Argonne Training Program on Extreme-Scale Computing (ATPESC)

Quick Start on ATPESC Computing Resources

JaeHyuk Kwack
Argonne National Laboratory

Date 07/30/2023

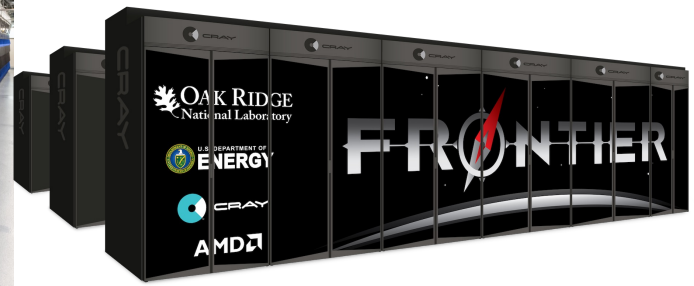
NNSA National Nuclear Security Administration
U.S. DEPARTMENT OF ENERGY | Office of Science
Argonne NATIONAL LABORATORY

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).

Argonne NATIONAL LABORATORY

OAK RIDGE National Laboratory



Argonne NATIONAL LABORATORY

ECP EXASCALE COMPUTING PROJECT

Leadership Computing Facility System

	Argonne LCF			Oak Ridge LCF	
System	Cray XC40	HPE	HPE	IBM	HPE
Name	Theta	Polaris	Aurora (in 2023)	Summit	Frontier
Compute nodes	4,392	560	10,624	4,608	9,408
Node architecture	1 x Intel Knights Landing, 64 cores	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	4,392 KNL processors	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)	192 GB DDR4 + 16 GB MCDRAM	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)	11.69	44	> 2 Exaflops DP	200	1.6 Exaflop DP

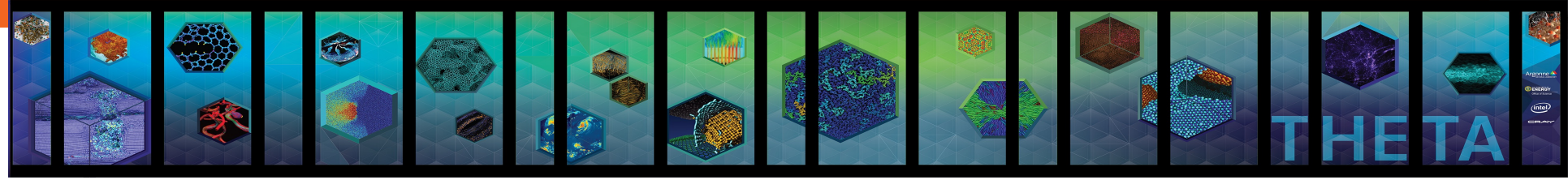
AVAILABLE RESOUCE FOR ATPESC

- ALCF Systems
  - Intel KNL processors (Theta)
  - AMD CPUs + NVIDIA A100 GPUs (ThetaGPU)
  - Intel CPUs + NVIDIA K80 GPUs (Cooley)
  - AMD CPUs + NVIDIA A100 GPUs (Polaris)
- OLCF
  - IBM Power9 CPUs + NVIDIA V100 GPUs (Ascent)
- NERSC
  - AMD CPUs + NVIDIA A100 GPUs (Perlmutter)
- Cloud resources
  - Intel DevCloud
  - AMD Accelerator Cloud (AAC)

ALCF Systems

- ***Theta - Cray* XC40**
  - 4,392 nodes, each with
    - 64 core Intel Knights Landing processor
    - 192 DDR4 memory with 16 GB MCDRAM
- ***ThetaGPU – NVIDIA DGX A100***
  - 24 DGX A100 nodes, each with
    - Two AMD Rome 64-core processors
    - Eight NVIDIA A100 GPUs with 40 GB HBM2 per GPU
    - 1 TB DDR4 memory
- ***Cooley (visualization & data analysis)* – Cray CS**
  - 126 nodes, each with
    - Two Intel Xeon E5-2620 Haswell 2.4 GHz 6-core processors
    - NVIDIA Tesla K80 graphics processing unit with 24 GB memory
    - 384 GB DDR4 memory
- ***Polaris – HPE***
  - 560 nodes, each with
    - One AMD Milan 7513P CPU
    - Four NVIDIA A100 GPUs with 40 GB HBM2 per GPU
    - 512 GB DDR4 memory

Theta



Theta serves as a bridge to the exascale system coming to Argonne

- Serves as a bridge between Mira and Aurora, transition and data analytics system
- Cray XC40 system. Runs Cray software stack
- 11.69 PF peak performance
- 4392 nodes with 2nd Generation Intel® Xeon Phi™ processor
  - Knights Landing (KNL), 7230 SKU 64 cores 1.3GHz
  - 4 hardware threads/core
- 192GB DDR4 memory 16GB MCDRAM on each node
- 128GB SSD on each node
- Cray Aries high speed interconnect in dragonfly topology
- Initial file system: 10PB Lustre file system, 200 GB/s throughput

Theta - Filesystems

- Lustre
  - Home directories (/home) are in /lus/swift/home
    - Default quota 50GiB
    - Your home directory is backed up
  - Project directory locations (/grand) in /lus/grand/projects
    - Theta, ThetaGPU, Cooley, Polaris: **/grand/ATPESC2023**
      - **CREATE A SUBDIRECTORY /grand/ATPESC2023/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**

Theta – Modules (Polaris, Theta, ThetaGPU)

- 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*

Theta - Compilers

- For all compilers (Intel, Cray, Gnu, etc):
  - **Use:** cc, CC, ftn
  - **Do not use** mpicc, MPICC, mpic++, mpif77, mpif90
    - *they do not generate code for the compute nodes*
- Selecting the compiler you want using **"module swap"** or **"module unload"** followed by **"module load"**
  - Intel
    - PrgEnv-intel *This is the default*
  - Cray
    - module swap PrgEnv-intel PrgEnv-cray
    - **NOTE:** links libsci by default
  - Gnu
    - module swap PrgEnv-intel PrgEnv-gnu
  - Clang/LLVM
    - module swap PrgEnv-intel PrgEnv-llvm

Theta - Job script

```
#!/bin/bash
#COBALT -t 10
#COBALT –n 2
#COBALT -A  ATPESC2023

# Various env settings are provided by Cobalt
echo $COBALT_JOBID  $COBALT_PARTNAME  $COBALT_JOBSIZE

aprun -n 16 -N 8 -d 1 -j 1 -cc depth ./a.out
status=$?

# could do another aprun here...

exit $status
```

Theta - aprun overview

- Start a parallel execution (equivalent of *mpirun*, *mpiexec* on other systems)
  - *Must be invoked from within a batch job that allocates nodes to you!*
- Options
  - -n *total_number_of_ranks*
  - -N *ranks_per_node*
  - -d *depth* [number of cpus (hyperthreads) per rank]
  - -cc **depth** [Note: **depth** is a keyword]
  - -j *hyperthreads* [cpus (hyperthreads) per compute unit (core)]
- Env settings you may need
  - -e OMP_NUM_THREADS=*nthreads*
  - -e KMP_AFFINITY=...
- See also **man aprun**

Submitting a Cobalt job

- qsub –A <project> -q <queue> -t <time> -n <nodes> ./jobscript.sh

  E.g.

  qsub –A Myprojname –q default –t 10 –n 32 ./jobscript.sh

- If you specify your options in the script via #COBALT, then just:
  - qsub jobscript.sh
- Make sure jobscript.sh is executable
- Without "-q", submits to the queue named "**default**"
  - For ATPESC reservations, specify e.g. "–q **ATPESC2023**" (see ***showres*** output)
  - For small tests outside of reservations, use e.g. "-q debug-cache-quad"
- ***Theta "default" (production) queue has 128 node minimum job size***
  - The ATPESC reservation does not have this restriction
- **man qsub** for more options

Managing your job

- qstat – show what's in the queue
  - qstat –u <username> # Jobs only for user
  - qstat <jobid> # Status of this particular job
  - qstat –fl <jobid> # Detailed info on job
- qdel <jobid>
- showres – show reservations currently set in the system
- **man qstat** for more options

Cobalt files for a job

- Cobalt will create 3 files per job, the basename <prefix> defaults to the jobid, but can be set with “qsub -O myprefix”
  - jobid can be inserted into your string e.g. "-O myprefix_$jobid"

- **Cobalt log file**: **<prefix>.cobaltlog**
  - created by Cobalt when job is submitted, additional info written during the job
  - contains submission information from qsub command, runjob, and environment variables
- **Job stderr file**: **<prefix>.error**
  - created at the start of a job
  - contains job startup information and any content sent to standard error while the user program is running
- **Job stdout file**: **<prefix>.output**
  - contains any content sent to standard output by user program

Interactive job

- Useful for short tests or debugging
- Submit the job with –I (letter I for Interactive)
  - Default queue and default project
    - qsub –I –n 32 –t 30
  - Specify queue and project:
    - qsub –I –n 1 –t 30 –q ATPESC2023 –A ATPESC2023
- Wait for job's shell prompt
  - *This is a new shell* with env settings e.g. COBALT_JOBID
  - Exit this shell to end your job
- From job's shell prompt, run just like in a script job, e.g. on Theta
  - aprun –n 512 –N 16 –d 1 –j 1 –cc depth ./a.out
- After job expires, apruns will fail. *Check* **qstat $COBALT_JOBID**

Core files and debugging

- Abnormal Termination Processing (ATP)
  - Set environment **ATP_ENABLED=1** in your job script before aprun
  - On program failure, generates a merged stack backtrace tree in file **atpMergedBT.dot**
  - View the output file with the program **stat-view** (module load stat)

- Notes on linking your program
  - make sure you load the "atp" module before linking
    - to check, *module list*

- Other debugging tools
  - You can generate STAT snapshots asynchronously
  - Full-featured debugging with DDT
  - More info at
    - https://www.alcf.anl.gov/sites/default/files/2020-05/Loy-comp_perf_workshop-debugging-2020-v1.2.pdf

Machine status web page

Running Jobs

Queued Jobs

Reservations

http://status.alcf.anl.gov/theta/activity (a.k.a. The Gronkulator)

ALCF ThetaGPU (x86+GPU)

- ThetaGPU is an extension of Theta and is comprised of 24 NVIDIA DGX A100 nodes for training artificial intelligence (AI) datasets, while also enabling GPU-specific and -enhanced high-performance computing (HPC) applications for modeling and simulation.

- Machine Specs
  - Architecture: AMD Rome CPU
  - Peak Performance: 3.8 petaflops
  - Processors per node: Two 64-core
  - GPU per node: 8 NVIDIA A100
  - Nodes: 24
  - Cores: 3,072
  - Number of GPUs: 192
  - Memory: 24 TB
  - GPU memory: 7.68 TB
  - Interconnect: 20 Mellanox QM9700 HDR200 40-port switches wired in a fat-tree topology

ThetaGPU - Environment

- ThetaGPU Login nodes
  - `$ ssh thetagpusn1` (or `$ ssh thetagpusn2`) from the Theta login nodes
- Use module commands on thetaGPU login nodes
- Module examples
  - openmpi for mpi
  - nvhpc for NVIDIA OpenMP compilers
- Update your .bashrc and .bash_profile as follows:

```
$ cat ~/.bashrc
# .bashrc
# Source global definitions
if [ -f /etc/bashrc ]
then
    . /etc/bashrc
elif [ -f /etc/bash.bashrc ]
then
    . /etc/bash.bashrc
fi
```

```
$ cat ~/.bash_profile
# .bash_profile
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi
# proxy settings
export HTTP_PROXY=http://theta-proxy.tmi.alcf.anl.gov:3128
export HTTPS_PROXY=http://theta-proxy.tmi.alcf.anl.gov:3128
export http_proxy=http://theta-proxy.tmi.alcf.anl.gov:3128
export https_proxy=http://theta-proxy.tmi.alcf.anl.gov:3128
```

- ALL bash jobscripts must also begin with `#!/bin/bash -l`(that's a lower-case L)

ThetaGPU Job Script

- More like a typical Linux cluster
- Job script
  - Example test.sh:

```
#!/bin/bash -l
NODES=`cat $COBALT_NODEFILE | wc -l`
PROCS=$((NODES * 16))
mpirun -n $PROCS myprog.exe
```

  - Submit on 1 node/gpu for 30 minutes

```
qsub -n 1 -t 30 -q training-gpu -A ATPESC2023 ./test.sh     # For 1 node with 8 GPUs
qsub -n 1 -t 30 -q single-gpu -A ATPESC2023 ./test.sh       # For 1 GPU
```

  - Submit on 1 node/gpu for 30 minutes for an interactive job

```
qsub -I -n 1 -t 30 -q training-gpu -A ATPESC2023            # For 1 node with 8 GPUs
qsub -I -n 1 -t 30 -q single-gpu -A ATPESC2023              # For 1 GPU
```

  - Refer to online user guide for more info
    - https://www.alcf.anl.gov/support-center/theta-gpu-nodes

ALCF Cooley (x86+GPU)

- Cooley, the ALCF's visualization cluster, enables users to analyze and visualize large-scale datasets, helping them to gain deeper insights into simulations and data generated on the facility's supercomputers.

- Machine Specs
  - Architecture: Intel Haswell
  - Peak Performance: 293 teraflops
  - Processors per node: Two 6-core, 2.4-GHz Intel E5-2620
  - GPU per node: 1 NVIDIA Tesla K80
  - Nodes: 126
  - Cores: 1,512
  - Memory: 47 TB
  - GPU memory: 3 TB
  - Interconnect: FDR InfiniBand network
  - Racks: 6

Cooley - Softenv (Cooley)

- Similar to **modules** package
- Keys are read at login time to set environment variables like PATH.
  - Cooley: ~/.soft.cooley
- To get started:

```
# This key selects Intel compilers to be used by mpi wrappers
+openmpi-2.1.5-intel
+intel-composer-xe
@default
# the end – do not put any keys after the @default
```

- After edits to .soft, type "resoft" or log out and back in again

Cooley Job Script

- More like a typical Linux cluster
- Job script
  - Example test.sh:

```
#!/bin/sh
NODES=`cat $COBALT_NODEFILE | wc -l`
PROCS=$((NODES * 12))
mpirun -n $PROCS myprog.exe
```

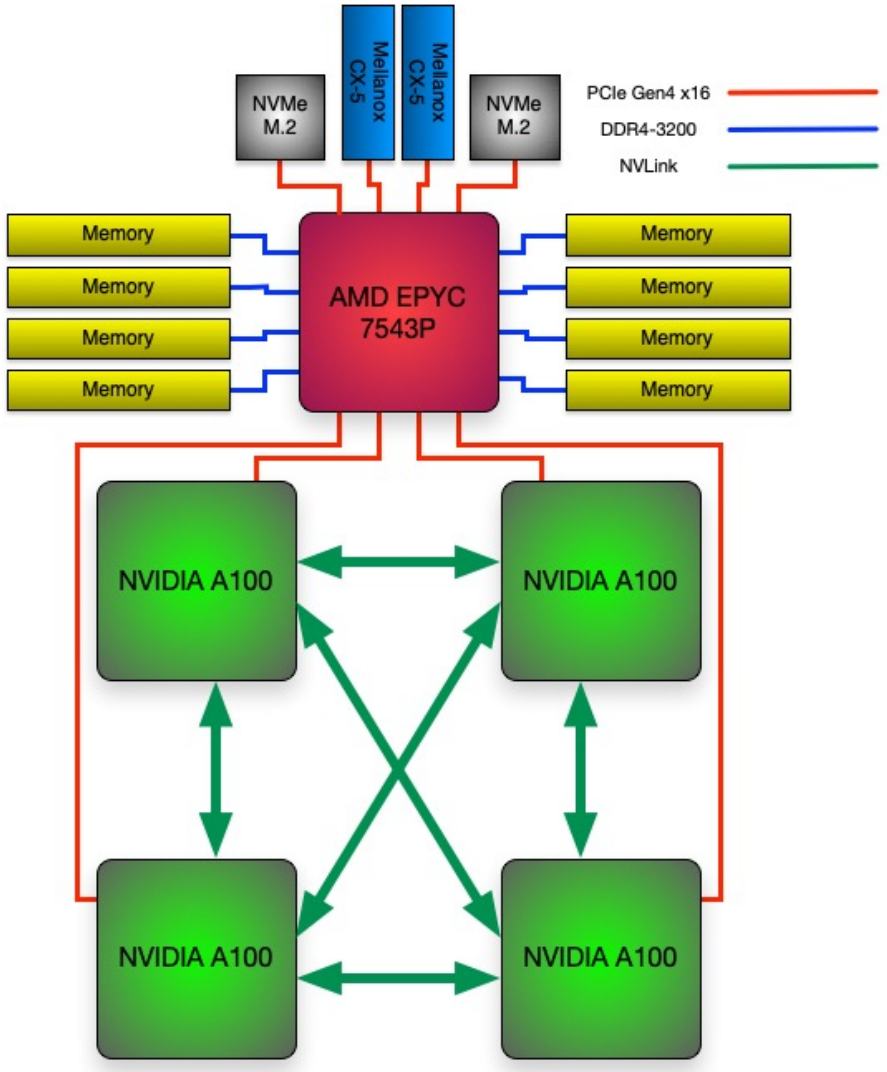
  - Submit on 5 nodes for 10 minutes

```
qsub -n 5 -t 10 –q training -A ATPESC2023 ./test.sh
```

  - Refer to online user guide for more info

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



Argonne NATIONAL LABORATORY

ECP EXASCALE COMPUTING PROJECT

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+

Argonne NATIONAL LABORATORY

ECP EXASCALE COMPUTING PROJECT

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

- Two parts to 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>
```

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

Polaris Queues

- Polaris had 3 main queues
  - https://www.alcf.anl.gov/support/user-guides/polaris/queueing-and-running-jobs/job-and-queue-scheduling/index.html
  - debug
    - 2 nodes max
    - 1 hour max
    - 10 minutes min
  - debug-scaling
    - 10 nodes max
    - 1 hour max
    - 10 minutes min
  - prod
    - 10 nodes min
    - 496 nodes max
    - 30 minutes min
    - Up to 6/12/24 hours max
    - Queue will route job to other queues

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
`--hostfile`	Indicate file with hostname

- Full list of options available from the man page
- https://www.alcf.anl.gov/support/user-guides/polaris/queueing-and-running-jobs/example-job-scripts/index.html

ALCF References

- Sample files (Polaris, Theta, ThetaGPU, Cooley)
  - /grand/ATPESC2023/EXAMPLES/track-0-getting-started/GettingStarted
  - /eagle/ATPESC2023/EXAMPLES/track-0-getting-started/GettingStarted
- Online docs
  - https://www.alcf.anl.gov/support-center
  - 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-assets/getting-started-bootcamp
  - Getting Started Presentations (*slides and videos*)
    - Theta and Cooley
    - https://www.alcf.anl.gov/workshops/2019-getting-started-videos
  - Debugging:
    - https://www.alcf.anl.gov/sites/default/files/2020-05/Loy-comp_perf_workshop-debugging-2020-v1.2.pdf

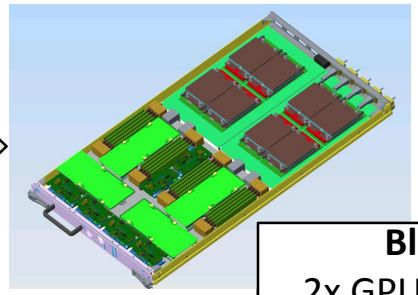
Cryptocard tips for ALCF systems

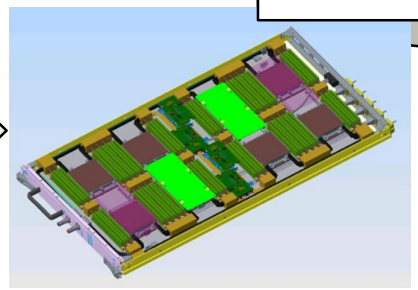
- The displayed value is a hex string. Type your PIN followed by all letters as CAPITALS.
- If you fail to authenticate the first time, you may have typed it incorrectly
  - Try again with the **same crypto string** (do NOT press button again)
- If you fail again, try a different ALCF host with a fresh 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

Perlmutter System Configuration

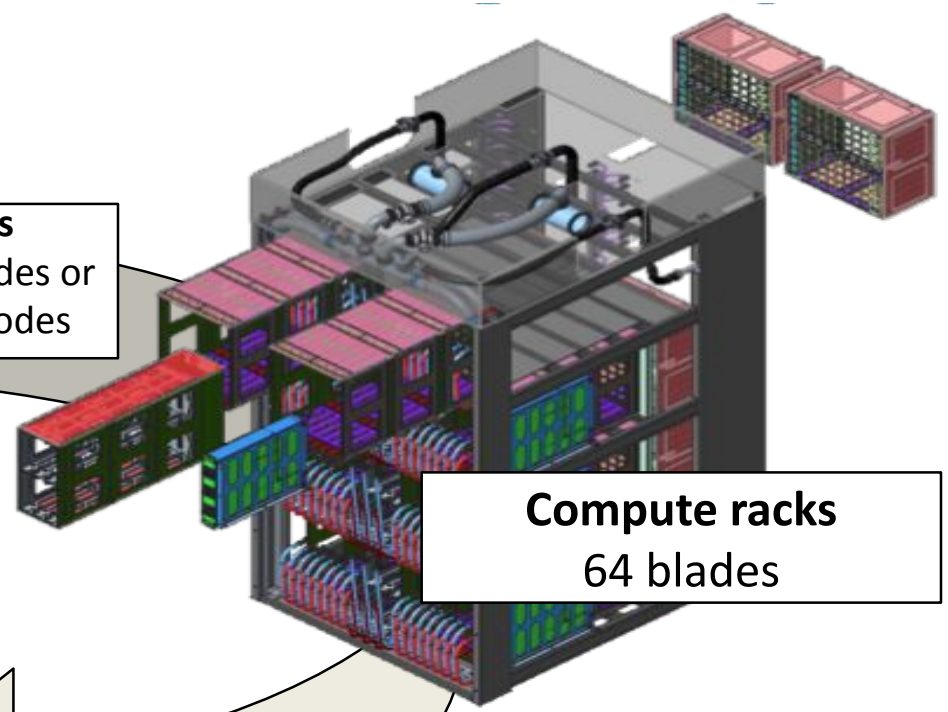
NVIDIA "Ampere" GPU Nodes
4x GPU + 1x CPU (>75 TF)
160 GiB HBM + DDR
4x 200G "Slingshot" NICs

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

Argonne NATIONAL LABORATORY

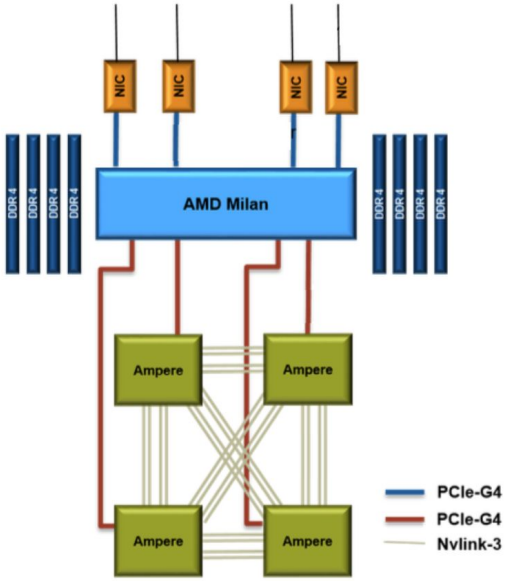
ECP EXASCALE COMPUTING PROJECT

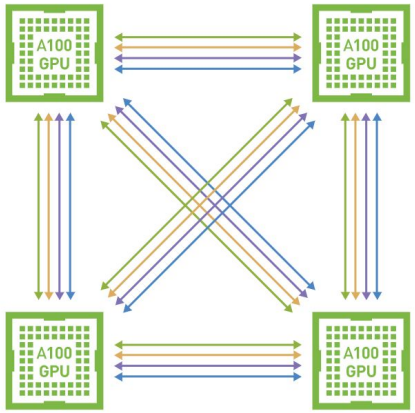
Perlmutter Nodes

GPU Nodes:

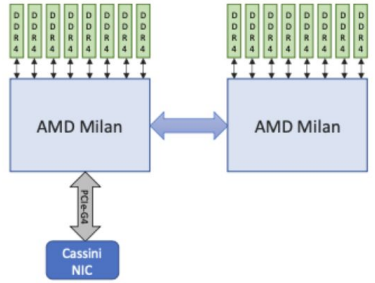
- Single AMD EPYC 7763 (Milan) CPU
- 64 cores per CPU
- Four NVIDIA A100 (Ampere) GPUs
- PCIe 4.0 GPU-CPU connection
- PCIe 4.0 NIC-CPU connection
- 4 HPE Slingshot 11 NICs
- 256 GB of DDR4 DRAM
- 40 GB of HBM per GPU with
- 1555.2 GB/s GPU memory bandwidth
- 204.8 GB/s CPU memory bandwidth
- 12 third 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

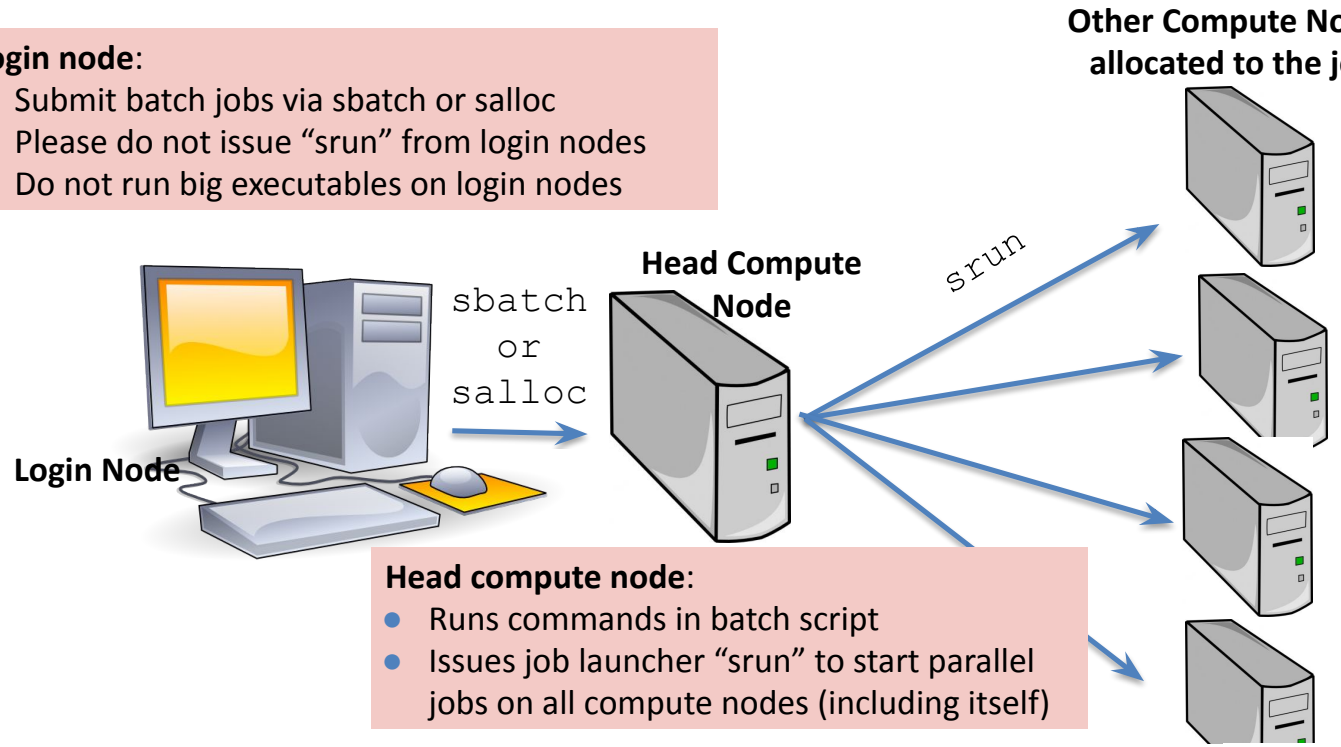
`module`	
`list`	To list the modules in your environment
`spider <name>`	To list available modules with <name> as substring, and how to load
`load/unload ..`	To load or unload module
`swap .. ..`	To swap modules
`show/display ..`	To see what a module loads, what env a module sets
`whatis ..`	Display 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

Argonne NATIONAL LABORATORY

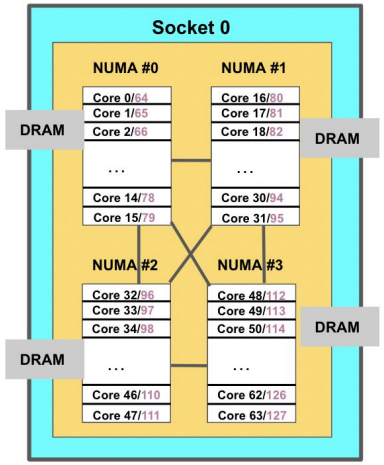
ECP EXASCALE COMPUTING PROJECT

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

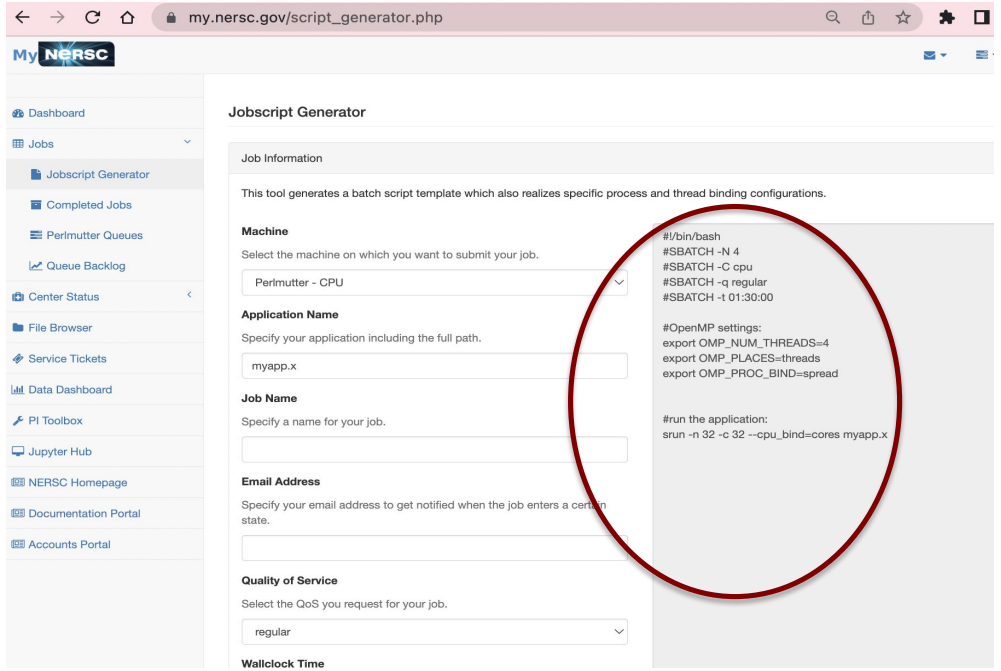
CPU on Perlmutter GPU



- 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 the reserved 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 --reservation=<reservation_name> -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



Argonne NATIONAL LABORATORY

ECP EXASCALE COMPUTING PROJECT

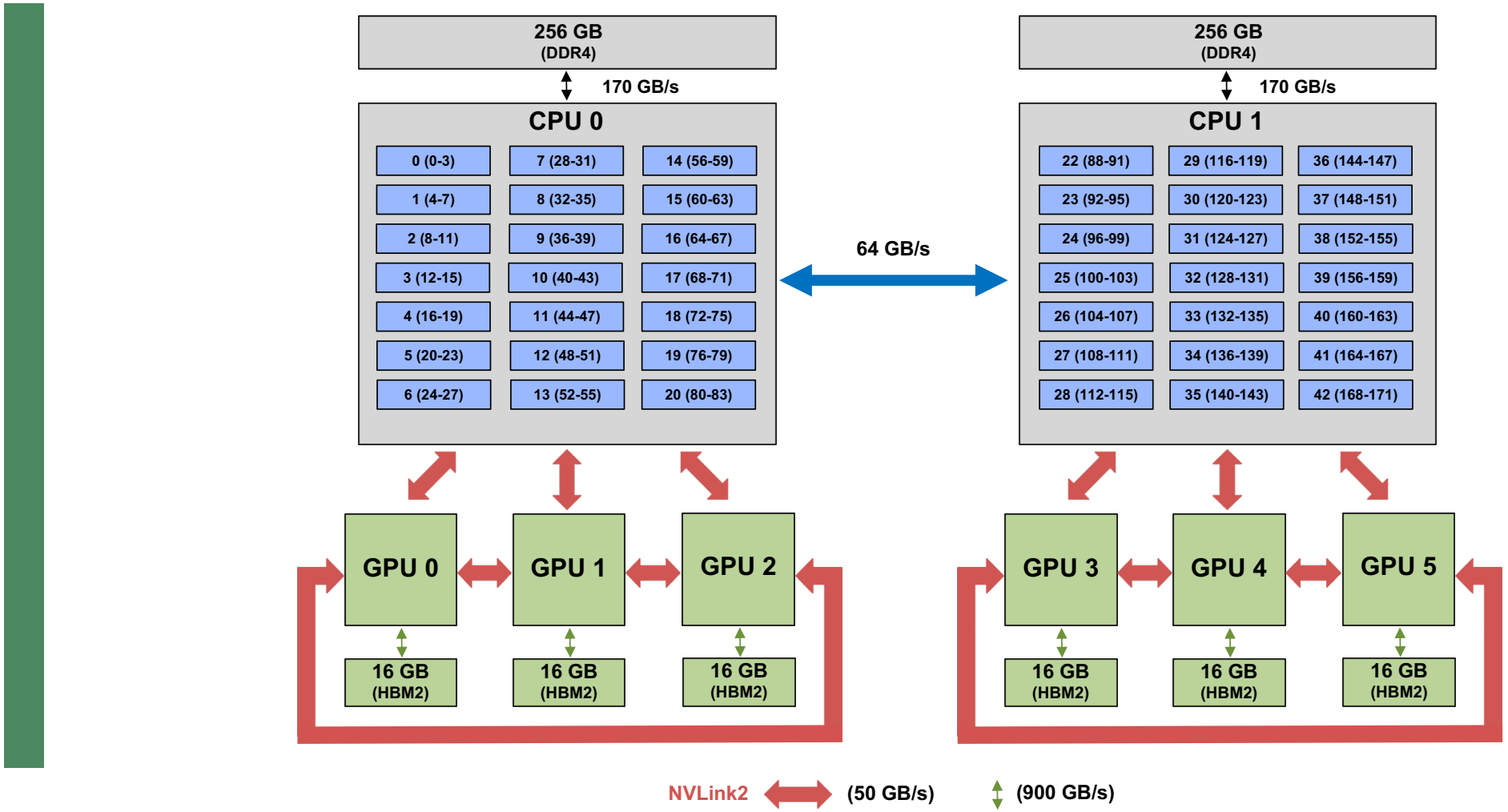
Perlmutter GPU Queue Policy (as of June 2023)

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

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

Ascent nodes - 2 IBM Power 9 + 6 NVIDIA V100



Argonne NATIONAL LABORATORY

ECP EXASCALE COMPUTING PROJECT

Ascent Available File systems

NFS Directories – This is where you might want to keep source code and build your application.

NOTE: These directories are read-only from the compute nodes!

`/ccsopen/home/<user_id>`

- Your personal home directory

`/ccsopen/proj/<project_id>`

- Can be accessed by all participants of this event
- You should create a directory here with your team name to collaborate (source code, scripts, etc.)

GPFS Directories (parallel file system) – This is where you should write data when running on Ascent's compute nodes.

`/gpfs/wolf/<project_id>/scratch/<user_id>`

- Your personal GPFS scratch directory

`/gpfs/wolf/<project_id>/proj-shared`

- Can be accessed by all participants of the event
- You should create a directory here with your team name to collaborate (data written from compute nodes)

Ascent Common LSF Commands

Function	LSF Command
Submit a batch script	bsub
Monitor Queue	jobstat / bjobs
Investigate Job	bhist
Alter Queued Job	bmod
Remove Queued Job	bkill
Hold Queued Job	bstop
Release Held Job	bresume

See manual pages for more info

Ascent Common bsub options

Option	Example Usage	Description
-W	#BSUB -w 1:00	Request walltime [hours:]minutes
-nnodes	#BSUB -nnodes 1	Number of nodes
-P	#BSUB -P <project_id>	Project to charge
-J	#BSUB -J MyJobName	Name of job
-o	#BSUB -o jobout.%J	File which STDOUT is directed (%J replaced by <job_id>)
-e	#BSUB -e joberr.%J	File which STDERR is directed (%J replaced by <job_id>)
-alloc_flags	#BSUB -alloc_flags "gpumps smt1"	Request CUDA MPS and set SMT level of CPU cores

See manual page for more info

Ascent jsrun – basic options

```
jsrun [-n #resource sets] [CPU cores, GPUs, tasks in each resource set] program [program args]
```

jsrun Flags		Description	Default Value
Long	Short		
--nrs	-n	Number of RS	All available physical cores
--tasks_per_rs	-a	Number of MPI tasks (ranks) per RS	N/A (total set instead [-p])
--cpu_per_rs	-c	Number of CPUs (physical cores) per RS	1
--gpu_per_rs	-g	Number of GPUs per RS	0
--bind	-b	Number of physical cores allocated per task	packed:1
--rs_per_host	-r	Number of RS per host (node)	N/A
--latency_priority	-l	Controls layout priorities	gpu-cpu,cpu-mem,cpu-cpu
--launch_distribution	-d	Order of tasks started on multiple RS	packed

See `man jsrun` for full list of options

Ascent jsrun Job Launcher – Tools & Documentation

Recent Presentation of "jsrun Basics"

- Slides: https://www.olcf.ornl.gov/wp-content/uploads/2019/12/jsrun_basics.pdf
- Recording: https://vimeo.com/393782415

`job-step-viewer`

- https://jobstepviewer.olcf.ornl.gov/

"Job Launcher (jsrun)" section of the Summit User Guide

- https://docs.olcf.ornl.gov/systems/summit_user_guide.html#job-launcher-jsrun

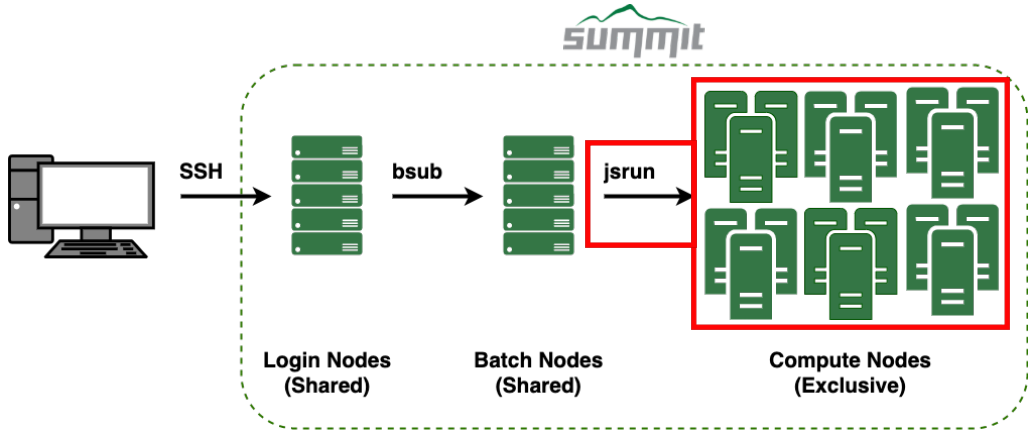
Ascent: must use jsrun to run on compute nodes

```
[t4p@login1: ~]$ hostname
login1

[t4p@login1: ~]$ bsub -P <project_id> -nnodes 1 -W 60 -Is /bin/bash
Job <15167> is submitted to default queue <batch>.
<<Waiting for dispatch ...>>
<<Starting on login1>>

[t4p@login1: ~]$ hostname
login1

[t4p@login1: ~]$ jsrun -n1 hostname
h49n16
```



The login nodes are shared among all participants (compiling, file editing, data analysis, etc.), so please **DO NOT RUN YOUR APPLICATIONS ON THE LOGIN NODES!!**

Ascent LSF Batch Script Example (non-interactive)

Batch script example

```
#!/bin/bash

#BSUB -W 30
#BSUB -nnodes 1
#BSUB -P <project_id>
#BSUB -o example.o%J
#BSUB -J example

jsrun -n2 -r2 -a1 -c1 hostname
```

- 30 minute walltime
- 1 node
- Project to charge
- Output file example.o.<job_id>
- Job name

Batch submission

```
[ascent-login1]$ bsub example.lsf
Job <29209> is submitted to default queue <batch>.
[ascent-login1]$
```

Ascent Other Useful Flags

`--smpiargs="-gpu"`

- `jsrun` flag that enables CUDA-Aware MPI
- If you are not familiar with CUDA-Aware MPI or GPUDirect, please see this tutorial: https://github.com/olcf-tutorials/MPI_ping_pong

`-alloc_flags "gpumps smt1"`

- `bsub` flag that allows you to start a CUDA MPS server or change the SMT mode of the physical CPU cores
- Multiple options are separated by a space-delimited list

Ascent Queue Policy

Number of Nodes	Max Walltime
1 – 2	2 hours
3 – 4	1 hour

There are a total of 15 schedulable compute nodes in Ascent, so please be respectful of others when requesting resources…

- Try to limit yourself to 1 compute node unless needed
- When you're finished with an allocation, please kill it (i.e., `exit` from within an interactive job or `bkill JOBID` for batch jobs).

Ascent: Other Helpful Links

OLCF Summit User Guide

- https://docs.olcf.ornl.gov/systems/summit_user_guide.html
  - NOTE: Ascent mounts different file systems than Summit, so please refer to info in these slides.
- NVIDIA's Nsight Profiling Tools
  - https://docs.olcf.ornl.gov/systems/summit_user_guide.html#profiling-gpu-code-with-nvidia-developer-tools

OLCF Training Archive

- Contains slides and recordings from previous OLCF training events.
- https://docs.olcf.ornl.gov/training/training_archive.html

ATPESC Resources

ALCF – Polaris, Theta, ThetaGPU and Cooley

- **Project name: ATPESC2023**
- **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 each machine (**command**: pbs_rstat on polaris, showres on theta, thetagpu, cooley)
- **Queue**
  - **Polaris (check *pbs_rstat*), Theta: ATPESC2023, ThetaGPU: training-gpu, or single-gpu, Cooley: training (check *showres*) or default for running without reservation**

ATPESC Resources

OLCF – Ascent

- **Project name: trn021**
- **Queue: running without reservation**
- *Ascent User Guide https://docs.olcf.ornl.gov/systems/ascent_user_guide.html*
- *Tools to learn how to use the `jsrun` job launcher*
  - *Hello_jsrun – A "Hello, World!"-type program to help understand resource layouts on Summit/Ascent nodes.*
  - *Jsrun Quick Start Guide – A very brief overview to help get you started*
  - *Job-step-viewer – A graphical tool to learn the basics of jsrun*
- *OLCF Tutorials at https://github.com/olcf-tutorials*

- See documents in your Argonne Folder for additional information
- *For other questions, email: help@olcf.ornl.gov*

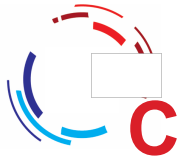
ATPESC Resources

NERSC – Perlmutter (HPE Cray XE)

- **Project name: ntrain5**
- **Reservations:** Please check the details of the reservations directly (**command**: scontrol show reservations)
- **Queue**
  - **atpesc_xxxxx (check *scontrol show reservations*) or default for running without reservation**
- **Reference**

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

ATPESC Resources



Cloud resources for Tools track

- Intel DevCloud
  - Test performance on Intel CPU, GPU, and FPGA architectures
  - Free access to Intel oneAPI toolkit and components and the latest Intel hardware
  - 120 days of access (extensions available)
  - Request free access at https://devcloud.intel.com/oneapi/

- AMD Accelerator Cloud (TBD)

Questions?

- *Use this presentation as a reference during ATPESC!*
- Supplemental info will be posted as well

Hands-on exercise

- On Theta
- On ThetaGPU
- On Cooley
- On Polaris
- On Perlmutter
- On Ascent

Hands-on exercise: Theta

```
$ ssh -Y {your_username}@theta.alcf.anl.gov          # Login to Theta
```

```
$ module list                                       # See loaded modules
```

```
jkwack@thetalogin6:~> module li
Currently Loaded Modulefiles:
  1) modules/3.2.11.4                                9) dmapp/7.1.1-7.0.2.1_3.31__g38cf134.ari          17) perftools-base/20.06.0
  2) intel/19.1.0.166                               10) gni-headers/5.0.12.0-7.0.2.1_3.33__g3b1768f.ari 18) PrgEnv-intel/6.0.7
  3) craype-network-aries                           11) xpmem/2.2.20-7.0.2.1_3.23__g87eb960.ari         19) craype-mic-knl
  4) craype/2.6.5                                   12) job/2.2.4-7.0.2.1_3.25__g36b56f4.ari            20) cray-mpich/7.7.14
  5) cray-libsci/20.06.1                            13) dvs/2.12_2.2.189-7.0.2.1_2.18__g456afdbd        21) nompirun/nompirun
  6) udreg/2.3.2-7.0.2.1_3.32__g8175d3d.ari         14) alps/6.6.59-7.0.2.1_4.32__g872a8d62.ari         22) adaptive-routing-a3
  7) ugni/6.0.14.0-7.0.2.1_4.34__ge78e5b0.ari       15) rca/2.2.20-7.0.2.1_3.32__g8e3fb5b.ari           23) darshan/3.3.0
  8) pmi/5.0.16                                     16) atp/3.6.4                                       24) xalt
```

```
$ module avail                                      # See available modules
```

```
$ showres                                           # Check reservation

$ qstat -u {your_username}                          # To see your jobs

$ qstat -fu {your_username}                         # To see your jobs with more verbose information
```

Hands-on exercise: Theta

```
$ cd /grand/ATPESC2023                                  # Go to the project folder in grand
$ cd usr                                                # Go to user space under project
$ mkdir {your_username}                                 # Create your space
$ cd {your_username}

$ cp -rf /grand/ATPESC2023/EXAMPLES/track-0-getting-started/GettingStarted .
$ cd GettingStarted/theta/

$ more hellompi.c                           # See the example source
$ more Makefile                             # An example of how to compile a code
    …
    CC=cc
    hellompi: hellompi.c
          which $(CC)
          $(CC) -g -O0 -o hellompi hellompi.c
    …
```

Hands-on exercise: Theta

```
$ more submit.sh
#!/bin/bash
#COBALT -n 1
#COBALT -t 30
#COBALT -A ATPESC2023
#COBALT -q ATPESC2023
#COBALT --attrs mcdram=cache:numa=quad
#COBALT --attrs filesystems=home,grand,eagle

echo "COBALT_JOBID = " $COBALT_JOBID
echo "COBALT_JOBSIZE (nodes) =" $COBALT_JOBSIZE
echo "COBALT_PARTNAME = " $COBALT_PARTNAME

rpn=8
depth=1

# option  long version            (explanation)
#
# -n                              "PEs" (ranks)
# -N      --pes-per-node          ranks per node
# -d      --cpus-per-pe           hyperthreads per rank
# -cc     --cpu-binding depth
# -j                              cpus (hyperthreads) per compute unit (core)

aprun -n $((COBALT_JOBSIZE*rpn)) -N $rpn -d $depth -j 1 -cc depth ./hellompi
status=$?

echo "Exit status of aprun is: $status"
exit $status
```

Hands-on exercise: Theta

```
$ export CRAYPE_LINK_TYPE=dynamic                                          # For dynamic linking
$ module swap PrgEnv-intel PrgEnv-cray; module swap PrgEnv-cray PrgEnv-intel    # To avoid errors while loading libraries

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

$ aprun -n 4 ./hellompi                                          # It won't work since you are on a login node
    XALT Error: unable to find aprun
```

Hands-on exercise: Theta

```
$ qsub -I -n 1 -t 30 --attrs filesystems=home,grand,eagle -A ATPESC2023 -q ATPESC2023     # Start an interactive job mode
```

```
Connecting to thetamom2 for interactive qsub...
Currently Loaded Modulefiles:
  1) modules/3.2.11.4                                 11) nodehealth/5.6.27-7.0.2.1_5.31__g20e015c.ari       21) perftools-base/20.06.0
  2) alps/6.6.59-7.0.2.1_4.32__g872a8d62.ari          12) system-config/3.6.3072-7.0.2.1_9.16__gca557bd7.ari 22) PrgEnv-intel/6.0.7
  3) nodestat/2.3.89-7.0.2.1_3.18__g8645157.ari       13) Base-opts/2.4.142-7.0.2.1_3.15__g8f27585.ari       23) craype-mic-knl
  4) sdb/3.3.812-7.0.2.1_3.32__gd6c4e58.ari           14) intel/19.1.0.166                                   24) cray-mpich/7.7.14
  5) udreg/2.3.2-7.0.2.1_3.32__g8175d3d.ari           15) craype-network-aries                               25) nompirun/nompirun
  6) ugni/6.0.14.0-7.0.2.1_4.34__ge78e5b0.ari         16) craype/2.6.5                                       26) adaptive-routing-a3
  7) gni-headers/5.0.12.0-7.0.2.1_3.33__g3b1768f.ari  17) cray-libsci/20.06.1                                 27) darshan/3.3.0
  8) dmapp/7.1.1-7.0.2.1_3.31__g38cf134.ari           18) pmi/5.0.16                                         28) xalt
  9) xpmem/2.2.20-7.0.2.1_3.23__g87eb960.ari          19) atp/3.6.4
 10) llm/21.4.631-7.0.2.1_4.15__g4fcec58.ari          20) rca/2.2.20-7.0.2.1_3.32__g8e3fb5b.ari
```

```
$ cd /grand/ATPESC2023/usr

$ cd {your_username}/GettingStarted/theta/

$ export CRAYPE_LINK_TYPE=dynamic                                          # For dynamic linking

$ module swap PrgEnv-intel PrgEnv-cray; module swap PrgEnv-cray PrgEnv-intel    # To avoid errors while loading libraries

$ aprun -n 4 ./hellompi
```

```
jkwack@thetamom2:/grand/ATPESC2023/usr/jkwack/GettingStarted/theta> aprun -n 4 ./hellompi
0: Hello!
1: Hello!
2: Hello!
3: Hello!
Application 28486031 resources: utime ~0s, stime ~2s, Rss ~10160, inblocks ~4276, outblocks ~8
```

Hands-on exercise: ThetaGPU

```
$ ssh thetagpusn1                    # Login to ThetaGPU from Theta,  (or, $ ssh thetagpusn2)

$ module list                        # See loaded modules
$ module avail                       # See available modules

$ showres                            # Check reservation (only for thetaGPU, not on theta)
$ qstat -u {your_username}           # To see your jobs (only jobs on thetaGPU, not on theta)
$ qstat -fu {your_username}          # To see your jobs with more verbose information
```

Hands-on exercise: ThetaGPU

```
$ vi ~/.bashrc
```

```
$ cat ~/.bashrc
    # .bashrc
    # Source global definitions
    if [ -f /etc/bashrc ]
    then
        . /etc/bashrc
    elif [ -f /etc/bash.bashrc ]
    then
        . /etc/bash.bashrc
    fi
```

```
$ vi ~/.bash_profile
```

```
$ cat ~/.bash_profile
    # .bash_profile
    # Get the aliases and functions
    if [ -f ~/.bashrc ]; then
            . ~/.bashrc
    fi
    # proxy settings
    export HTTP_PROXY=http://theta-proxy.tmi.alcf.anl.gov:3128
    export HTTPS_PROXY=http://theta-proxy.tmi.alcf.anl.gov:3128
    export http_proxy=http://theta-proxy.tmi.alcf.anl.gov:3128
    export https_proxy=http://theta-proxy.tmi.alcf.anl.gov:3128
```

Hands-on exercise: ThetaGPU

```
$ source ~/.bashrc
$ cd /grand/ATPESC2023/usr/{your_username}/GettingStarted/thetaGPU/          # Go to the example folder

$ more Makefile                                                             # An example of how to compile a code
    …
    CC=mpicc
    hellompi: hellompi.c
          which $(CC)
          $(CC) -g -O0 -o hellompi hellompi.c
    …
```

Hands-on exercise: ThetaGPU

```
$ more submit.sh                                    # An example of job script

    #!/bin/bash -l
    #COBALT -n 1
    #COBALT -t 30
    #COBALT -A ATPESC2023
    #COBALT -q single-gpu
    #COBALT --attrs filesystems=home,grand,eagle

    mpirun -n 16 ./hellompi
    status=$?

    echo "mpirun status is $status"
    exit $status
```

Hands-on exercise: ThetaGPU

```
$ mpicc -o hellompi hellompi.c                    # Build the example

$ make clean; make                                # Another way to build the example


$ nvidia-smi                                      # NVIDIA A100 GPUs are visible since you are on a login node
```

```
jkwack@thetagpusn1:/grand/ATPESC2023/usr/jkwack/GettingStarted/thetaGPU$ nvidia-smi

Command 'nvidia-smi' not found, but can be installed with:

apt install nvidia-340       (You will have to enable component called 'restricted')
apt install nvidia-utils-390 (You will have to enable component called 'restricted')

Ask your administrator to install one of them.
```

Hands-on exercise: ThetaGPU

```
$ qsub -I -n 1 -t 30 --attrs filesystems=home,grand,eagle -A ATPESC2023 -q single-gpu     # Start an interactive job mode
```

```
jkwack@thetagpusn1:~$ qsub -I -n 1 -t 30 --attrs filesystems=home,grand,eagle -A ATPESC2023 -q single-gpu
Job routed to queue "single-gpu".
Wait for job 10144800 to start...
Opening interactive session to thetagpu06-gpu0
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-144-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu 27 Jul 2023 02:31:38 PM CDT

  System load:  1.15            Users logged in:              0
  Usage of /:   1.5% of 1.72TB  IPv4 address for docker0:     172.17.0.1
  Memory usage: 3%              IPv4 address for enp226s0:    10.230.2.194
  Swap usage:   0%              IPv4 address for infinibond0: 172.23.2.194
  Processes:    3260            IPv4 address for infinibond0: 172.22.2.194

 * Introducing Expanded Security Maintenance for Applications.
   Receive updates to over 25,000 software packages with your
   Ubuntu Pro subscription. Free for personal use.

     https://ubuntu.com/pro

Last login: Thu Jul 27 12:08:48 2023 from thetagpusn1.mcp

Currently Loaded Modules:
  1) openmpi/openmpi-4.0.5   2) Core/StdEnv
```

Hands-on exercise: ThetaGPU

```
$ cd /grand/ATPESC2023/usr/{your_username}/GettingStarted/thetaGPU/

$ mpirun -n 4 ./hellompi
```

```
jkwack@thetagpu06:/grand/ATPESC2023/usr/jkwack/GettingStarted/thetaGPU$ mpirun -n 4 ./hellompi
0: Hello!
1: Hello!
2: Hello!
3: Hello!
```

```
$ nvidia-smi
```

```
jkwack@thetagpu06:/grand/ATPESC2023/usr/jkwack/GettingStarted/thetaGPU$ nvidia-smi
Thu Jul 27 14:34:28 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.161.03   Driver Version: 470.161.03   CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA A100-SXM...  On   | 00000000:07:00.0 Off |                    0 |
| N/A   22C    P0    53W / 400W |      0MiB / 40536MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Hands-on exercise: Cooley

```
$ ssh -Y {your_username}@cooley.alcf.anl.gov                    # Login to Cooley


$ softenv                                                       # Check available environment
$ vi .soft.cooley                                               # Update your environment
$ cat .soft.cooley
      +openmpi-2.1.5-intel
      +intel-composer-xe
      @default
$ resoft                                                        # Apply the updated environment
$ which mpicc
      /soft/libraries/mpi/openmpi-2.1.5/intel/bin/mpicc
```

Hands-on exercise: Cooley

```
$ showres                                                    # Check reservation
$ qstat -u {your_username}                                   # To see your jobs
$ qstat -fu {your_username}                                  # To see your jobs with more verbose information

$ qsub -I -n 1 -t 30 --attrs filesystems=home,grand,eagle -A ATPESC2023 -q training      # Start an interactive job mode

$ cd /grand/ATPESC2023/usr/{your_username}/GettingStarted/cooley/              # Go to the example folder
$ more Makefile                                              # An example of how to compile a code
    …
    CC=mpicc

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

Hands-on exercise: Cooley

```
$ more submit.sh                                    # An example of job script

    #!/bin/bash
    #COBALT -n 1
    #COBALT -t 30
    #COBALT -A ATPESC2023
    #COBALT -q training
    #COBALT --attrs filesystems=home,grand,eagle

    NODES=`cat $COBALT_NODEFILE | wc -l`
    PROCS=$((NODES * 12))

    mpirun -n $PROCS ./hellompi
    status=$?

    echo "mpirun status is $status"
    exit $status
```

Hands-on exercise: Cooley

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

$ mpirun -n 4 ./hellompi
```

```
[jkwack@cc006 ~]$ cd /grand/ATPESC2023/usr/jkwack/GettingStarted/cooley/
[jkwack@cc006 cooley]$ make clean; make
/bin/rm -f *.error *.output *.cobaltlog hellompi
which mpicc
/soft/libraries/mpi/openmpi-2.1.5/intel/bin/mpicc
mpicc -g -O0 -o hellompi hellompi.c
[jkwack@cc006 cooley]$ mpirun -n 4 ./hellompi
0: Hello!
1: Hello!
2: Hello!
3: Hello!
```

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-01:~> module li

Currently Loaded Modules:
  1) craype-x86-rome          3) craype-network-ofi       5) nvhpc/21.9      7) cray-dsmml/0.2.2    9) cray-pmi/6.1.2       11) cray-pals/1.1.7     13) PrgEnv-nvhpc/8.3.3
  2) libfabric/1.11.0.4.125   4) perftools-base/22.05.0   6) craype/2.7.15   8) cray-mpich/8.1.16  10) cray-pmi-lib/6.0.17  12) cray-libpals/1.1.7  14) craype-accel-nvidia80
```

```
$ qstat -u {your_username}                                       # To see your jobs

$ pbs_rstat                                                      # Check reservation
```

```
jkwack@polaris-login-01:~> pbs_rstat
Resv ID    Queue    User     State            Start / Duration / End
--------------------------------------------------------------------
M561743.po M561743  appmm2pb CO           Fri 13:00 / 565200 / Fri Aug 04 02:00
R563886.po R563886  richp@po CO     Fri Aug 04 02:00 / 21600 / Fri Aug 04 08:00
R563888.po R563888  richp@po CO     Sat Aug 05 02:00 / 21600 / Sat Aug 05 08:00
R563890.po R563890  richp@po CO     Tue Aug 08 02:00 / 21600 / Tue Aug 08 08:00
R563891.po R563891  richp@po CO     Wed Aug 09 02:00 / 21600 / Wed Aug 09 08:00
R563892.po R563892  richp@po CO     Thu Aug 10 02:30 / 19800 / Thu Aug 10 08:00
R563894.po R563894  richp@po CO     Fri Aug 11 02:00 / 21600 / Fri Aug 11 08:00
R563895.po R563895  richp@po CO     Sat Aug 12 02:00 / 21600 / Sat Aug 12 08:00
R563919.po R563919  richp@po CO     Mon Aug 07 22:30 / 12600 / Tue Aug 08 02:00
R563922.po R563922  richp@po CO     Wed Aug 09 12:30 / 50400 / Thu Aug 10 02:30
R563923.po R563923  richp@po CO     Thu Aug 10 14:00 / 43200 / Fri Aug 11 02:00
R563925.po R563925  richp@po CO     Fri Aug 11 18:45 / 8100 / Fri Aug 11 21:00
R564206.po R564206  hzheng@* CO     Sat Aug 05 14:00 / 43200 / Sun Aug 06 02:00
```

Hands-on exercise: Polaris

```
$ qsub -I -l select=1 -l walltime=00:30:00 -l filesystems=home:grand:eagle  -A ATPESC2023 -q debug
```

```
jkwack@polaris-login-01:~> qsub -I -l select=1 -l walltime=00:30:00 -l filesystems=home:grand:eagle  -A ATPESC2023 -q debug
qsub: waiting for job 564215.polaris-pbs-01.hsn.cm.polaris.alcf.anl.gov to start
qsub: job 564215.polaris-pbs-01.hsn.cm.polaris.alcf.anl.gov ready


Currently Loaded Modules:
  1) craype-x86-rome          3) craype-network-ofi       5) nvhpc/21.9      7) cray-dsmml/0.2.2    9) cray-pmi/6.1.2       11) cray-pals/1.1.7     13) PrgEnv-nvhpc/8.3.3
  2) libfabric/1.11.0.4.125   4) perftools-base/22.05.0   6) craype/2.7.15   8) cray-mpich/8.1.16  10) cray-pmi-lib/6.0.17  12) cray-libpals/1.1.7  14) craype-accel-nvidia80
```

```
$ cd /grand/projects/ATPESC2023/usr/{your_username}/GettingStarted/polaris/

$ more makefile
    …
    CC=cc


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

Hands-on exercise: Polaris

```
$ more submit.sh

    #!/bin/bash
    #PBS -l select=1
    #PBS -l walltime=00:30:00
    #PBS -q debug
    #PBS -l filesystems=home:grand:eagle
    #PBS -A ATPESC2023
    #PBS -q debug

    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@x3005c0s7b1n0:/grand/ATPESC2023/usr/jkwack/GettingStarted/polaris> mpirun -n 4 --ppn 4 ./hellompi
2: Hello!
0: Hello!
1: Hello!
3: Hello!
```

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

Hands-on exercise: Perlmutter

```
$ ssh -Y {your_username}@perlmutter.nersc.gov                    # Login to Perlmutter

$ module avail                                                   # See available modules

$ module list                                                    # See loaded modules
```

```
train53@perlmutter:login10:~> module list

Currently Loaded Modules:
  1) craype-x86-milan                    5) PrgEnv-gnu/8.3.3       9) craype/2.7.20          13) xalt/2.10.2             17) craype-accel-nvidia80
  2) libfabric/1.15.2.0                  6) cray-dsmml/0.2.2      10) gcc/11.2.0             14) Nsight-Compute/2022.1.1  18) gpu/1.0
  3) craype-network-ofi                  7) cray-libsci/23.02.1.1 11) perftools-base/23.03.0 15) Nsight-Systems/2022.2.1
  4) xpmem/2.5.2-2.4_3.49__gd0f7936.shasta 8) cray-mpich/8.1.25   12) cpe/23.03              16) cudatoolkit/11.7
```

```
$ sqs                                                    # To see your jobs

$ scontrol show reservations                             # Check reservation

train53@perlmutter:login10:~> scontrol show reservations

…

ReservationName=atpesc_jul30 StartTime=2023-07-30T13:00:00 EndTime=2023-07-30T17:00:00 Duration=04:00:00

   Nodes=nid[001004-001005,001008-001009,001012-001013,001016-001017,001020-001021,001024-001025,001028-001029,001032-001033,001036-001037,001040-
001041] NodeCnt=20 CoreCnt=1280 Features=(null) PartitionName=gpu_ss11 Flags=

   TRES=cpu=2560

   Users=(null) Groups=(null) Accounts=ntrain5_g Licenses=(null) State=INACTIVE BurstBuffer=(null) Watts=n/a

   MaxStartDelay=(null)

…
```

Hands-on exercise: Perlmutter

```
$ salloc -q shared -C gpu -A ntrain5 --reservation=<reservation_name> -c 32 -G 1 -N 1 -t 30:00
```

```
train53@perlmutter:login16:~> salloc -q shared -C gpu -A ntrain5 -c 32 -G 1 -N 1 -t 30:00
salloc: Pending job allocation 12636790
salloc: job 12636790 queued and waiting for resources
salloc: job 12636790 has been allocated resources
salloc: Granted job allocation 12636790
salloc: Waiting for resource configuration
salloc: Nodes nid001017 are ready for job
```

```
$ cp -r /global/homes/t/train53/GettingStarted .

$ cd GettingStarted/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 --reservation=<reservation_name>
    #SBATCH -c 32
    #SBATCH -G 1
    #SBATCH -N 1
    #SBATCH -t 60:00

    srun -n 8 -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 8 ./hellompi
```

```
train53@nid001017:~/ATPESC_perlmutter> srun -n 4 -c 8 ./hellompi
0: Hello!
1: Hello!
3: Hello!
2: Hello!
```

```
$ nvidia-smi
```

- More references for Perlmutter
  - https://docs.nersc.gov/systems/perlmutter/

Hands-on exercise: Ascent

```
$ ssh -Y {your_username}@login1.ascent.olcf.ornl.gov          # Login to Ascent
$ module avail                                                # See available modules
$ module list                                                 # See loaded modules
```

```
[jkwack@login1.ascent ascent]$   module li

Currently Loaded Modules:
  1) xl/16.1.1-10   2) spectrum-mpi/10.4.0.3-20210112   3) lsf-tools/2.0   4) DefApps
```

```
$ bjobs -u {your_username}                                    # To see your jobs
```

Hands-on exercise: Ascent

```
$ bsub -P trn021 -nnodes 1 -W 120  -Is /bin/bash
```

```
[jkwack@login1.ascent ~]$ bsub -P trn021 -nnodes 1 -W 120  -Is /bin/bash
Job <233482> is submitted to default queue <batch>.
<<Waiting for dispatch ...>>
<<Starting on login1>>
bash-4.4$
```

```
$ cp -r /ccsopen/proj/trn021/GettingStarted .

$ cd GettingStarted/ascent

$ more makefile
    …
    CC=cc

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

Hands-on exercise: Ascent

```
$ more submit.sh

    #!/bin/bash
    #BSUB -nnodes 1
    #BSUB -W 0:30
    #BSUB -P trn021

    jsrun -n 4 ./hellompi
```

```
$ mpicc -o hellompi hellompi.c          # Build the example
$ make clean; make                      # Another way to build the example
$ jsrun -n 4 ./hellompi
```

```
bash-4.4$ jsrun -n 4 ./hellompi
3: Hello!
0: Hello!
1: Hello!
2: Hello!
```

- More references for Ascent
  - https://docs.olcf.ornl.gov/systems/ascent_user_guide.html

Thank you!

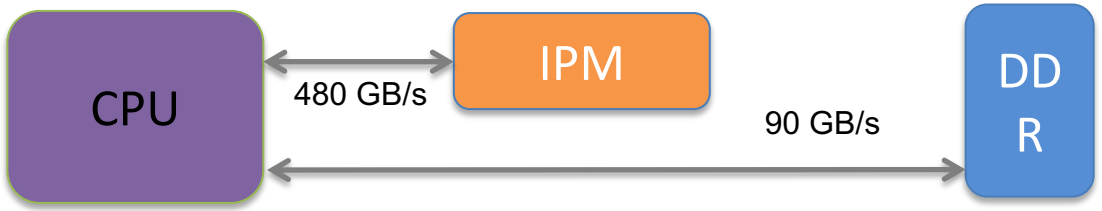
Supplemental Info

Theta Memory Modes - IPM and DDR

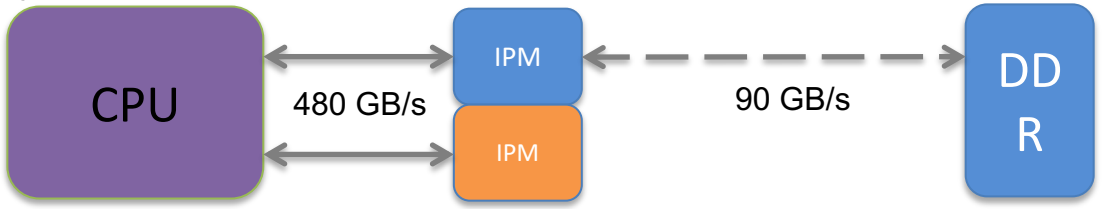
Selected at node boot time

Cache

Flat



Hybrid



- **Two memory types**
- In Package Memory (IPM)
  - 16 GB MCDRAM
  - ~480 GB/s bandwidth
- Off Package Memory (DDR)
  - Up to 384 GB
  - ~90 GB/s bandwidth
- **One address space**
- Possibly multiple NUMA domains
- **Memory configurations**
- Cached: DDR fully cached by IPM
  - Flat: user managed
- Hybrid: ¼, ½ IPM used as cache
  - **Managing memory:**
- jemalloc & memkind libraries
- Pragmas for static memory allocations

Theta queues and modes

- MCDRAM and NUMA modes can only be set by the system when nodes are rebooted. *Users cannot directly reboot nodes.*
- Submit job with the --attrs flag to get the mode you need. E.g.
  - qsub –n 32 –t 60 –attrs mcdram=cache:numa=quad ./jobscript.sh
- Other mode choices
  - mcdram: cache, flat, split, equal
  - numa: quad, a2a, hemi, snc2, snc4
- Queues
  - Normal jobs use queue named "default"
  - Debugging: debug-cache-quad, debug-flat-quad
    - Note: pre-set for mcdram/numa configuration
  - "qstat –Q" lists all queues