ARGONNE

ATPESC2024

EXTREME-SCALE COMPUTING

Quick Start on ATPESC Computing Resources

JaeHyuk Kwack
Argonne National Laboratory

Outline

- The DOE Leadership Computing Facility
- ALCF Polaris System
- OLCF Odo System
- NERSC Perlmutter System
- Hands-on

The DOE Leadership Computing Facility

- Collaborative, multi-lab, DOE/SC initiative ranked top national priority in *Facilities for the Future of Science: A Twenty-Year Outlook.*
- Mission: Provide the computational and data science resources required to solve the most important scientific & engineering problems in the world.
- Highly competitive user allocation program (INCITE, ALCC).
- Projects receive 100x more hours than at other generally available centers.
- LCF centers partner with users to enable science & engineering breakthroughs (Liaisons, Catalysts).

Polaris
Argonne National Laboratory
U.S. Department of Energy
intel
Hewlett Packard Enterprise
Aurora
Argonne National Laboratory
Oak Ridge National Laboratory
Summit
IBM
Oak Ridge National Laboratory
U.S. Department of Energy
CRAY
AMD
FRONTIER

Leadership Computing Facility System

	Argonne LCF		Oak Ridge LCF	
System	HPE	HPE	IBM	HPE
Name	Polaris	Aurora (in 2024)	Summit	Frontier
Compute nodes	560	10,624	4,608	9,408
Node architecture	1 x AMD Milan CPU + 4x NVIDIA A100 GPU	2 x Intel Xeon SPR + 6 x Intel PVC GPU	2 x IBM POWER9 CPU + 6 x NVIDIA V100 GPUs	1 x AMD EYPC CPU + 4 x AMD MI250x GPU
Processing Units	560 CPUs + 2,240 GPUs	21,248 CPUs + 63,744 GPUs	9,216 CPUs + 27,648 GPUs	9,408 CPUs + 37,362 GPUs
Memory per node, (gigabytes)	512 GB DDR4 + 160 GB HBM2 + 1600 GB SSD	128 GB HBM2e on CPU + 1024 GB DDR5 on CPU + 768 GB HBM2e on GPU	512 GB DDR4 + 96 GB HBM2 + 1600 GB NVM	512 GB DDR4 + 512 GB HBM2e
Peak performance, (petaflops)	44	> 2 Exaflops DP	200	1.6 Exaflop DP

AVAILABLE COMPUTING RESOUCES FOR ATPESC

ALCF Systems

AMD CPUs + NVIDIA A100 GPUs (Polaris)

OLCF

AMD CPUs + AMD MI-250x GPUs (Odo)

NERSC

AMD CPUs + NVIDIA A100 GPUs (Perlmutter)

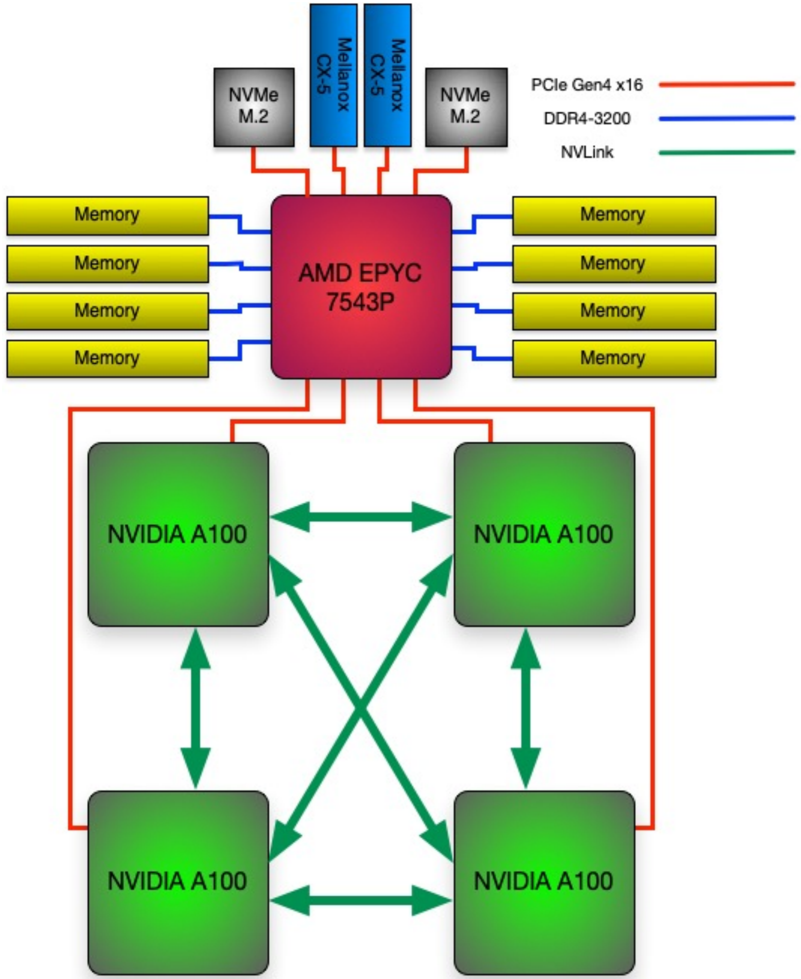
Cloud resources

Intel Developer Cloud (with Intel Data Center GPU Max 1100)
AMD Accelerator Cloud (AAC)

ALCF Polaris System

Polaris Single Node Configuration

# of AMD EPYC 7543P CPUs	1
# of NVIDIA A100 GPUs	4
Total HBM2 Memory	160 GB
HBM2 Memory BW per GPU	1.6 TB/s
Total DDR4 Memory	512 GB
DDR4 Memory BW	204.8 GB/s
# OF NVMe SSDs	2
Total NVMe SSD Capacity	3.2 TB
# of Mellanox NICs	2
Total Injection BW (w/ Cassini)	25 (50) GB/s
PCIe Gen4 BW	64 GB/s
NVLink BW	600 GB/s
Total GPU DP Tensor Core Flops	78 TF



Polaris System Configuration

# of River Compute racks	40
# of Apollo Gen10+ Chassis	280
# of Nodes	560
# of AMD EPYC 7543P CPUs	560
# of NVIDIA A100 GPUs	2240
Total GPU HBM2 Memory	87.5TB
Total CPU DDR4 Memory	280 TB
Total NVMe SSD Capacity	1.75 PB
Interconnect	HPE Slingshot
# of Cassini NICs	1120
# of Rosetta Switches	80
Total Injection BW (w/ Cassini)	28 TB/s 13 TB/s
Total GPU DP Tensor Core Flops	44 PF
Total Power	1.8 MW

Apollo 6500 Gen10+

Polaris Filesystems

- Lustre
  - Home directories (/home)
    - Default quota 50GiB
    - Your home directory is backed up
  - Project directory locations (/eagle) in /eagle/projects
    - Polaris: **/eagle/ATPESC2024**
      - **CREATE A SUBDIRECTORY /eagle/ATPESC2024/usr/your_username**
    - Access controlled by unix group of your project
    - Default quota 1TiB
    - Project directories are NOT backed up
  - With large I/O on Lustre, be sure to consider **stripe width**

Polaris Modules

- A tool for managing a user's environment
  - Sets your PATH to access desired front-end tools
  - *Your compiler version can be changed here*
- *module commands*
  - *help*
  - *list ← what is currently loaded*
  - *avail*
  - *load*
  - *unload*
  - *switch|swap*
  - *use ← add a directory to MODULEPATH*
  - *display|show*

Polaris Compiling

- Cray Programming Environment (PE)
  - HPE provides compiler wrappers by default which includes various libraries (including MPI libraries)
    - Integrates with modules environment
    - HPE provided modules will add headers/libraries/compiler+linker options to compiler
    - -craype-verbose to show actual compile/link command
  - PrgEnv-nvidia (default)
    - cc -> nvc
    - CC -> nvc++
    - ftn -> nvfortran
    - Support CUDA and OpenMP target offload
    - nvcc still available but not used by wrappers
  - PrgEnv-gnu
    - cc -> gcc
    - CC -> g++
    - ftn -> gfortran
- Libraries found in
  - /opt/nvidia
  - /opt/cray

Polaris Running MPI Applications

- Jobs run directly on the compute nodes. The `mpiexec` command runs applications using the Parallel Application Launch Service (PALS)

- mpiexec
  - Execute MPI applications on compute nodes using mpiexec

`-n`	Total number of MPI ranks
`-ppn`	Total number of MPI ranks per node
`--cpu-bind`	CPU binding for application
`--depth`	Number of CPUs per rank
`--env`	Set environment variables (e.g., OMP_NUM_THREADS=nthreads)
`--hostfile`	Indicate file with hostname

- Full list of options available from the man page
- https://docs.alcf.anl.gov/polaris/running-jobs/

Polaris Jobs

- Two parts for running jobs
  - Interacting with scheduler
  - Launching job using `mpiexec`
- Shell script
  - describes parameters for scheduler
  - Commands to run included mpiexec to launch
  - Runs on ‘head’ node of your job
    - Permissible to run computation in your shell script
  - Need to load any of your non-default modules which provide library paths
- `qsub -q prod ./run.sh`
  - Will return the jobid
  - Output and error logs are in submission directory

```
#!/bin/bash
#PBS -A $PROJECT
#PBS -l walltime=01:00:00
#PBS -l select=4
#PBS -l system=polaris
#PBS -l filesystems=home:eagle:grand

rpn=4 # assume 1 process per GPU
procs=$((PBS_NODES*rpn))

# job to “run” from your submission directory
cd $PBS_O_WORKDIR

module load <something>

set +x # report all commands to stderr
env
mpiexec -n $procs -ppn $rpn --cpu-bind core -
genvall ./bin <opts>
```

Interactive job

- Useful for short tests or debugging
- Submit the job with `–I` (letter `I` for Interactive)
  - Debug queue:
    - `qsub -I -l select=1 -l walltime=1:00:00 -q debug -A ATPESC2024 -l filesystems=home:eagle:grand`
  - Using reservation:
    - `qsub -I -l select=1 -l walltime=1:00:00 -q R203xxxx -A ATPESC2024 -l filesystems=home:eagle:grand`
    - Check the reservation queues with `pbs_rstat`
- Wait for job's shell prompt
  - Exit this shell to end your job
- From job's shell prompt, run just like in a script job,
  - `mpiexec -n 8 -ppn 4 ./a.out`
- After job expires, `mpiexec` will fail. *Check* **`qstat $PBS_JOBID`**

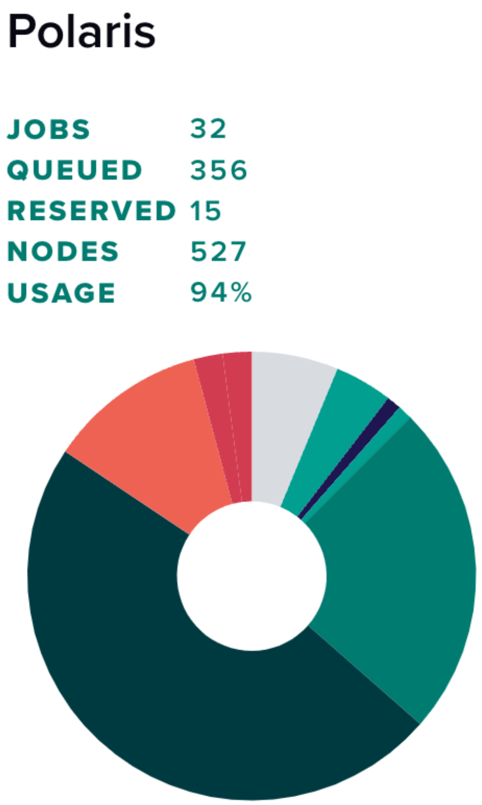
Polaris Scheduler – PBS Professional

- Primary commands
  - qsub
    - Request resources and start your script on the head node
    - `-A` Allocation
    - `-l` Options
    - `-I` Interactive mode
    - `-q` Which queue to submit otherwise default queue
  - qstat
    - Check on the status of requests
    - `-Q` List queues
    - `-f <jobid>` Detailed information about a job
    - `-x <jobid>` Information about a completed job
  - qalter
    - Update your requests
  - qdel
    - Cancel/delete jobs
  - pbs_rstat
    - Check reservations

Polaris Queues

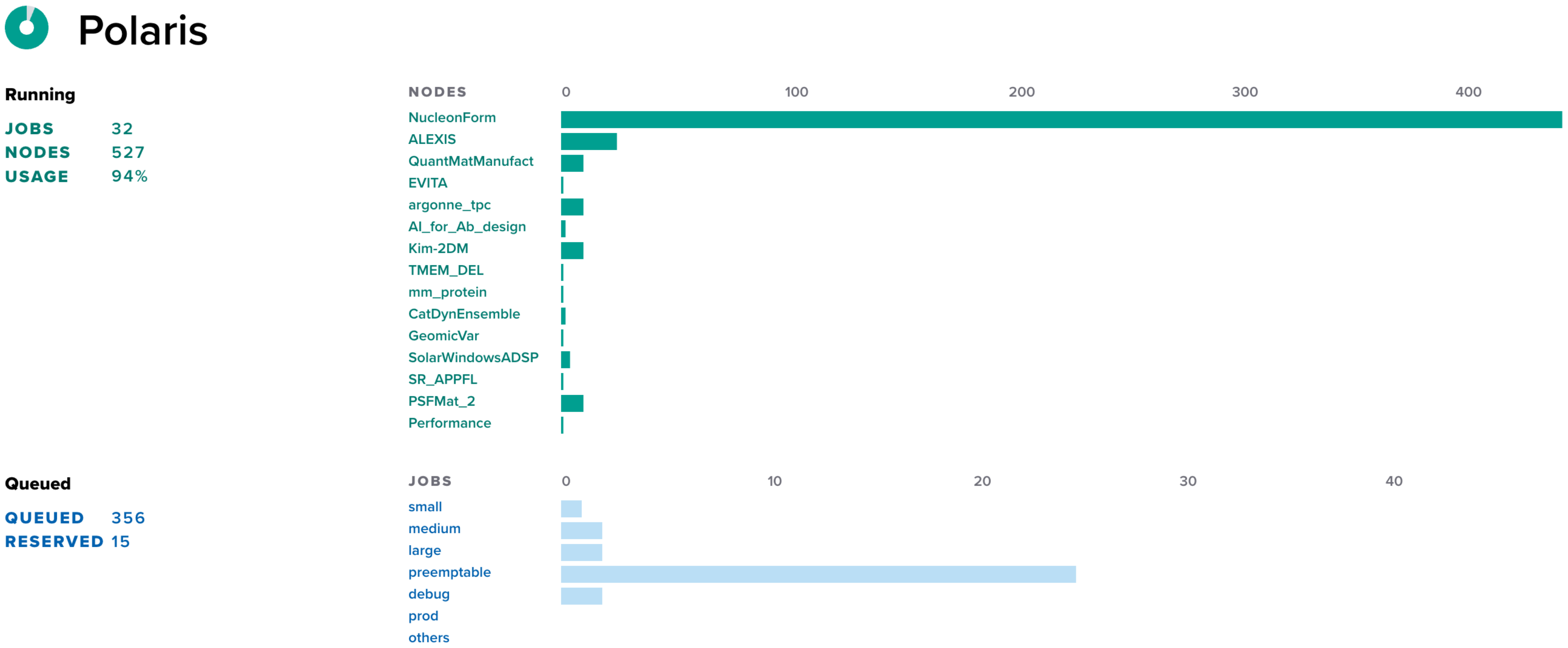
- Polaris had 3 main queues
  - https://docs.alcf.anl.gov/polaris/running-jobs/
  - debug
    - 2 nodes max
    - 1 hour max
    - 5 minutes min
  - debug-scaling
    - 10 nodes max
    - 1 hour max
    - 5 minutes min
  - prod
    - 10 nodes min
    - 496 nodes max
    - 5 minutes min
    - 24 hours max

Machine status web page



Polaris

JOBS	32
QUEUED	356
RESERVED	15
NODES	527
USAGE	94%



Polaris

Running

JOBS	32
NODES	527
USAGE	94%

Queued

QUEUED	356
RESERVED	15

https://alcf.anl.gov/support-center/machine-status

Cryptocard tips for ALCF systems

- The displayed value is a hex string. Type your PIN followed by all letters as CAPITALS.
- If you fail to authenticate the first time, you may have typed it incorrectly
  - Try again with the **same crypto string** (do NOT press button again)
- If you fail again, try a different ALCF host with a fresh crypto #
  - A successful login resets your count of failed logins
- Too many failed logins → your account locked
  - Symptom: You get password prompt but login denied even if it is correct
- Too many failed logins from a given IP → the IP will be blocked
  - Symptom: connection attempt by ssh or web browser will just time out

ALCF References

- Sample files
  - /eagle/ATPESC2024/EXAMPLES/track-0-getting-started/

- Online docs
  - https://www.alcf.anl.gov/support-center
  - https://docs.alcf.anl.gov/polaris/getting-started/
  - ALCF Polaris Beginners Guide (Instructions, examples, and videos)
    - https://github.com/argonne-lcf/ALCFBeginnersGuide/tree/master/polaris
    - https://www.alcf.anl.gov/support-center/training/getting-started-polaris-bootcamp

OLCF Odo System

Frontier System overview

- HPE Cray EX Supercomputer architecture
- 74 cabinets, 128 nodes per cabinet (9408 nodes)
- 3rd Gen AMD EPYC 64-core CPU
- 4 AMD Instinct MI250X GPUs
- HPE Slingshot interconnect
- Peak 1.206 Exaflops on HPL benchmark
- Cray, AMD, and GNU software stacks
- Orion - 679 PB multi-tier Lustre filesystem
- NFS storage (/ccs/home, /ccs/proj)

CRAY
OAK RIDGE National Laboratory
U.S. DEPARTMENT OF ENERGY
Hewlett Packard Enterprise
AMD
FRONTIER

Odo System overview

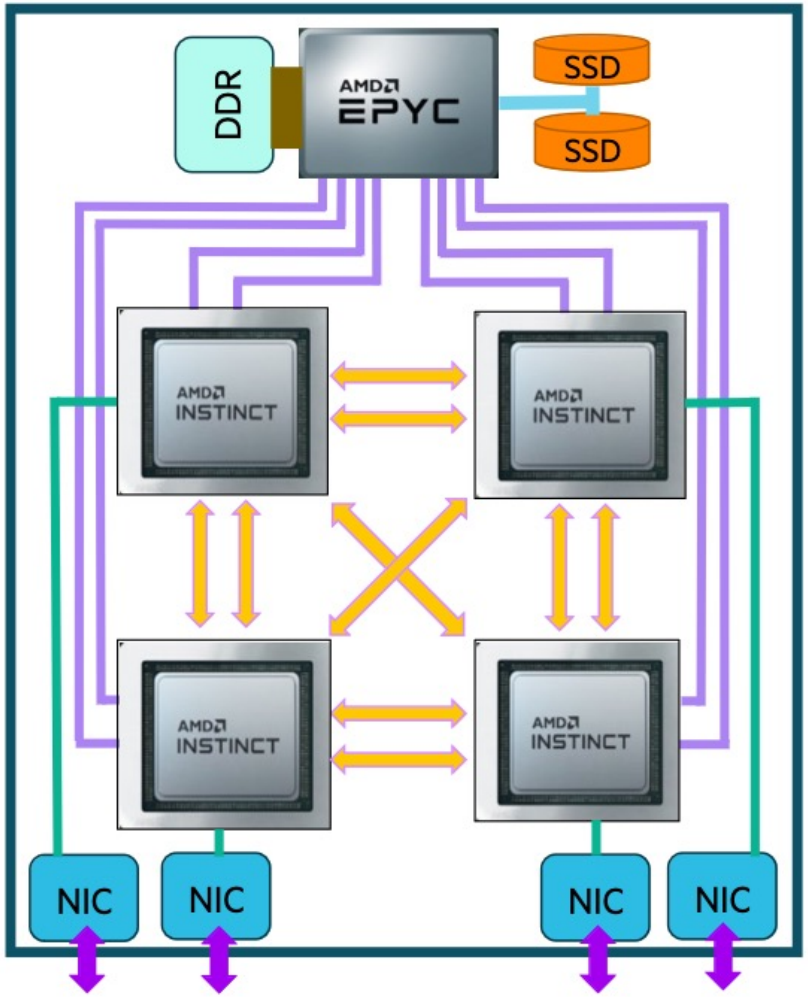
- HPE Cray EX Supercomputer architecture
- 74 cabinets, 128 nodes per cabinet (9408 nodes)

Odo is 30 Frontier nodes, and uses the GPFS filesystem (/gpfs/alpine2), and NFS on the Open Enclave for home areas (/ccsopen/home, /ccsopen/proj)

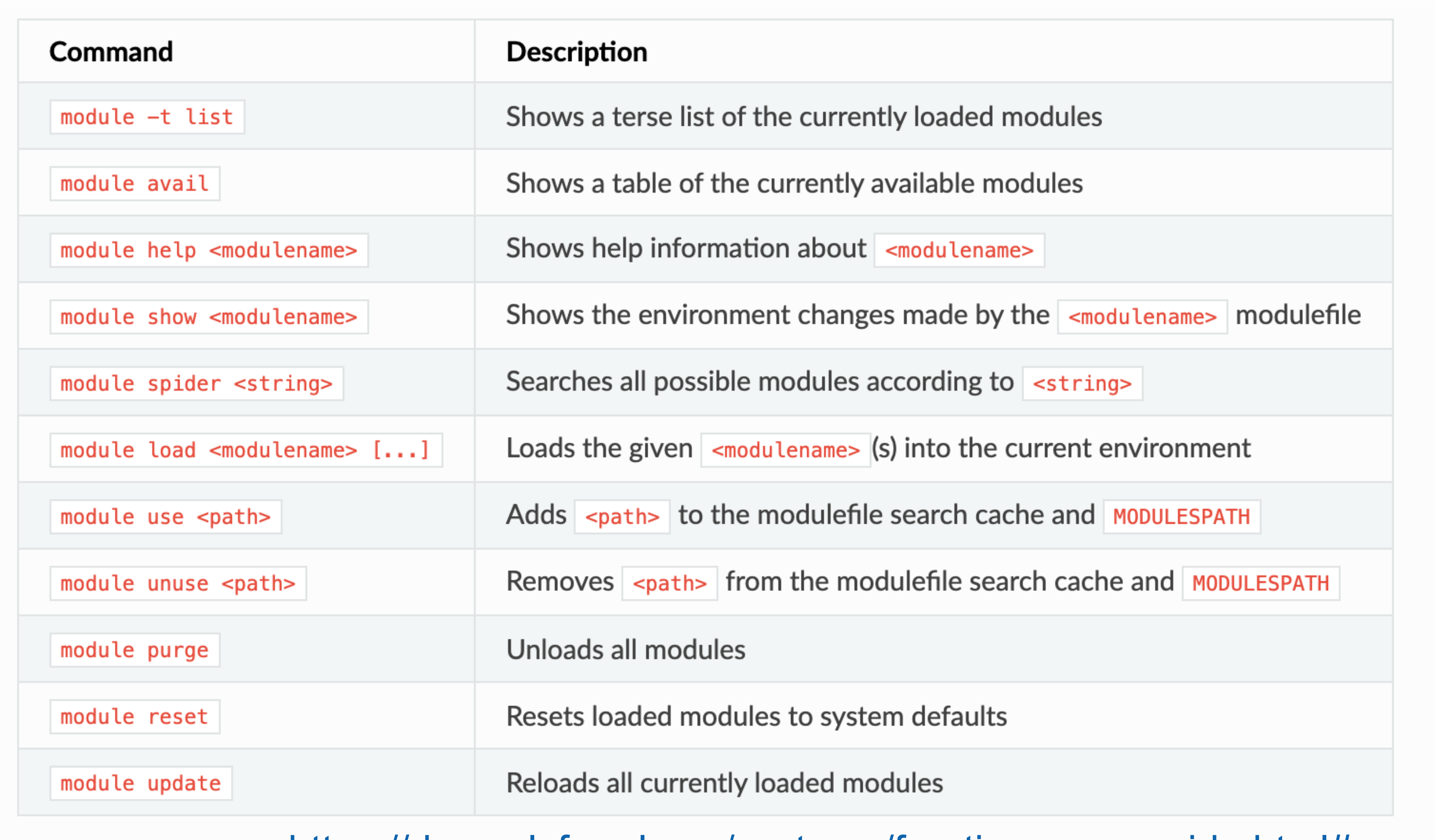
- Orion - 679 PB multi-tier Lustre filesystem
- NFS storage (/ccs/home, /ccs/proj)

Compute Node Configuration

- 1x AMD Optimized 3rd Gen EPYC 64 core processor
  - 2 hardware threads per physical core,
  - 2.0GHz base clock, 3.7GHz boost clock
- 512 GB DDR4 memory with 205 GB/s peak bandwidth
- 2x NVMe 2TB SSDs, peak 8 GB/s R, 4 GB/s W, >1.5M IOPs
- 4x AMD MI250X Instinct GPUs
  - 128 GB High-Bandwidth Memory (HBM2E)
  - 3.2 TB/s peak bandwidth
  - 53 TFLOPS double-precision peak for modeling & simulation
  - 2 Graphic Compute Dies (GCDs)
- AMD Infinity Fabric between CPU and GPUs
  - Peak host-to-device (H2D) and device-to-host (D2H) data transfers of 36+36 GB/s per link
- AMD Infinity Fabric between MI250Xs
  - Peak device-to-device bandwidth of 50+50 GB/s per link, low latency
- 4x HPE Slingshot Interconnect 200 GbE NICs
  - Provides 100 GB/s to other nodes, 25 GB/s per port



Odo Module Commands

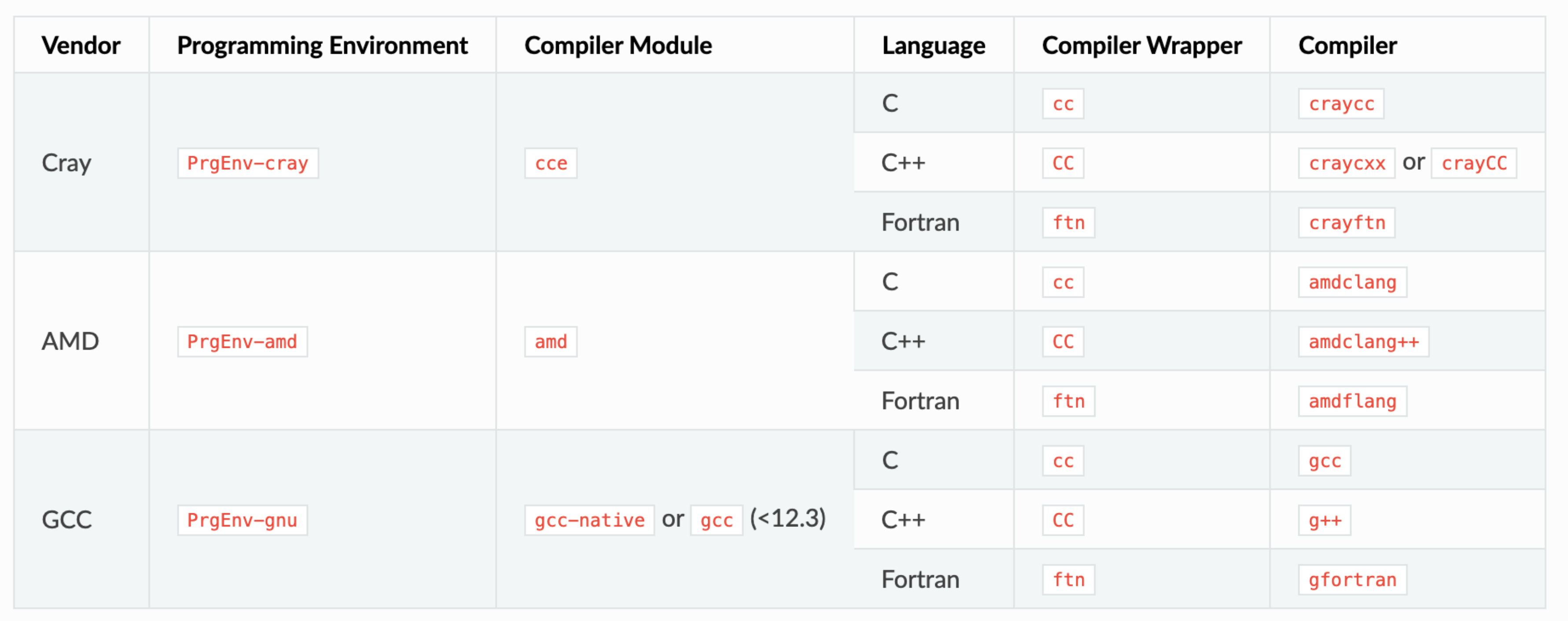


Command	Description
`module -t list`	Shows a terse list of the currently loaded modules
`module avail`	Shows a table of the currently available modules
`module help <modulename>`	Shows help information about `<modulename>`
`module show <modulename>`	Shows the environment changes made by the `<modulename>` modulefile
`module spider <string>`	Searches all possible modules according to `<string>`
`module load <modulename> [...]`	Loads the given `<modulename>`(s) into the current environment
`module use <path>`	Adds `<path>` to the modulefile search cache and `MODULESPATH`
`module unuse <path>`	Removes `<path>` from the modulefile search cache and `MODULESPATH`
`module purge`	Unloads all modules
`module reset`	Resets loaded modules to system defaults
`module update`	Reloads all currently loaded modules

https://docs.olcf.ornl.gov/systems/frontier_user_guide.html#general-usage

Odo Compiling

- Cray, AMD, and GCC compilers are provided through modules. The Cray and AMD compilers are both based on LLVM/Clang. There is also a system/OS versions of GCC available in /usr/bin. The table below lists details about each of the module-provided compilers.

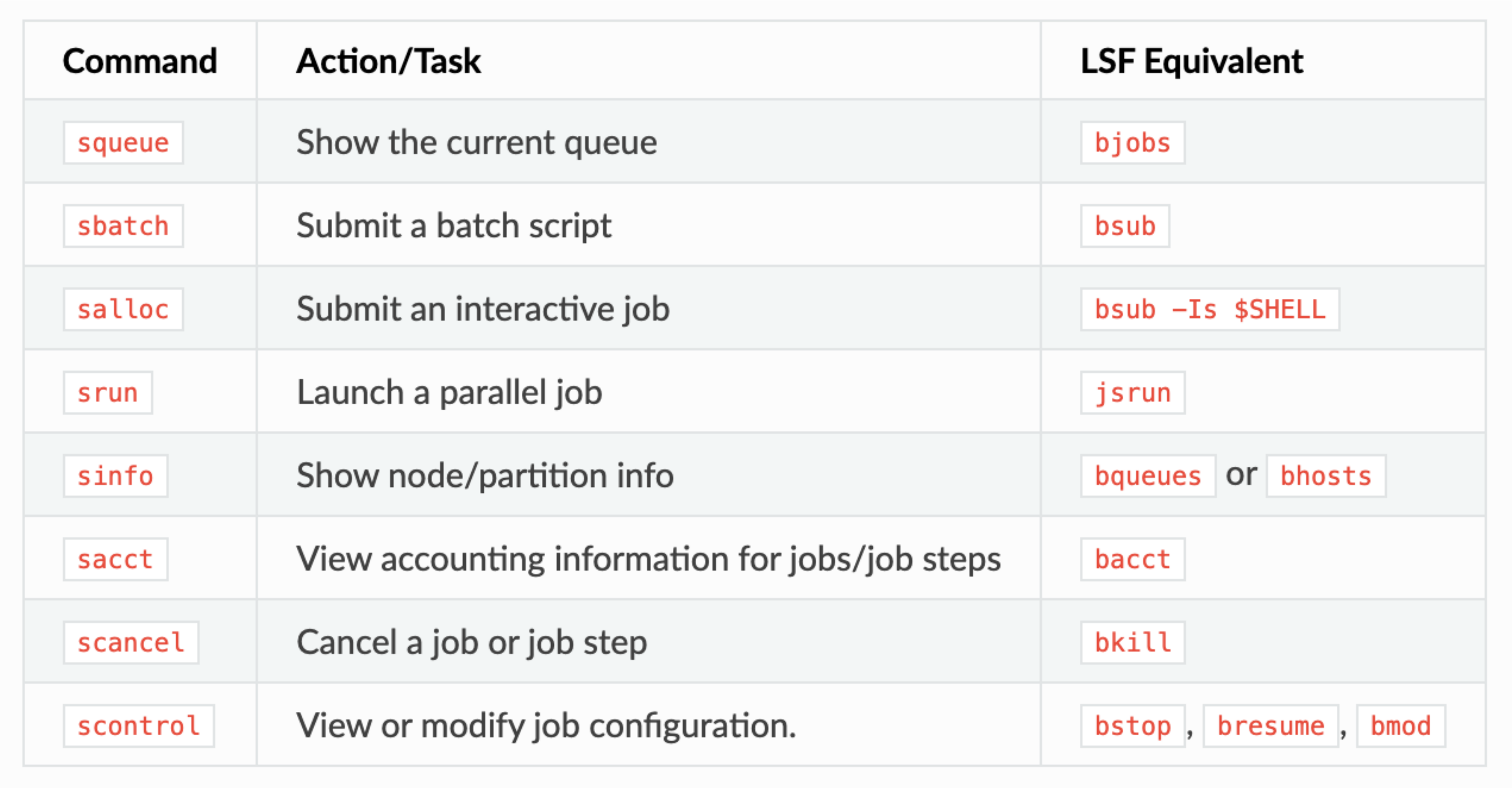


Vendor	Programming Environment	Compiler Module	Language	Compiler Wrapper	Compiler
Cray	`PrgEnv-cray`	`cce`	C	`cc`	`craycc`
			C++	`CC`	`craycxx` or `crayCC`
			Fortran	`ftn`	`crayftn`
AMD	`PrgEnv-amd`	`amd`	C	`cc`	`amdclang`
			C++	`CC`	`amdclang++`
			Fortran	`ftn`	`amdflang`
GCC	`PrgEnv-gnu`	`gcc-native` or `gcc` (<12.3)	C	`cc`	`gcc`
			C++	`CC`	`g++`
			Fortran	`ftn`	`gfortran`

- https://docs.olcf.ornl.gov/systems/frontier_user_guide.html#compiling

Odo Slurm Workload Manager

- Slurm Commands (vs. LSF commands)



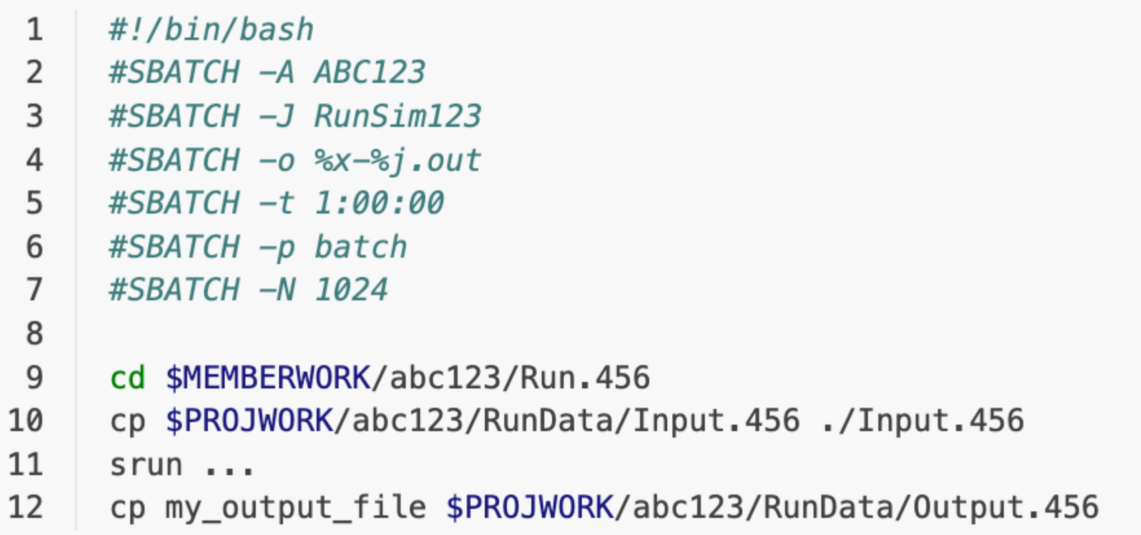
Command	Action/Task	LSF Equivalent
`squeue`	Show the current queue	`bjobs`
`sbatch`	Submit a batch script	`bsub`
`salloc`	Submit an interactive job	`bsub -Is $SHELL`
`srun`	Launch a parallel job	`jsrun`
`sinfo`	Show node/partition info	`bqueues` or `bhosts`
`sacct`	View accounting information for jobs/job steps	`bacct`
`scancel`	Cancel a job or job step	`bkill`
`scontrol`	View or modify job configuration.	`bstop` , `bresume` , `bmod`

- https://docs.olcf.ornl.gov/systems/frontier_user_guide.html#slurm

Odo Batch Scripts

- To submit a batch script,
  - Use the command, `sbatch myjob.sl`
- Description of the example

Line	Description
2	OLCF project to charge
3	Job name
4	Job standard output file (%x: the job name, %j: the Job ID)
5	Walltime requested (in HH:MM:SS format).
6	Partition (queue) to use
7	Number of compute nodes requested
9	Change into the run directory
10	Copy the input file into place
11	Run the job (add layout details)
12	Copy the output file to an appropriate location.



```
 1   #!/bin/bash
 2   #SBATCH -A ABC123
 3   #SBATCH -J RunSim123
 4   #SBATCH -o %x-%j.out
 5   #SBATCH -t 1:00:00
 6   #SBATCH -p batch
 7   #SBATCH -N 1024
 8
 9   cd $MEMBERWORK/abc123/Run.456
10   cp $PROJWORK/abc123/RunData/Input.456 ./Input.456
11   srun ...
12   cp my_output_file $PROJWORK/abc123/RunData/Output.456
```

- https://docs.olcf.ornl.gov/systems/frontier_user_guide.html#batch-scripts

Interactive job

- Useful for short tests or debugging

- Submit the job with `salloc`
  - `salloc -A trn028 -J RunSim123 -t 1:00:00 -p batch -N 1`

- Wait for job's shell prompt
  - Exit this shell to end your job
- From job's shell prompt, run just like in a script job,
  - `srun –N 1 -n 8 --ntasks-per-node=8 ./a.out`

- https://docs.olcf.ornl.gov/systems/frontier_user_guide.html#interactive-jobs

Monitoring and Modifying Jobs

- Primary commands
  - `scancel`: Cancel or Signal a Job
  - `squeue`: View the queue
    - `squeue -l` : Show all jobs currently in the queue
    - `squeue -l -u $USER` : Show all of your jobs currently in the queue
  - `scontrol show job`: get Detailed Job information
- https://docs.olcf.ornl.gov/systems/frontier_user_guide.html#monitoring-and-modifying-batch-jobs

Running MPI Applications

- Jobs run directly on the compute nodes. The `srun` command is used to execute an MPI binary on one or more compute nodes in parallel.
- `srun`

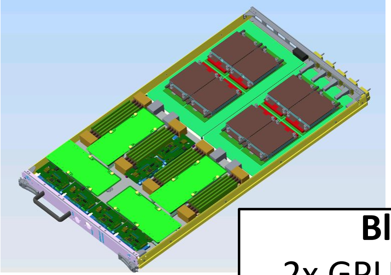
`-N`	Number of nodes
`-n`	Total number of MPI tasks
`-c`	Logical cores per MPI task
`--ntasks-per-node=<ntasks>`	A maximum count of tasks per node
`--gpus`	Specify the number of GPUs
`--gpu-per-node`	Specify the number of GPUs per node

- https://docs.olcf.ornl.gov/systems/frontier_user_guide.html#srun

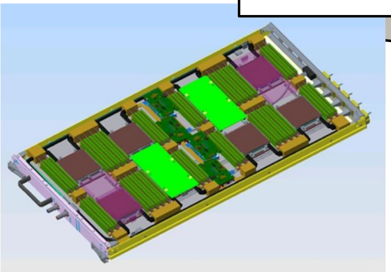
NERSC Perlmutter System

Perlmutter System Configuration

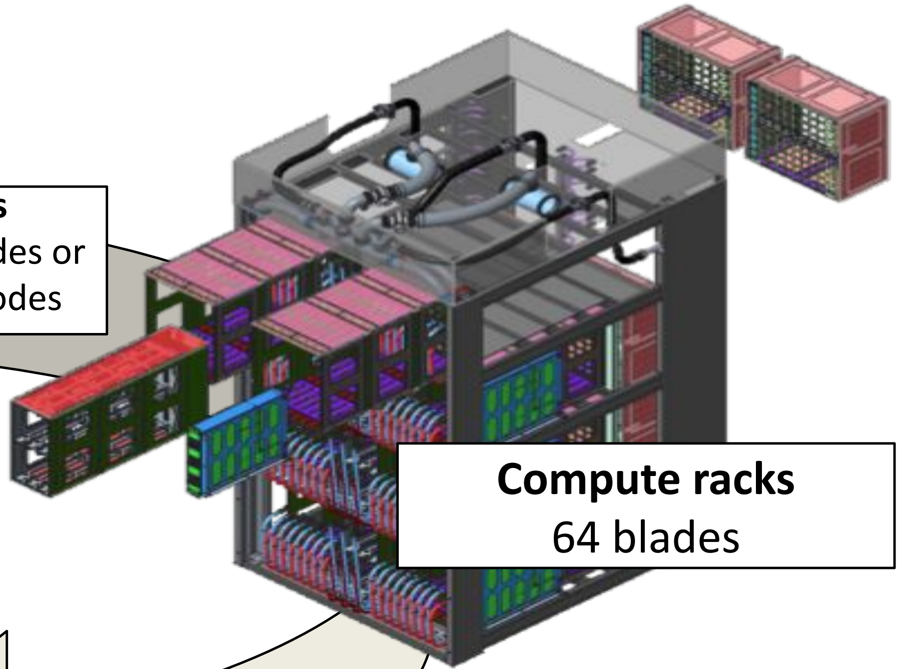
NVIDIA "Ampere" GPU Nodes
4x GPU + 1x CPU (>75 TF)
160 GiB HBM + DDR
4x 200G "Slingshot" NICs



AMD "Milan" CPU Node
2x CPUs
> 256 GiB DDR4
1x 200G "Slingshot" NIC



Blades
2x GPU nodes or
4x CPU nodes



Compute racks
64 blades

Centers of Excellence
Network
Storage
App. Readiness
System SW

→

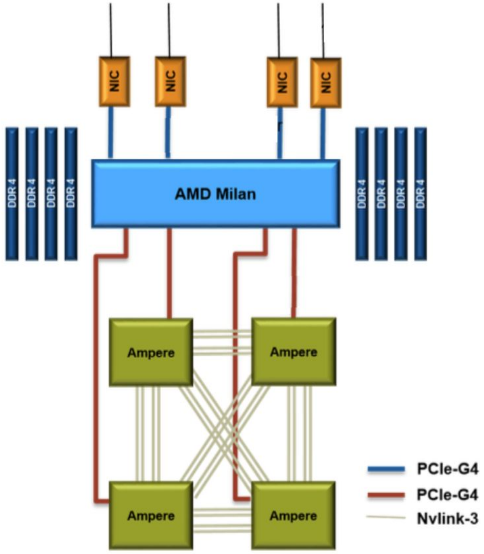
Perlmutter system
GPU racks
CPU racks
~6 MW

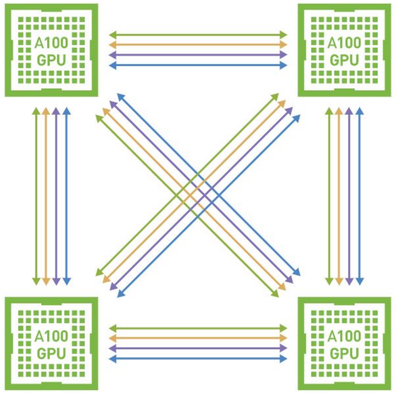
Perlmutter Nodes

GPU Nodes:

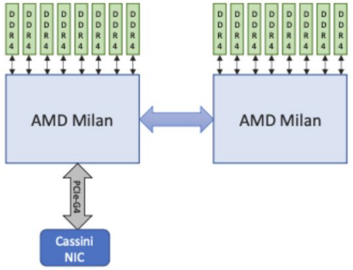
- Single AMD EPYC 7763 (Milan) CPU
- 64 cores per CPU
- Four NVIDIA A100 (Ampere) GPUs
- PCIe 4.0 GPU-CPU connection
- PCIe 4.0 NIC-CPU connection
- 4 HPE Slingshot 11 NICs
- 256 GB of DDR4 DRAM
- 40 GB of HBM per GPU with
- 1555.2 GB/s GPU memory bandwidth
- 204.8 GB/s CPU memory bandwidth
- 12 third generation NVLink links between each pair of gpus
- 25 GB/s/direction for each link

Data type	GPU TFLOPS
FP32	19.5
FP64	9.7
TF32 (tensor)	155.9
FP16 (tensor)	311.9
FP64 (tensor)	19.5





CPU Nodes:



- 2x AMD EPYC 7763 (Milan) CPUs
- 64 cores per CPU
- AVX2 instruction set
- 512 GB of DDR4 memory total
- 204.8 GB/s memory bandwidth per CPU
- 1x HPE Slingshot 11 NIC
- PCIe 4.0 NIC-CPU connection
- 39.2 GFlops per core
- 2.51 TFlops per socket
- 4 NUMA domains per socket (NPS=4)

Perlmutter Modules Environment

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

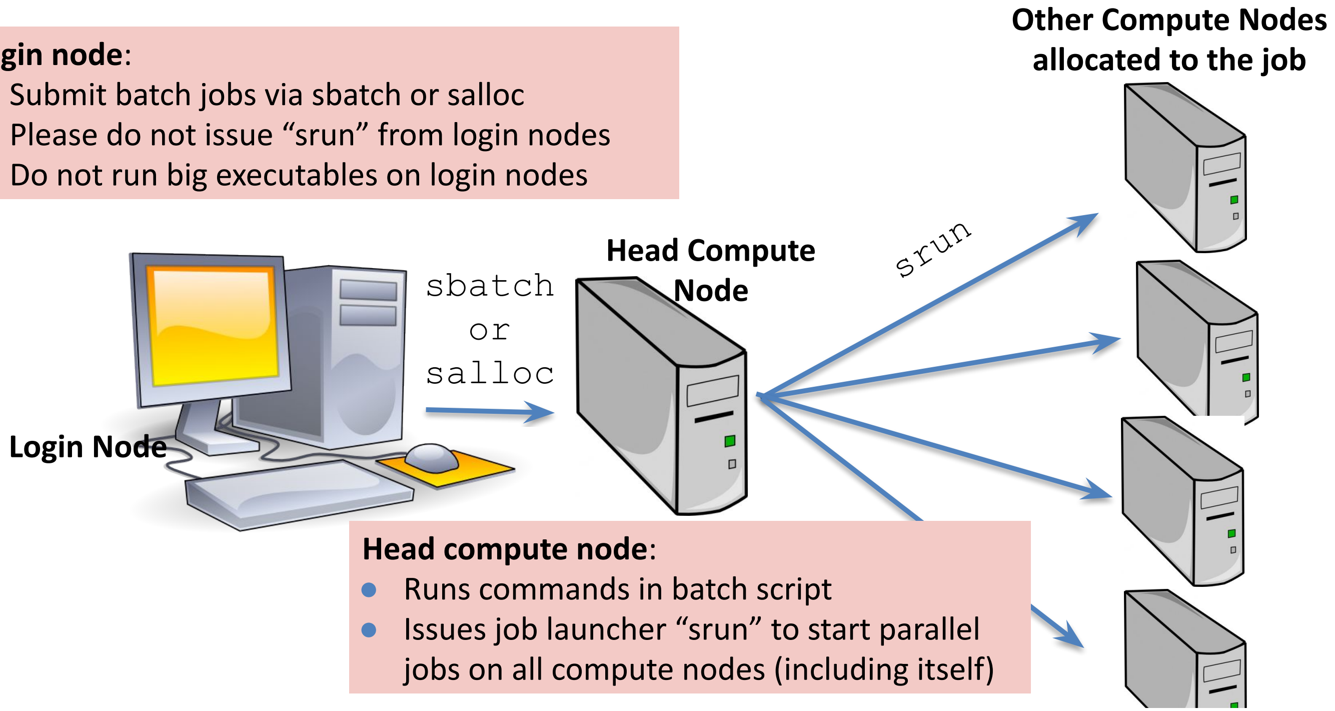
`module`	
`list`	To list the modules in your environment
`spider <name>`	To list available modules with <name> as substring, and how to load
`load/unload ..`	To load or unload module
`swap .. ..`	To swap modules
`show/display ..`	To see what a module loads, what env a module sets
`whatis ..`	Display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

Perlmutter 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.
- 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, …

Perlmutter: 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_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

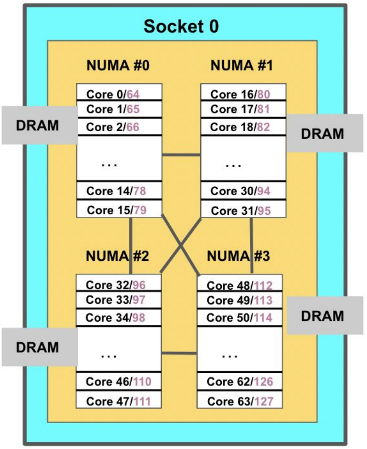
```
% salloc -N 2 -q interactive -C cpu -t 10:00
<wait_for_session_prompt. Land on a compute node>
% srun -n 64 ./helloWorld
```

Perlmutter: CPU 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)

tpn = Number of MPI tasks per node

CPU on Perlmutter GPU



- 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/

Perlmutter: Shared QOS for the reserved nodes

The “shared” QOS allows multiple executables from different users to share a node

To use nodes to be shared by multiple users, ATPESC attendees can request with salloc or sbatch with flags such as:

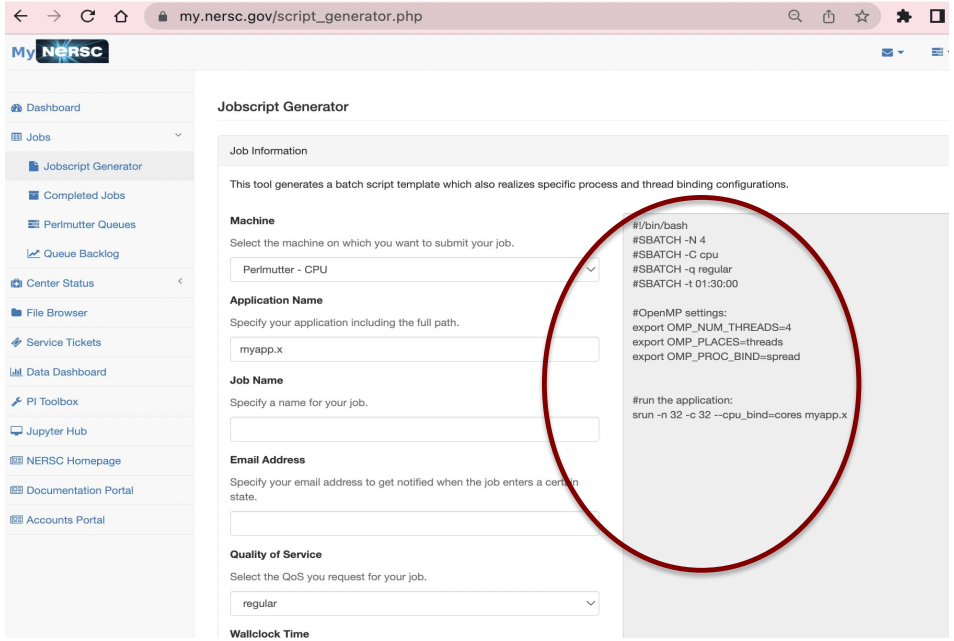
```
-C gpu -q shared -A ntrain5 -N 1 -c 32 -G 1 -t 60:00
```

Please notice the `-q shared` and `-c 32 -G1` options. It will get each user 1/4 of node CPU and 1 GPU. And users can run CPU or GPU jobs in this allocation.

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

Perlmutter Job script generator:

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



Jobscript Generator

Job Information

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

Machine
Select the machine on which you want to submit your job.
Perlmutter - CPU

Application Name
Specify your application including the full path.
myapp.x

Job Name
Specify a name for your job.

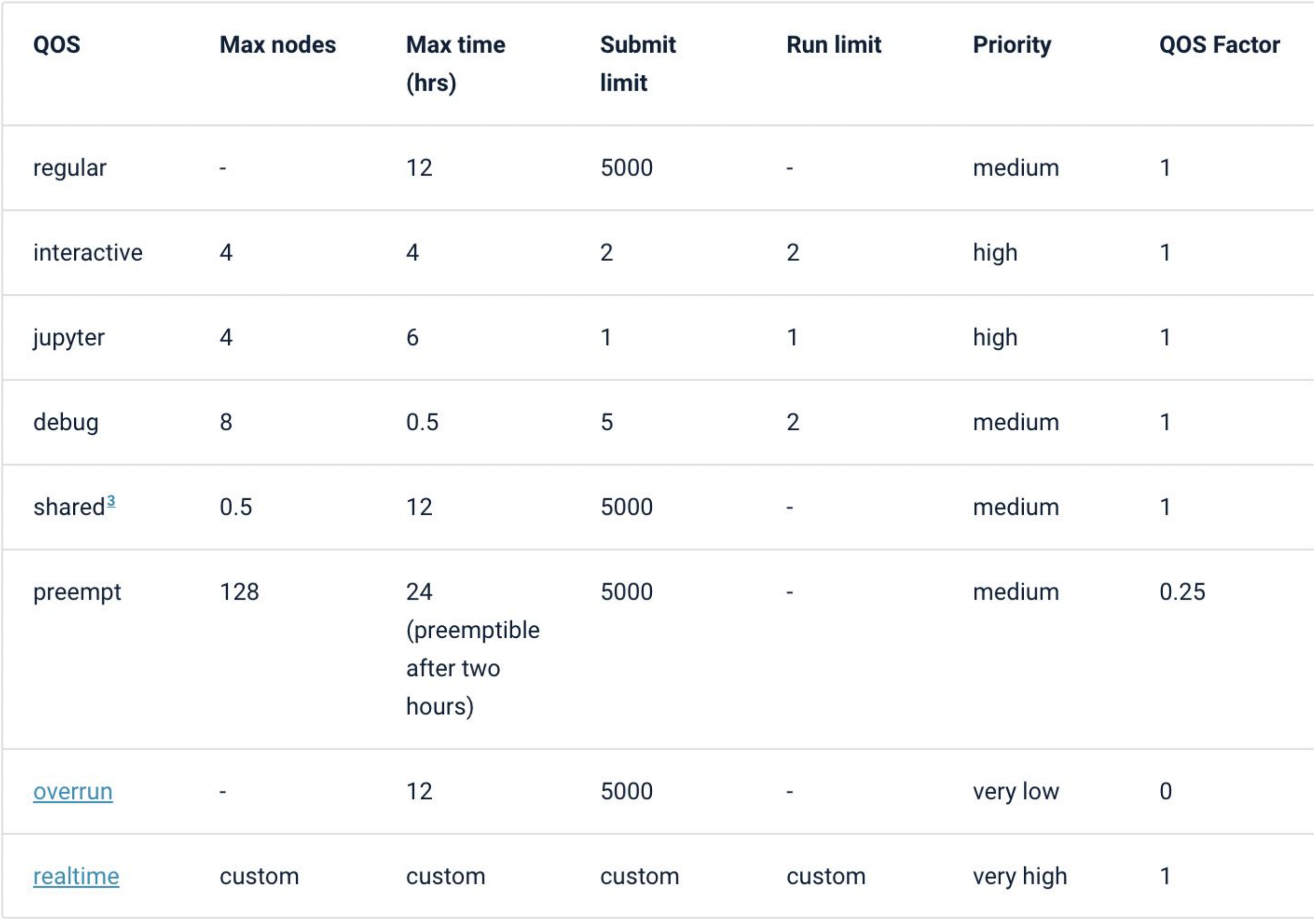
Email Address
Specify your email address to get notified when the job enters a certain state.

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
```

Perlmutter GPU Queue Policy



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

Perlmutter: 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

Intel Developer Cloud

Intel Developer Cloud

- https://www.intel.com/content/www/us/en/developer/tools/devcloud/overview.html
- Intel® Developer Cloud offers several configurations that are tuned to various workloads. From AI and inference training to FPGA development to edge prototyping and preproduction deployment, you can use the environment that best matches your business needs.
- Available Intel Hardware
  - Intel Xeon Max CPU Series
  - Intel Data Center GPU Flex Series
  - Intel Data Center GPU Max Series
  - Intel Gaudi AI Accelerator
- During ATPESC 2024, Intel Data Center GPU Max 1100 GPUs will be used
  - https://www.intel.com/content/www/us/en/products/sku/232876/intel-data-center-gpu-max-1100/specifications.html
  - 56 Xe cores, 448 Xe Vector Engines, TDP 300W, 48 GB HBM2e memory
  - A smaller PVC variant than Aurora PVC (Intel Data Center GPU Max 1550)

Sharing SSH Key to get access

- Generate an SSH Key
  - https://console.cloud.intel.com/docs/guides/ssh_keys.html#generate-an-ssh-key
  - Linux OS
    - Launch a Terminal on your local system
    - To generate an SSH key, copy and paste the following to your Terminal.
      - `ssh-keygen -t rsa -b 4096 -f ~/.ssh/id_rsa`
    - If you’re prompted to overwrite, select No.
    - Copy and paste this command in your Terminal to show the generated SSH key.
      - `cat ~/.ssh/id_rsa.pub`
  - Windows PowerShell
    - Launch a new PowerShell window on your local system.
    - Optional: If you haven't generated a key before, create an .ssh directory.
      - `mkdir $env:UserProfile\.ssh`
    - Copy and paste the following to your terminal to generate SSH Keys
      - `ssh-keygen -t rsa -b 4096 -f $env:UserProfile\.ssh\id_rsa`
    - If you are prompted to overwrite, select No.
    - Copy and paste this command to show the generated SSH key.
      - `cat $env:UserProfile\.ssh\id_rsa.pub`
- Add your SSH Key to the following google sheet:
  - https://docs.google.com/spreadsheets/d/1tUGD799Y2ECpRH2kMcYVwfxf_1HI67Zmf86OcpUT9Kk/edit?gid=0#gid=0

Cheat Sheets

ATPESC Resources

ALCF – Polaris

-**Project name:** **ATPESC2024**

-**Note:** use your ALCF Username. The password will be your old/newly established PIN + token code displayed on the token.

-**Support:** ALCF staff available to help you **via slack**!! and support@alcf.anl.gov

-**Reservations:** Please check the details of the reservations directly on Polaris (**command**: *pbs_rstat* on polaris)

-**Queue**

- -**Polaris (check *pbs_rstat*), or default for running without reservation**

-**User guide:** **https://docs.alcf.anl.gov/polaris/getting-started/**

ATPESC Resources

OLCF – Odo

-**Project name:** **trn028**

-**Support:** help@olcf.ornl.gov

-**Queue: running without reservation**

-**User guide:**

-**Frontier User Guide:** **https://docs.olcf.ornl.gov/systems/frontier_user_guide.html#frontier-user-guide**

-**Odo User Guide:** **https://docs.olcf.ornl.gov/systems/odo_user_guide.html**

ATPESC Resources

NERSC – Perlmutter

- -**Project name:** **ntrain5**
- -**Support:** help desk
- -**Queue: running without reservation**
- -**User guide:** **https://docs.nersc.gov/**

ATPESC Resources

Cloud resources for Tools track

- Intel Developer Cloud
  - Test performance on Intel Data Center GPU Max 1100
  - Add your ssh key to the following googld sheet in order to get access
    - https://docs.google.com/spreadsheets/d/1tUGD799Y2ECpRH2kMcYVwfxf_1HI67Zmf86OcpUT9Kk/edit?usp=sharing
  - Login instruction will be shared before the tools session

- [Optional] AMD Accelerator Cloud (TBD)

Questions?

- *Use this presentation as a reference during ATPESC!*

- Supplemental info will be posted as well

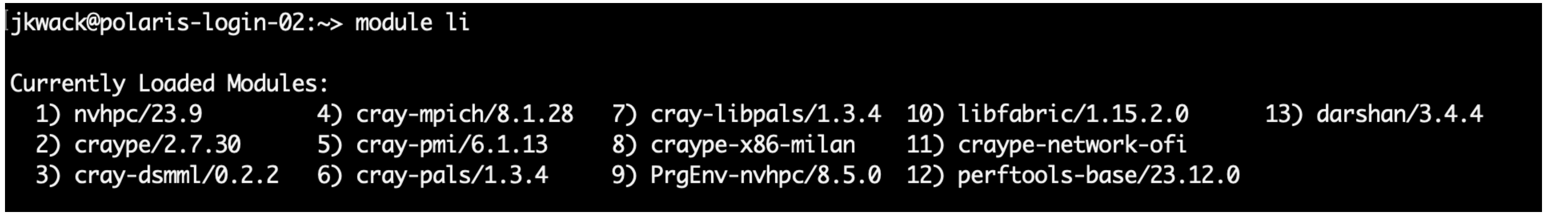
Hands-on Exercises

Hands-on exercise

- Polaris hands-on
- Odo hands-on
- Perlmutter hands-on

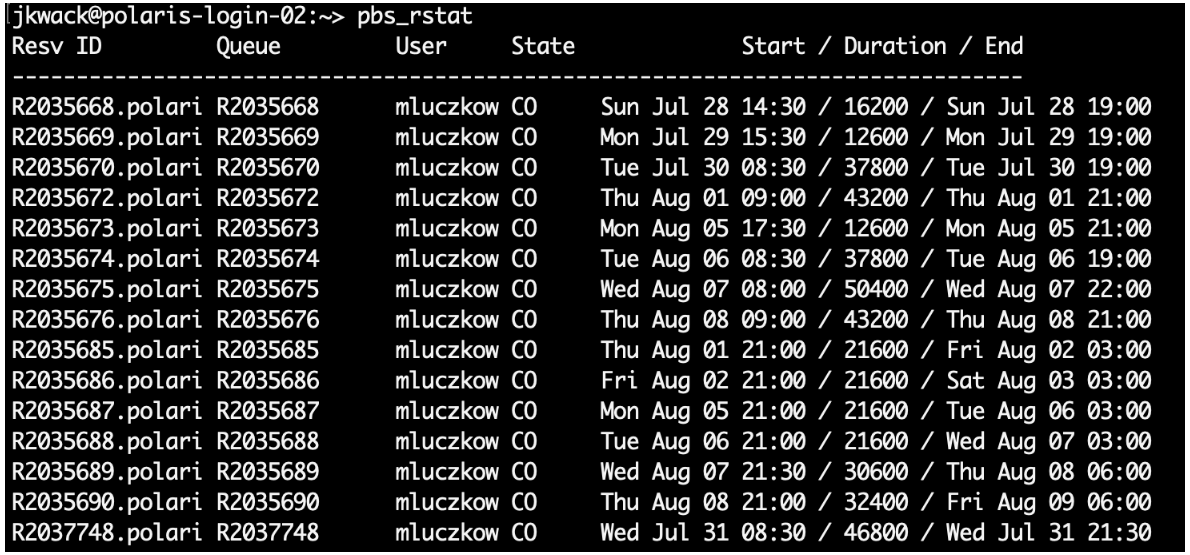
Hands-on exercise: Polaris

```
$ ssh -Y {your_username}@polaris.alcf.anl.gov                # Login to Polaris
$ module avail                                               # See available modules
$ module list                                                # See loaded modules
```



```
jkwack@polaris-login-02:~> module li

Currently Loaded Modules:
  1) nvhpc/23.9          4) cray-mpich/8.1.28   7) cray-libpals/1.3.4  10) libfabric/1.15.2.0      13) darshan/3.4.4
  2) craype/2.7.30       5) cray-pmi/6.1.13     8) craype-x86-milan    11) craype-network-ofi
  3) cray-dsmml/0.2.2    6) cray-pals/1.3.4     9) PrgEnv-nvhpc/8.5.0  12) perftools-base/23.12.0
```

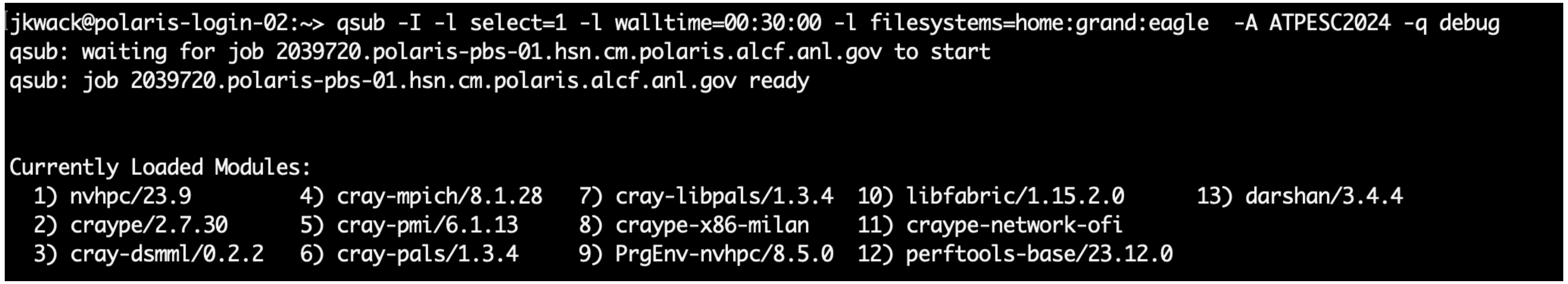
```
$ qstat -u ${USER}                                           # To see your jobs
$ pbs_rstat                                                  # Check reservation
```



```
jkwack@polaris-login-02:~> pbs_rstat
Resv ID         Queue         User     State             Start / Duration / End
-------------------------------------------------------------------------------
R2035668.polari R2035668      mluczkow CO     Sun Jul 28 14:30 / 16200 / Sun Jul 28 19:00
R2035669.polari R2035669      mluczkow CO     Mon Jul 29 15:30 / 12600 / Mon Jul 29 19:00
R2035670.polari R2035670      mluczkow CO     Tue Jul 30 08:30 / 37800 / Tue Jul 30 19:00
R2035672.polari R2035672      mluczkow CO     Thu Aug 01 09:00 / 43200 / Thu Aug 01 21:00
R2035673.polari R2035673      mluczkow CO     Mon Aug 05 17:30 / 12600 / Mon Aug 05 21:00
R2035674.polari R2035674      mluczkow CO     Tue Aug 06 08:30 / 37800 / Tue Aug 06 19:00
R2035675.polari R2035675      mluczkow CO     Wed Aug 07 08:00 / 50400 / Wed Aug 07 22:00
R2035676.polari R2035676      mluczkow CO     Thu Aug 08 09:00 / 43200 / Thu Aug 08 21:00
R2035685.polari R2035685      mluczkow CO     Thu Aug 01 21:00 / 21600 / Fri Aug 02 03:00
R2035686.polari R2035686      mluczkow CO     Fri Aug 02 21:00 / 21600 / Sat Aug 03 03:00
R2035687.polari R2035687      mluczkow CO     Mon Aug 05 21:00 / 21600 / Tue Aug 06 03:00
R2035688.polari R2035688      mluczkow CO     Tue Aug 06 21:00 / 21600 / Wed Aug 07 03:00
R2035689.polari R2035689      mluczkow CO     Wed Aug 07 21:30 / 30600 / Thu Aug 08 06:00
R2035690.polari R2035690      mluczkow CO     Thu Aug 08 21:00 / 32400 / Fri Aug 09 06:00
R2037748.polari R2037748      mluczkow CO     Wed Jul 31 08:30 / 46800 / Wed Jul 31 21:30
```

Hands-on exercise: Polaris

```
$ qsub -I -l select=1 -l walltime=00:30:00 -l filesystems=home:grand:eagle  -A ATPESC2024 -q R2035668
```



```
jkwack@polaris-login-02:~> qsub -I -l select=1 -l walltime=00:30:00 -l filesystems=home:grand:eagle  -A ATPESC2024 -q debug
qsub: waiting for job 2039720.polaris-pbs-01.hsn.cm.polaris.alcf.anl.gov to start
qsub: job 2039720.polaris-pbs-01.hsn.cm.polaris.alcf.anl.gov ready


Currently Loaded Modules:
  1) nvhpc/23.9        4) cray-mpich/8.1.28   7) cray-libpals/1.3.4  10) libfabric/1.15.2.0      13) darshan/3.4.4
  2) craype/2.7.30     5) cray-pmi/6.1.13     8) craype-x86-milan    11) craype-network-ofi
  3) cray-dsmml/0.2.2  6) cray-pals/1.3.4     9) PrgEnv-nvhpc/8.5.0  12) perftools-base/23.12.0
```

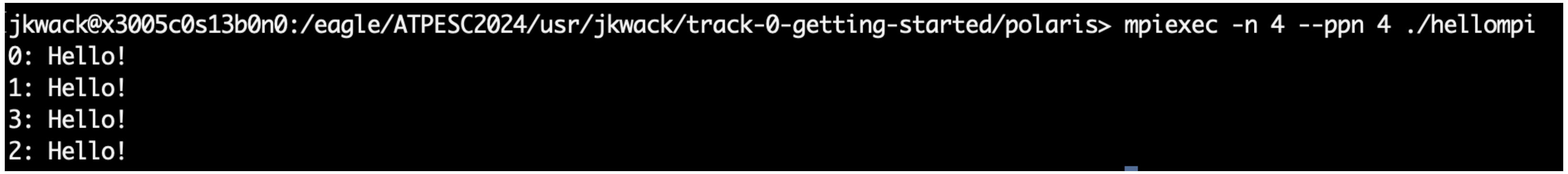
```
$ cd /eagle/ATPESC2024/usr/

$ mkdir $USER

$ cd $USER

$ cp -rf /eagle/ATPESC2024/EXAMPLES/track-0-getting-started .

$ cd track-0-getting-started/polaris/

$ more Makefile
    …
    CC=cc

    hellompi: hellompi.c
          which $(CC)
          $(CC) -g -O0 -o hellompi hellompi.c
    …
```

Hands-on exercise: Polaris

```
$ cat submit.sh
#!/bin/bash
#PBS -l select=1
#PBS -l walltime=00:30:00
#PBS -l filesystems=home:grand:eagle
#PBS -A ATPESC2024
#PBS -q R2035668

cd $PBS_O_WORKDIR
mpiexec -n 4 --ppn 4 ./hellompi
status=$?

echo "mpiexec status is $status"
exit $status
```

Hands-on exercise: Polaris

```
$ cc -o hellompi hellompi.c             # Build the example
$ make clean; make                      # Another way to build the example


$ mpiexec -n 4 --ppn 4 ./hellompi
```



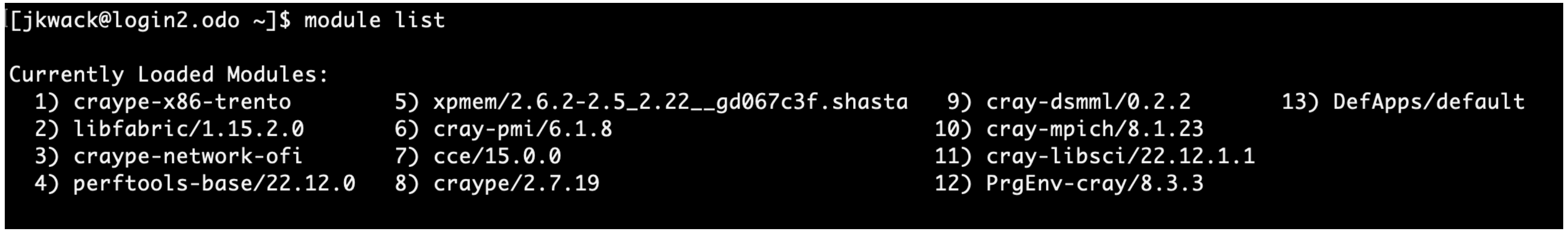
```
jkwack@x3005c0s13b0n0:/eagle/ATPESC2024/usr/jkwack/track-0-getting-started/polaris> mpiexec -n 4 --ppn 4 ./hellompi
0: Hello!
1: Hello!
3: Hello!
2: Hello!
```

More references for Polaris

- https://github.com/argonne-lcf/ALCFBeginnersGuide/tree/master/polaris
- https://www.alcf.anl.gov/support-center/training/getting-started-polaris-bootcamp
- https://docs.alcf.anl.gov/polaris/getting-started/

Hands-on exercise: Odo

```
$ ssh -Y {your_username}@odo.olcf.ornl.gov          # Login to Odo

$ module avail                                      # See available modules

$ module list                                       # See loaded modules
```

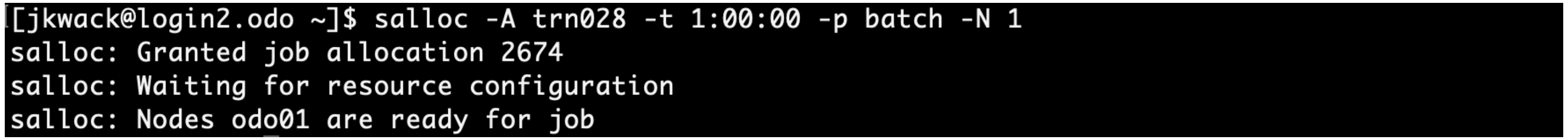


```
[jkwack@login2.odo ~]$ module list

Currently Loaded Modules:
  1) craype-x86-trento        5) xpmem/2.6.2-2.5_2.22__gd067c3f.shasta   9) cray-dsmml/0.2.2       13) DefApps/default
  2) libfabric/1.15.2.0       6) cray-pmi/6.1.8                         10) cray-mpich/8.1.23
  3) craype-network-ofi       7) cce/15.0.0                             11) cray-libsci/22.12.1.1
  4) perftools-base/22.12.0   8) craype/2.7.19                          12) PrgEnv-cray/8.3.3
```

```
$ squeue -l -u $USER                                     # To see your jobs
```

Hands-on exercise: Odo

```
$ salloc -A trn028 -t 1:00:00 -p batch -N 1
```

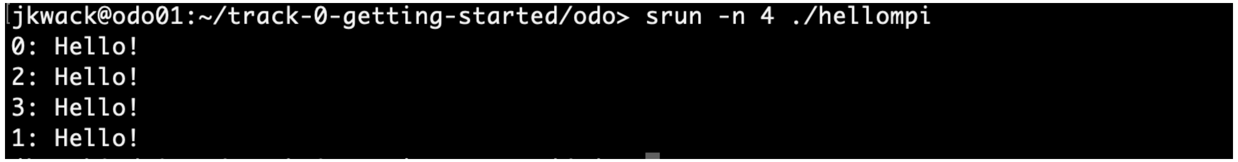


```
[jkwack@login2.odo ~]$ salloc -A trn028 -t 1:00:00 -p batch -N 1
salloc: Granted job allocation 2674
salloc: Waiting for resource configuration
salloc: Nodes odo01 are ready for job
```

```
$ cp -r /ccsopen/proj/trn028/track-0-getting-started .
$ cd track-0-getting-started/odo/
$ more Makefile
    …
    CC=cc

    hellompi: hellompi.c
           which $(CC)
           $(CC) -g -O0 -o hellompi hellompi.c
    …
```

Hands-on exercise: Odo

```
$ more submit.sh
     #!/bin/bash
     #SBATCH -p batch
     #SBATCH -A trn028
     #SBATCH -N 1
     #SBATCH -t 60:00

     srun -n 8  ./hellompi
     status=$?

     echo "mpiexec status is $status"
     exit $status
$ cc -o hellompi hellompi.c                    # Build the example
$ make clean; make                             # Another way to build the example
$ srun -n 4 ./hellompi
```

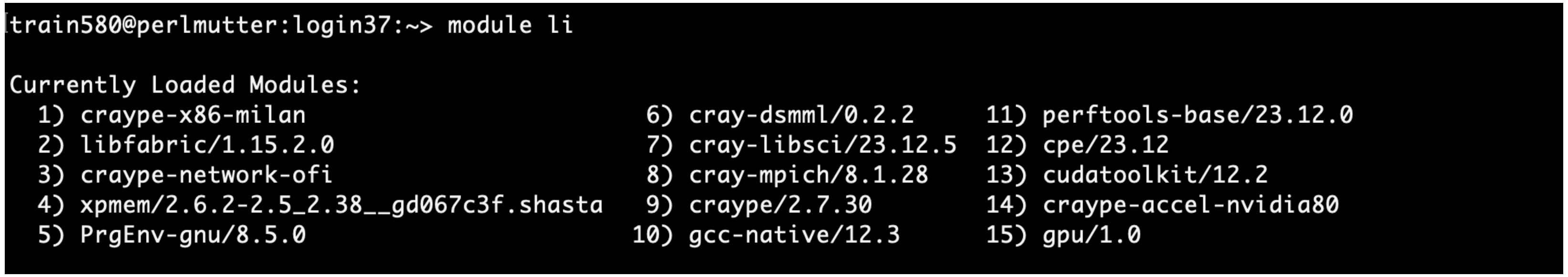


```
jkwack@odo01:~/track-0-getting-started/odo> srun -n 4 ./hellompi
0: Hello!
2: Hello!
3: Hello!
1: Hello!
```

- More references for Odo and Frontier
  - https://docs.olcf.ornl.gov/systems/odo_user_guide.html
  - https://docs.olcf.ornl.gov/systems/frontier_user_guide.html

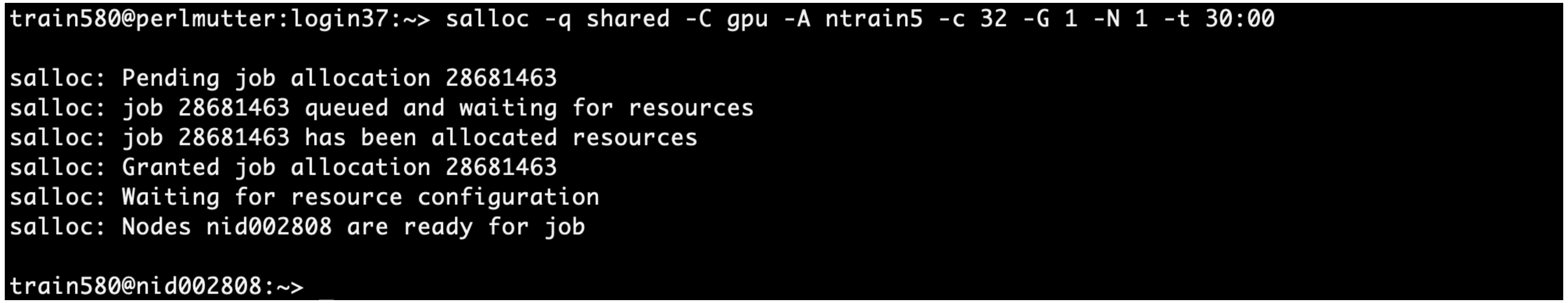
Hands-on exercise: Perlmutter

```
$ ssh -Y {your_username}@perlmutter.nersc.gov                          # Login to Perlmutter
$ module avail                                                         # See available modules
$ module list                                                          # See loaded modules
```



```
train580@perlmutter:login37:~> module li

Currently Loaded Modules:
  1) craype-x86-milan                          6) cray-dsmml/0.2.2     11) perftools-base/23.12.0
  2) libfabric/1.15.2.0                        7) cray-libsci/23.12.5  12) cpe/23.12
  3) craype-network-ofi                        8) cray-mpich/8.1.28    13) cudatoolkit/12.2
  4) xpmem/2.6.2-2.5_2.38__gd067c3f.shasta     9) craype/2.7.30        14) craype-accel-nvidia80
  5) PrgEnv-gnu/8.5.0                         10) gcc-native/12.3      15) gpu/1.0
```

```
$ sqs                                                                  # To see your jobs
$ salloc -q shared -C gpu -A ntrain5 -c 32 -G 1 -N 1 -t 30:00
```



```
train580@perlmutter:login37:~> salloc -q shared -C gpu -A ntrain5 -c 32 -G 1 -N 1 -t 30:00

salloc: Pending job allocation 28681463
salloc: job 28681463 queued and waiting for resources
salloc: job 28681463 has been allocated resources
salloc: Granted job allocation 28681463
salloc: Waiting for resource configuration
salloc: Nodes nid002808 are ready for job

train580@nid002808:~>
```

Hands-on exercise: Perlmutter

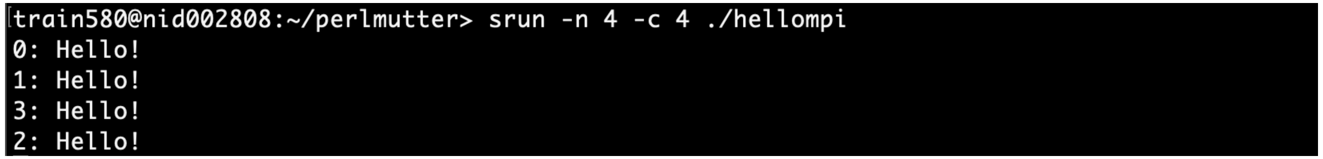
```
$ cp -r /global/homes/t/train580/track-0-getting-started/perlmutter .

$ cd perlmutter/


$ more Makefile
    …
    CC=cc

    hellompi: hellompi.c
         which $(CC)
         $(CC) -g -O0 -o hellompi hellompi.c
    …
```

Hands-on exercise: Perlmutter

```
$ more submit.sh
#!/bin/bash
#SBATCH -q shared
#SBATCH -C gpu
#SBATCH -A ntrain5
#SBATCH -G 1
#SBATCH -N 1
#SBATCH -t 60:00

srun -n 4 -c 4 ./hellompi
status=$?

echo "mpiexec status is $status"
exit $status


$ sbatch submit.sh
```

Hands-on exercise: Perlmutter

```
$ cc -o hellompi hellompi.c                 # Build the example
$ make clean; make                          # Another way to build the example


$ srun -n 4 –c 4 ./hellompi
```



```
train580@nid002808:~/perlmutter> srun -n 4 -c 4 ./hellompi
0: Hello!
1: Hello!
3: Hello!
2: Hello!
```

```
$ nvidia-smi
```

More references for Perlmutter

- https://docs.nersc.gov/jobs/

Thank you!

Supplemental Info

-

ARGONNE

ATPESC2024

EXTREME-SCALE COMPUTING

ARGONNE TRAINING PROGRAM ON EXTREME-SCALE COMPUTING

Produced by Argonne National Laboratory, a U.S. Department of Energy Laboratory managed by UChicagoArgonne, LLC under contract DE-AC02-06CH11357.

Special thanks to the National Energy Research Scientific Computing Center (NERSC) and Oak Ridge Leadership Computing Facility (OLCF) for the use of their resources during the training event.