-q shared | --qos=shared`
    - ½ node max per job

How do I debug my script?

- Override options in the script with CLI options
- Helpful for debugging or scaling tests
  - Use the debug queue
    - `sbatch -q debug -t 10 script.sh`
  - Scale testing
    - `sbatch -N 2 script.sh`
    - `sbatch -N 20 script.sh`

How do I see if my jobs working?

- `squeue` - Slurm queue
  - view information about jobs in the Slurm queue
  - Returns information from all jobs
    - Can be a lot on a big system like Perlmutter
- `sqs`
  - NERSC shortcut with some helpful output options
- Shows job state `R` - Running, `PD` - Pending
- `TIME` - How long the job has been running

```
clively@nid001465:/global/u1/c/clively> sqs
JOBID             ST USER      NAME              NODES TIME_LIMIT       TIME  SUBMIT_TIME          QOS             START_TIME          FEATURES       NODELIST(REASON
24125002          PD clively   /global/cfs/      1          10:00       0:00  2024-05-22T03:31:22  cron            2024-05-29T03:31:00  cron           (BeginTime)
24125000          PD clively   /global/cfs/      1          10:00       0:00  2024-05-22T02:12:08  cron            2024-05-23T02:11:00  cron           (BeginTime)
24125001          PD clively   /global/cfs/      1          10:00       0:00  2024-05-22T00:21:49  cron            2024-05-23T00:21:00  cron           (BeginTime)
25915811          CG clively   interactive       1          10:00      10:17  2024-05-22T07:29:14  gpu_debug       2024-05-22T07:30:26  gpu&a100       nid004053
25915593          R  clively   interactive       1        1:00:00      15:47  2024-05-22T07:25:48  gpu_interactive 2024-05-22T07:26:16  gpu&a100       nid001465
```

How do I end a job?

- `scancel` - Slurm cancel
  - Send stop signal to jobs or job steps managed by Slurm
    - Stop job running too long or with the wrong parameters
    - Conserve your NERSC hours if you made a mistake!

```
clively@login22:/global/u1/c/clively> sbatch -q debug -t 20 job_script.sh
Submitted batch job 26735450
clively@login22:/global/u1/c/clively> sqs
JOBID             ST USER      NAME          NODES TIME_LIMIT       TIME  SUBMIT_TIME          QOS             START_TIME
  FEATURES        NODELIST(REASON
24125002          PD clively   /global/cfs/  1          10:00       0:00  2024-06-12T03:31:37  cron            2024-06-19T03:31:00
  cron            (BeginTime)
24125000          PD clively   /global/cfs/  1          10:00       0:00  2024-06-12T02:11:56  cron            2024-06-13T02:11:00
  cron            (BeginTime)
24125001          PD clively   /global/cfs/  1          10:00       0:00  2024-06-12T00:21:11  cron            2024-06-13T00:21:00
  cron            (BeginTime)
26735450          PD clively   job_script.s  1          20:00       0:00  2024-06-12T20:48:40  gpu_debug       N/A
  gpu&a100        (Resources)
clively@login22:/global/u1/c/clively> scancel 26735450
clively@login22:/global/u1/c/clively> sqs
```

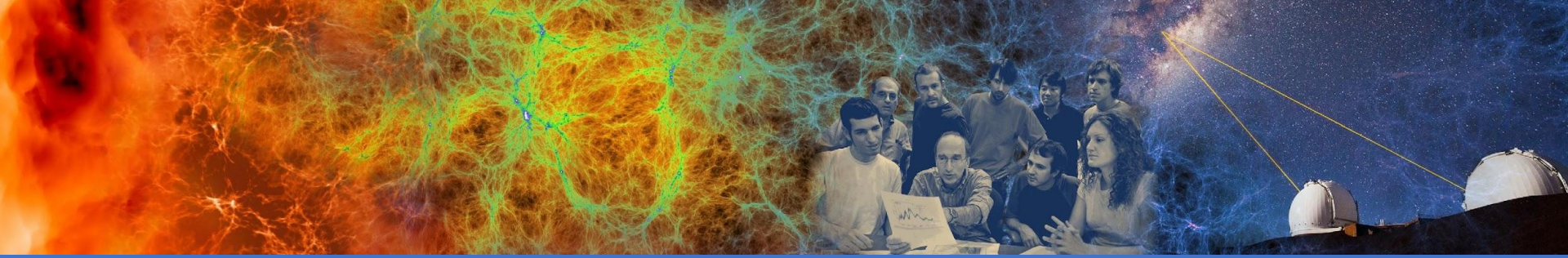
How to look at completed jobs?

- `sacct` - Slurm accounting
  - Accounting data for all jobs and job steps in the Slurm job accounting log or Slurm database
  - By default shows jobs completed in the last day

```
clively@login22:/global/u1/c/clively> sacct
JobID           JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
24125002     /global/c+       cron      nstaff          2  REQUEUED    126:0
24125002.ba+      batch                 nstaff          2    FAILED    126:0
24125002.ex+     extern                 nstaff          2 COMPLETED      0:0
24125001     /global/c+       cron      nstaff          2  REQUEUED      0:0
24125001.ba+      batch                 nstaff          2 COMPLETED      0:0
24125001.ex+     extern                 nstaff          2 COMPLETED      0:0
24125000     /global/c+       cron      nstaff          2  REQUEUED      0:0
24125000.ba+      batch                 nstaff          2 COMPLETED      0:0
24125000.ex+     extern                 nstaff          2 COMPLETED      0:0
26705767             fsp regular_m+      nstaff        256    FAILED      1:0
26705767.ba+      batch                 nstaff        256    FAILED      1:0
26705767.ex+     extern                 nstaff        256 COMPLETED      0:0
26734859     job_scrip+   gpu_ss11     m4388_g        128 COMPLETED      0:0
26734859.ba+      batch                 m4388_g        128 COMPLETED      0:0
26734859.ex+     extern                 m4388_g        128 COMPLETED      0:0
26734859.0   hello_mpi+                m4388_g          2 COMPLETED      0:0
26734988     job_scrip+   gpu_ss11     m4388_g          0 CANCELLED+     0:0
```

How to look at completed jobs?

- `sacct -j jobid`
  - Shows information about one jobid
- `sacct --name science --constraint gpu`
  - Search through jobs by other attributes
- `sacct --state running`
  - Select jobs based on their current state



Jobs in containers

Running jobs in containers

- Containers are a great
  - Make your software portable between systems
  - Decrease start time of large jobs
    - python
- NERSC Supports two container technologies
  - Shifter
  - podman-hpc - ***New***
    - Can build images on login nodes!
- We don’t support Singularity/Apptainer on Perlmutter

What is a container?

- A way to pack up all your software
- Docker is just one technology
- On your personal computer
  - Build
    - `docker build …`
  - Ship
    - `docker push …`
  - Run
    - `docker run …`

```
#Dockerfile
FROM ubuntu:latest

RUN apt-get update &&
    apt-get install -y \
    cmake python3-pip

RUN pip install pandas

COPY code /mycode
WORKDIR /mycode
RUN cmake --build .
```

Where do I ship it?

- NERSC has a registry
  - `registry.nersc.gov`
  - Build
    - `docker build -t registry.nersc.gov/m0000/test:v1.0 .`
  - Ship
    - `docker login registry.nersc.gov`
    - `docker push registry.nersc.gov/m0000/test:v1.0`
  - Run with Shifter or Podman-HPC

How do I run a Shifter container?

- Pull your image before you start your job
  - `shifterimg pull registry/image:tag`

```
#!/bin/bash
#SBATCH -A m0000
#SBATCH -q regular
#SBATCH -N 4
#SBATCH -t 8:00:00
#SBATCH -C cpu
#SBATCH -J science
#SBATCH -o %x_%j.out
#SBATCH -e %x_%j.err
#SBATCH --image=registry/image:tag

srun -n $SLURM_NNODES shifter hostname
```

How do I run a Shifter container?

- Extra options for `shifter`
  - `--volume=/pscratch/sd/u/user:/scratch`
  - `--env=MYENV=1234`
  - `--clearenv`
  - `--workdir=/work`
  - `--module=...`
    - `none`
    - `mpich`
    - `cvmfs`
    - `gpu`
    - `cuda-mpich`
    - `nccl-2.15`
    - `network`

How do I run a podman-hpc container?

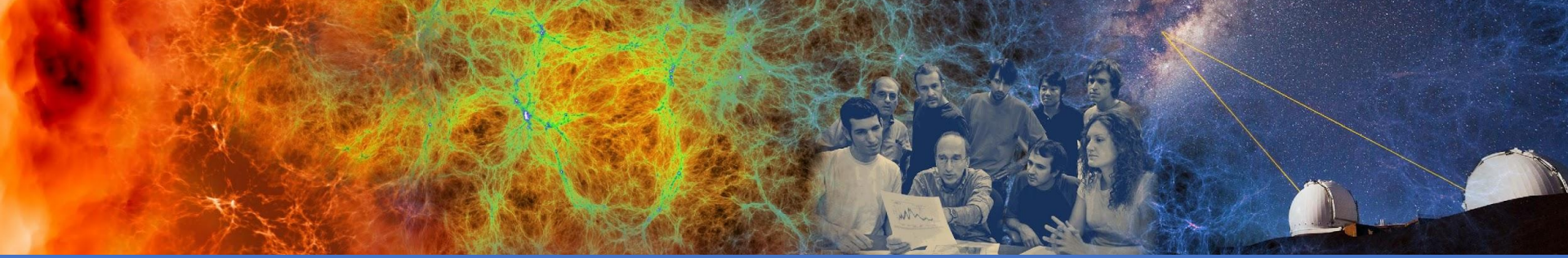
- Pull your image before you start your job
  - `podman-hpc pull registry/image:tag`

```
#!/bin/bash
#SBATCH -A m0000
#SBATCH -q regular
#SBATCH -N 4
#SBATCH -t 8:00:00
#SBATCH -C cpu
#SBATCH -J science
#SBATCH -o %x_%j.out
#SBATCH -e %x_%j.err

srun -n $SLURM_NNODES \
     podman-hpc run registry/image:tag hostname
```

podman

How do I run a podman-hpc container?

- Pull, Or build images on login nodes, then migrate to scratch
  - `podman-hpc build -t image_name:tag .`
  - `podman-hpc migrate image_name:tag`
- Docker/Podman options work
  - `--volume=/pscratch/sd/u/user:/scratch`
  - `--net host`
- Extra options similar to shifter modules
  - `--mpi`
  - `--gpu`
  - `--cuda-mpi`



Multiple jobs and Workflows

I have multiple things I need to do

- Bundling jobs with slurm
  - Run multiple executables sequentially or simultaneously
- Use a Slurm job array
  - Same job task with different inputs
- Workflow tools
  - GNU Parallel
    - Many small tasks, fit onto one node
  - More complex tasks
    - Parsl, Fireworks, etc.

Bundling work into one job

```
#!/bin/bash
#SBATCH -A m0000
#SBATCH -q regular
#SBATCH -N 4
#SBATCH -t 8:00:00
#SBATCH -C cpu
#SBATCH -J science
#SBATCH -o %x_%j.out
#SBATCH -e %x_%j.err

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
```

- Bundling jobs with slurm
  - Programs run **sequentially**
  - Only have to wait for scheduler once
    - Reuse the same allocated nodes for different steps in your workflow

Bundling work into one job

```
#!/bin/bash
#SBATCH -A m0000
#SBATCH -q regular
#SBATCH -N 4
#SBATCH -t 8:00:00
#SBATCH -C cpu
#SBATCH -J science
#SBATCH -o %x_%j.out
#SBATCH -e %x_%j.err

srun -N 1 -n 256 ./a.out input0 &
srun -N 1 -n 256 ./a.out input1 &
srun -N 1 -n 256 ./a.out input3 &
srun -N 1 -n 256 ./a.out input4 &
wait
```

- Bundling jobs with slurm
  - Programs run **simultaneously**
  - Only have to wait for scheduler once
    - This example runs same program with different inputs per `srun`

Using Job Arrays

```
#!/bin/bash
#SBATCH -A m0000
#SBATCH -q regular
#SBATCH -N 1
#SBATCH -t 8:00:00
#SBATCH -C cpu
#SBATCH -J science
#SBATCH -o %x_%j.out
#SBATCH -e %x_%j.err
#SBACTH --array=1-4

echo $SLURM_ARRAY_JOB_ID

srun -n 256 ./a.out $SLURM_ARRAY_JOB_ID
```

- Slurm manages each job independently
  - If one task fails it won't affect others
- Good option for getting
  - Large statistics on same inputs
  - Parameter sweep over input files

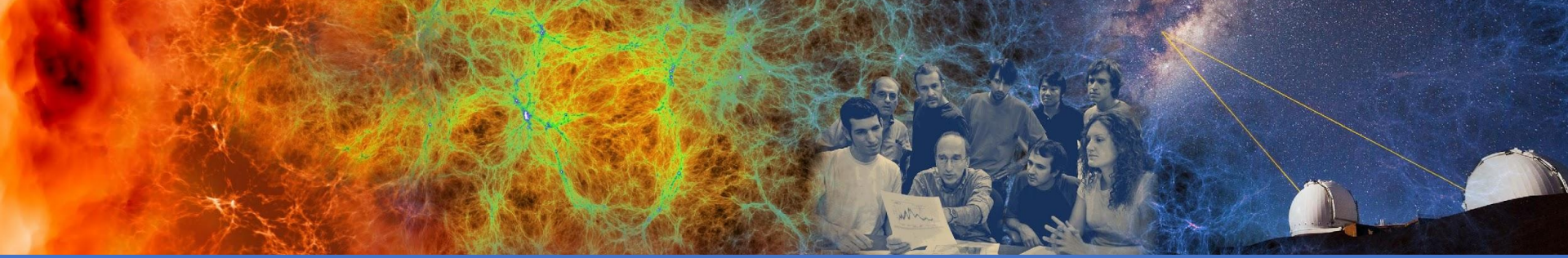
Using GNU Parallel

```
#!/bin/bash
#SBATCH -A m0000
#SBATCH -q regular
#SBATCH -N 1
#SBATCH -t 8:00:00
#SBATCH -C cpu
#SBATCH -J science
#SBATCH -o %x_%j.out
#SBATCH -e %x_%j.err

module load parallel

parallel -j256 ./a.out {} ::: inputs*
```

- You manage tasks inside of allocation
  - Great for many small tasks
    - Faster start times than `sruns`
  - Reuse allocation for all your tasks
- As tasks finish the next one starts
  - Use allocation efficiently

More complex workflows with dependencies

- Use a workflow management system
  - Parsl/FuncX/Globus Compute
  - Fireworks
  - Many more…
  - Write code to define workflow
  - Often written in python
  - Handle dependencies between different types of tasks
  - github.com/CrossFacilityWorkflows/DOE-HPC-workflow-training
    - Resources from previous training with ALCF and OLCF
- Reach out at `help.nersc.gov` with more questions



Best Practices

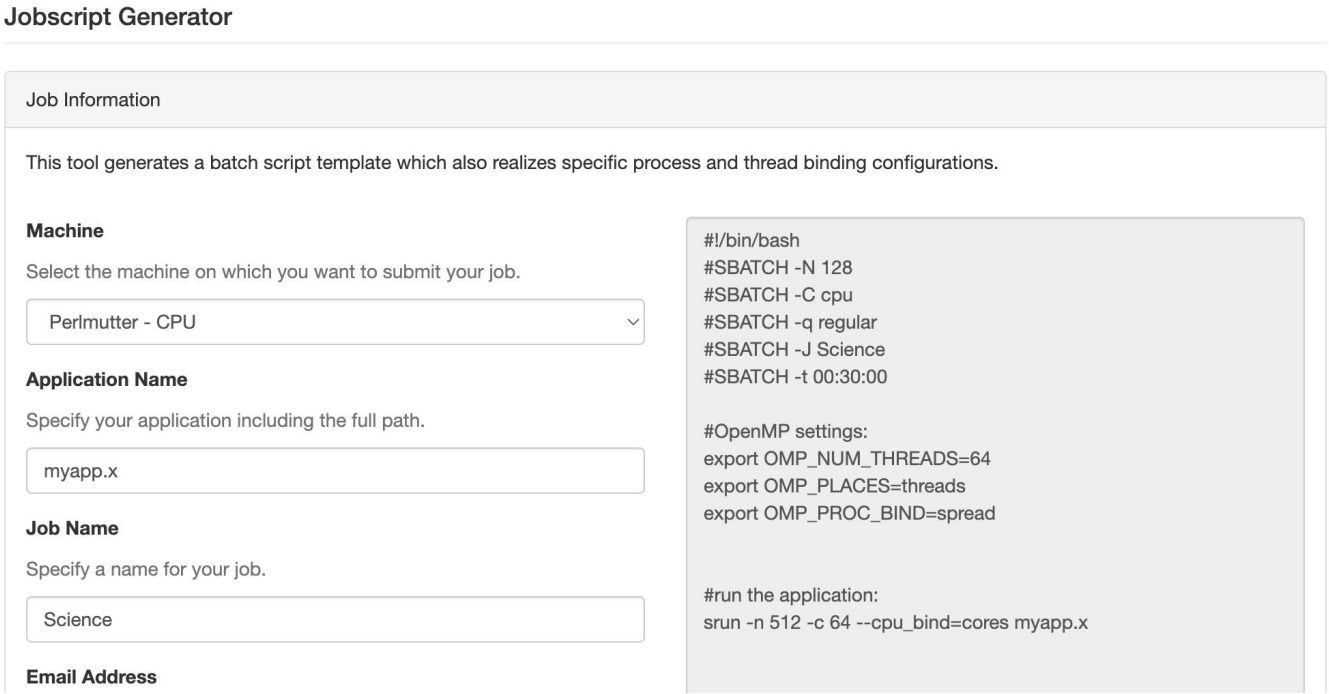
Jobs Scheduling

- Each job has a priority value
  - Grouped by user, QOS, and account
  - Only two jobs per these groupings gain priority at a time
    - More jobs can run, only two will age
- Main scheduler uses priority list
  - Schedules a few days in the future
- Backfill scheduler puts shorter jobs in “holes”
  - Prioritize utilization

Jobs Scheduling Tips

- One job with a large allocation
  - Per node priority ageing is the highest
  - Can get scheduled first
- Shorter time length jobs
  - Easier to schedule as backfill
  - Use a workflow manager
- Choose the right time from Slurm
  - Balance between enough runtime
  - Waiting in the queue for a long job

Job script generator: More advanced threading options



Options for OpenMP Code

```
#!/bin/bash
#SBATCH -A m0000
#SBATCH -q regular
#SBATCH -N 1
#SBATCH -t 8:00:00
#SBATCH -C cpu
#SBATCH -J science
#SBATCH -o %x_%j.out
#SBATCH -e %x_%j.err

export OMP_NUM_THREADS=8
export OMP_PLACES=cores
export OMP_PROC_BIND=spread

srun -n 256 ./a.out $SLURM_ARRAY_JOB_ID
```

- OpenMP
  - config through env variables
- Some libraries use OpenMP by default
  - BLAS/LAPACK
  - numpy in python
    - Small numpy arrays can be faster with less threads

Options for MPI codes

```
#!/bin/bash
#SBATCH -A m0000
#SBATCH -q regular
#SBATCH -N 2
#SBATCH -t 8:00:00
#SBATCH -C cpu
#SBATCH -J science
#SBATCH -o %x_%j.out
#SBATCH -e %x_%j.err


srun -n 32 -c 16 --cpu_bind=cores ./a.out
```

- Settings to Address NUMA Performance
  - Use `--cpu_bind=cores` when
    - `#MPI tasks ≤ #cores`
  - Use `--cpu_bind=threads` when
    - `#MPI tasks > #cores`

Options for Hybrid OpenMP/MPI codes

```
#!/bin/bash
#SBATCH -A m0000
#SBATCH -q regular
#SBATCH -N 2
#SBATCH -t 8:00:00
#SBATCH -C cpu
#SBATCH -J science
#SBATCH -o %x_%j.out
#SBATCH -e %x_%j.err

export OMP_NUM_THREADS=8
export OMP_PLACES=cores
export OMP_PROC_BIND=spread

srun -n 32 -c 16 --cpu_bind=cores ./a.out
```

- Hybrid MPI/OpenMP code
  - Number of cores per task `-c`
  - `-c` ≥ `OMP_NUM_THREADS`
  - Give enough cpus to be able to use OpenMP threads efficiently

Options for gpu codes

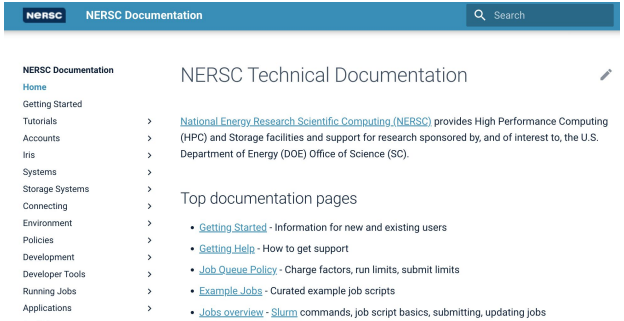
```
#!/bin/bash
#SBATCH -A m0000
#SBATCH -q regular
#SBATCH -N 2
#SBATCH -t 8:00:00
#SBATCH -C gpu
#SBATCH -J science
#SBATCH -o %x_%j.out
#SBATCH -e %x_%j.err

export OMP_NUM_THREADS=8
export OMP_PLACES=cores
export OMP_PROC_BIND=spread

srun -n 8 -c 8 --gpus-per-task=1 \
    --cpu_bind=cores ./a.out
```

- GPU codes
  - Can specify the number of gpus per task
    - `--gpus-per-task=n`
- More advanced
  - Specific gpu mapping
    - `--gpu-bind`

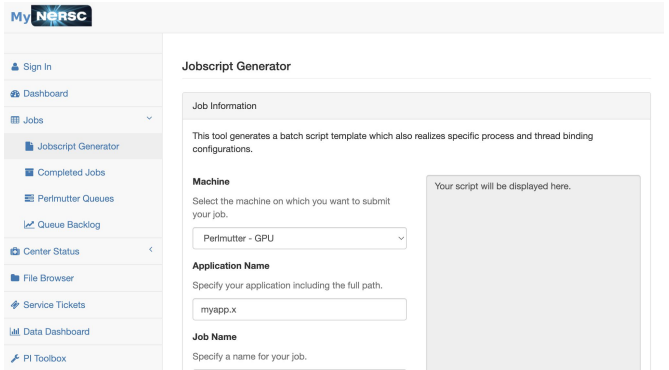
What did we cover?

- What is a job?
- How to run your code as a job?
- Running a job in container
- Workflows
- Docs and Script Generator

https://docs.nersc.gov



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




Thank You for listening and Welcome to NERSC!
NERSC