

Getting Started on ALCF Systems

Christopher Knight
Catalyst Team

Outline

<https://www.alcf.anl.gov/user-guides>

- Mira (Blue Gene Q) Quick Reference Guide
- Theta (KNL)
 - System Overview
 - Software & Environment Modules
 - Building your code
 - Queuing and running jobs with qsub & aprun
- Cooley (x86)
 - System Overview
 - Compiling and queuing jobs
- Tips for troubleshooting



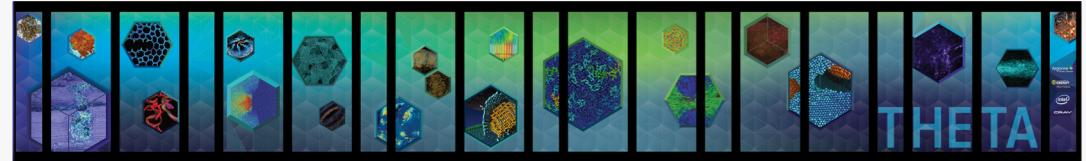
Mira - Quick Reference Guide

<https://www.alcf.anl.gov/mira>

- Compiling codes
 - <https://www.alcf.anl.gov/user-guides/compiling-and-linking-bgq>
 - <https://www.alcf.anl.gov/user-guides/bgq-compiling-and-linking-faqs>
- Threading: <https://www.alcf.anl.gov/user-guides/threading-bgq>
- Job submission: <https://www.alcf.anl.gov/user-guides/bgq-cobalt-job-control>
- Queues: <https://www.alcf.anl.gov/user-guides/job-scheduling-policy-bgq-systems>
- Debugging & Profiling: <https://www.alcf.anl.gov/user-guides/debugging-profiling>
- Checking status of job: <https://status.alcf.anl/mira/activity>
- Additional Reference Material
 - <https://www.alcf.anl.gov/presentations>
 - https://www.alcf.anl.gov/files/Knight_Software_JobSubmission2018.pdf
 - https://www.alcf.anl.gov/files/GettingStarted_Mira_2017.pdf

Theta

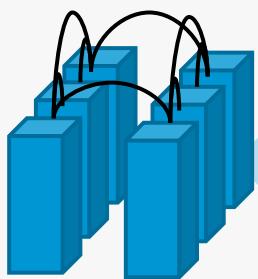
<https://www.alcf.anl.gov/theta>



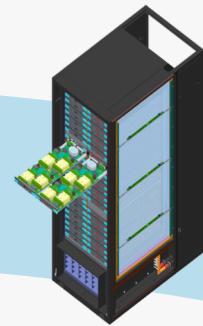
- Serves as a bridge between Mira and Aurora; simulation, data & learning system
- Cray XC40 system running the Cray software stack
- 11.69 PF peak performance
- 4392 nodes with 2nd Generation Intel® Xeon Phi™ processor
 - codenamed Knights Landing (KNL), 7230 SKU 64 cores, 1.3 GHz
 - 4 hardware threads/core
- 192 GB DDR4 & 16 GB MCDRAM memory on each node
- 128 GB SSD on each node
- Cray Aries high-speed interconnect in dragonfly topology
- Project filesystem is 10 PB Lustre with 210 GB/s throughput

Theta - System Overview

<https://www.alcf.anl.gov/theta>



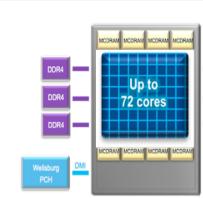
System: 24 Cabinets
4392 Nodes, 1152 Switches
Dual-plane, 12 groups, Dragonfly 12.1 TB/s Bi-Sec
11.7 PF Peak
70 TB MCDRAM, 843 TB DRAM



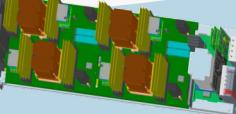
Cabinet: 3 Chassis, 75kW liquid/air cooled
510.72 TF 3TB MCDRAM, 36TB DRAM



Chassis: 16 Blades, 16 Cards
64 Nodes, 16 Switches
170.24 TF 1TB MCDRAM, 12TB DRAM



Node: KNL Socket
192 GB DDR4 (6 channels) **2.66 TF** 16GB MCDRAM
128 GB SSD



Compute Blade:
4 Nodes/Blade + Aries switch
10.64 TF 64GB MCDRAM
768GB DRAM

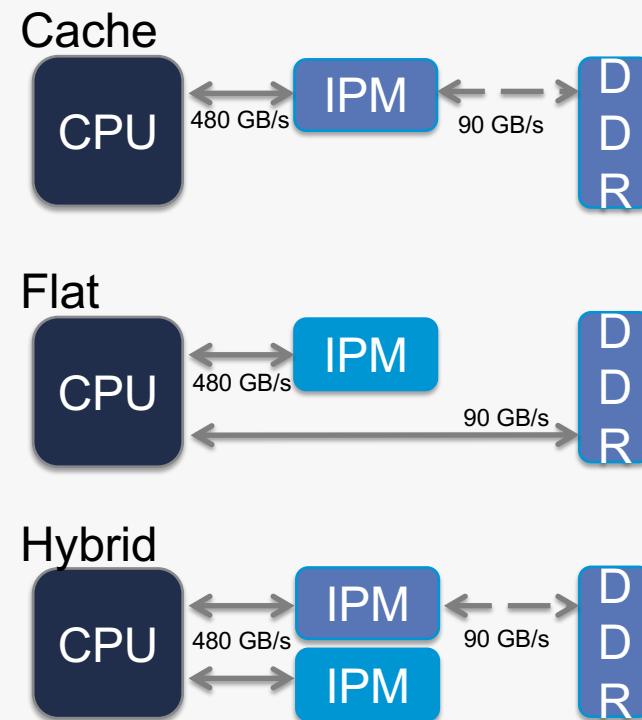


Sonexion Storage
4 Cabinets
Lustre file system
10 PB usable
210 GB/s

Theta - Memory Modes

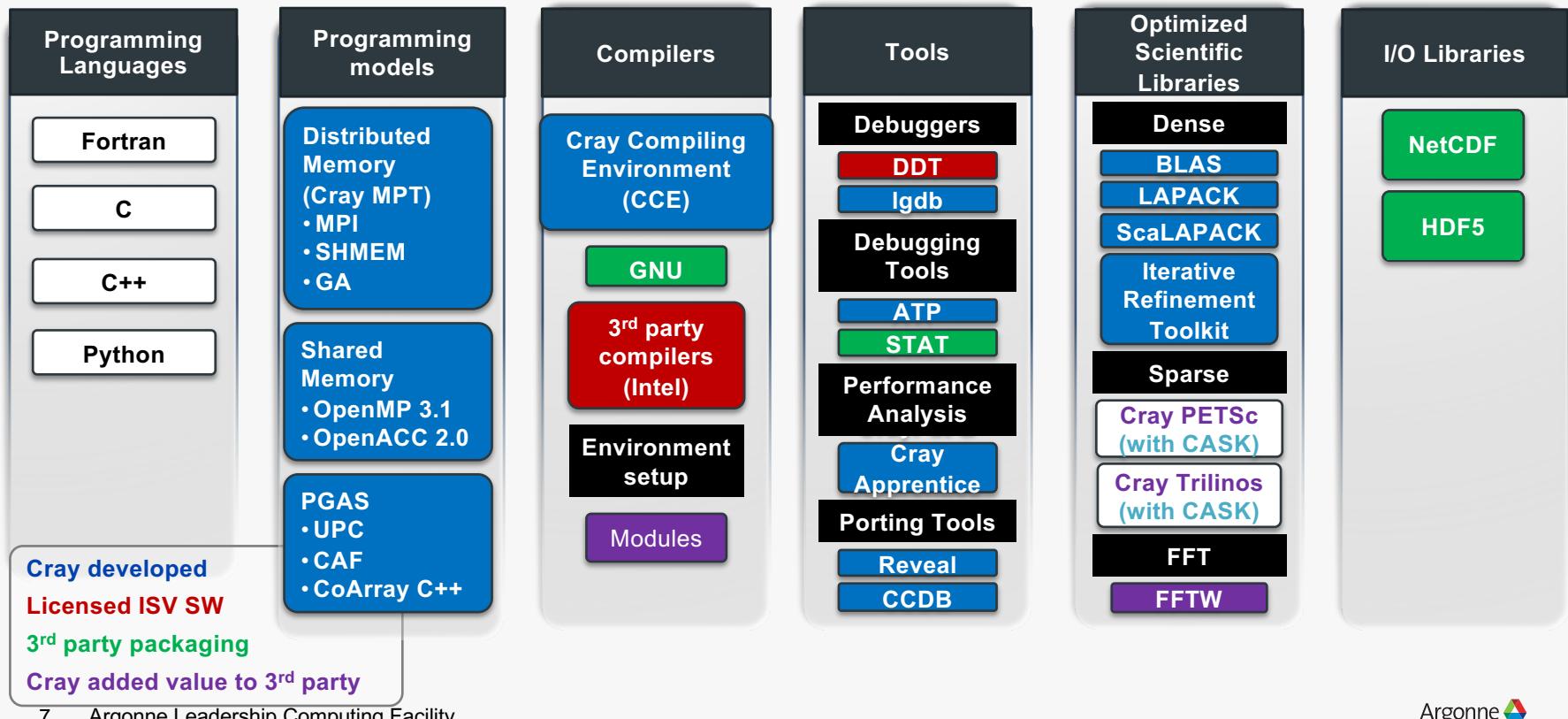
<https://www.alcf.anl.gov/user-guides/xc40-memory-modes>

- Two memory types
 - In Package Memory (IPM)
 - 16 GB MCDRAM @ ~480 GB/s
 - Off Package Memory (DDR)
 - 192 GB @ ~90GB/s
- Single address space; multiple NUMA domains
- Memory configurations
 - Cached: DDR fully cached by IPM
 - Flat: User managed
 - Hybrid: $\frac{1}{4}$, $\frac{1}{2}$, IPM used as cache
- Managing memory
 - jemalloc & memkind libraries
 - Pragmas for static memory allocations



Theta - Cray Programming Environment

<https://www.alcf.anl.gov/user-guides/software-and-libraries>



Theta - Non-system Software & Libraries

<https://www.alcf.anl.gov/user-guides/software-and-libraries>

- Compilers: </soft/compilers>
 - llvm and intel beta releases
- Debuggers: </soft/debuggers>
 - DDT
- Libraries: </soft/libraries>
 - argobots, bolt, breakpad
- Performance tools: </soft/perf-tools>
 - Darshan, HPCToolkit, memlog, TAU
- Visualization: </soft/visualization>
 - Paraview, vtk, visit, ffmpeg
- Machine/Deep Learning & workflow
 - Intel Optimized Tensorflow, Keras, Neon, MXNet, Caffe2, Theano, CNTK, PyTorch, Sci-kit Learn, Graph Analytics (Cray Graph Engine), Horovod
- Optimized with performance libraries Intel MKL, MKL-DNN, LibXSMM, etc...
- Workflow/Data analysis: Singularity containers, Jupyter Hub, MongoDB, Apache Spark, R, Balsam
- Python: Intel, Cray, Anaconda

Theta - Modules

<https://modules.sourceforge.net>

- 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
 - List available module commands: `module help`
 - List currently loaded modules: `module list`
 - List all available modules: `module avail`
 - Add module to environment: `module load <mod>`
 - Remove module from environment: `module unload <mod>`
 - Swap loaded module with new one: `module switch <mod_old> <mod_new>`
 - List information about module: `module show <mod>`

Theta - Compiler Wrappers

<https://www.alcf.anl.gov/user-guides/compiling-and-linking-vc40>

- For all compilers (Intel, Cray, GNU, Clang)
 - Use cc, CC, ftn
 - Do not use mpicc, mpiCC, mpic++, mpif77, mpif90, etc... as they do not generate code for compute nodes
- Select compiler you want: `module swap <PrgEnv-old> <PrgEnv-new>`
 - Intel (default): `PrgEnv-intel`
 - Cray: `module swap PrgEnv-intel PrgEnv-cray`
 - GNU: `module swap PrgEnv-intel PrgEnv-gnu`
 - Clang: `module swap PrgEnv-intel PrgEnv-llvm`
- Cray wrappers
 - List complete command executed: `--craype-verbose`
 - Can disable automatic linking with libsci: `module unload cray-libsci`

Theta - Preparing to Submit Job

<https://www.alcf.anl.gov/user-guides/allocation-accounting-sbank>

- Check that you are a member of a project: `projects`
- Check available disk space
 - \$HOME directory: `myquota`
 - Project directories: `myprojectquotas`
 - Project directories should be used for production work
- Check that your project has core-hours available
 - Use `sbank` command to query allocation details
 - Allocation available to project: `sbank l a -p <project_name>`
 - Charges against project by user: `sbank l u -p <project_name> -u <user>`
 - Charges on Theta are based on number of nodes
 - Jobs smaller than 128 nodes are allocated 128 nodes

Theta - Cobalt

<https://www.alcf.anl.gov/user-guides/cobalt-job-control-xc40>

- Resource management software on all ALCF systems
 - Syntax is similar to a PBS-style script
- Job management commands
 - Submit a job: `qsub`
 - Query job status: `qstat`
 - Delete a job: `qdel`
 - Alter job parameters: `qalter`
 - Move job to different queue: `qmove`
 - Place queued job (non-running) on hold: `qhold`
 - Release hold on job: `qrsls`

Theta - qsub

<https://www.alcf.anl.gov/user-guides/running-jobs-xc40>

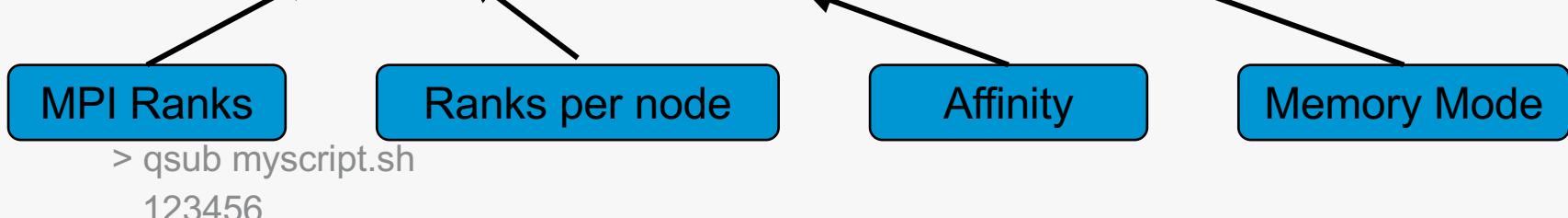
- Standard options
 - Project to charge: `-A <project_name>`
 - Queue: `-q <queue>`
 - Maximum walltime: `-t <time_in_minutes>`
 - Number of nodes: `-n <number_of_nodes>`
 - Prefix for output files: `-O <file_prefix>`
 - E-mail notifications: `-M <email_address>`
 - Dependencies: `--dependencies <jobid1>:<jobid2>`
 - Interactive job: `-I` or `--interactive`

Theta - Submitting Script Jobs

<https://www.alcf.anl.gov/user-guides/running-jobs-vc40>

- Executable is invoked within script (bash, csh, ...)
- `aprun` is used to launch executables on compute nodes

```
> cat myscript.sh  
#!/bin/sh  
#COBALT -A <project_name> -t 10 -n 16 -O <prefix_name> -q default  
#COBALT --attrs mcdram=cache:numa=quad  
echo "Starting Cobalt job script"  
aprun -n 1024 -N 64 -d 1 -j 1 --cc depth <app> <app_args>
```



Theta - aprun Overview

<https://www.alcf.anl.gov/user-guides/running-jobs-xc40>

- `aprun` options
 - Total number of MPI ranks: `-n <total_number_ranks>`
 - Number of MPI ranks per node: `-N <number_ranks_per_node>`
 - Number of hyperthreads per MPI rank (depth): `-d <num_hardware_threads_per_rank>`
 - Number of hyperthreads per core: `-j <number_hardware_threads_per_core>`
 - MPI rank and thread placement: `--cc depth`
 - Environment variables: `-e <VAR1=1> -e <VAR2=1>`
 - Core specialization: `-r <number_hardware_threads>`
- Environment settings you may need
 - `-e OMP_NUM_THREADS=<num_threads>`
 - `--cc none -e KMP_AFFINITY=<affinity>`
- See also man `aprun`

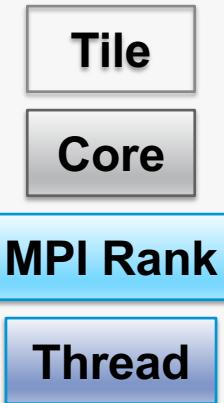
Theta - aprun Overview

<https://www.alcf.anl.gov/user-guides/running-jobs-xc40>

- Theta's KNL nodes have 32 tiles with 2 cores each (4 hardware threads per core)
- Example #1: 2 nodes, 64 ranks/node, 1 thread/rank, 1 rank/core
 - `aprun -n 128 -N 64 -d 1 -j 1 --cc depth <app> <app_args>`



```
nname= nid02937 rnk= 0 tid= 0 ht= {0}  
nname= nid02937 rnk= 1 tid= 0 ht= {1}  
nname= nid02937 rnk= 2 tid= 0 ht= {2}  
nname= nid02937 rnk= 3 tid= 0 ht= {3}  
nname= nid02937 rnk= 4 tid= 0 ht= {4}  
nname= nid02937 rnk= 5 tid= 0 ht= {5}
```



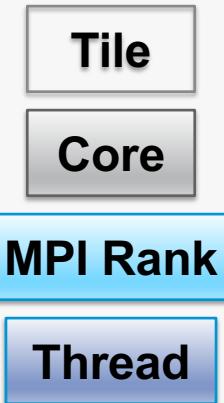
Theta - aprun Overview

<https://www.alcf.anl.gov/user-guides/running-jobs-xc40>

- Theta's KNL nodes have 32 tiles with 2 cores each (4 hardware threads per core)
- Example #1: 2 nodes, 32 ranks/node, 4 thread/rank, 2 threads/core
 - `aprun -n 64 -N 32 -d 4 -j 2 --cc depth -e OMP_NUM_THREADS=4 <app> <app_args>`



```
nname= nid02937 rnk= 0 tid= 0 ht= {0}  
nname= nid02937 rnk= 0 tid= 1 ht= {1}  
nname= nid02937 rnk= 0 tid= 2 ht= {64}  
nname= nid02937 rnk= 0 tid= 3 ht= {65}  
nname= nid02937 rnk= 1 tid= 0 ht= {2}  
nname= nid02937 rnk= 1 tid= 1 ht= {3}
```



Theta - aprun Overview

<https://www.alcf.anl.gov/user-guides/running-jobs-xc40>

- Affinity
 - Use `-d` and `--cc depth` to let ALPS control affinity
 - Use `--cc none` if you want to use OpenMP (or KMP) env. variables to specify affinity
- Core specialization with `-r <number.hardware_threads>`
 - Offload OS and MPI to unused hardware threads (e.g. reduce variability)
- Allocating memory in flat mode
 - Default memory allocation in DDR (NUMA 0)
 - Only allocate memory to HBM (NUMA 1): `numactl -m 1`
 - Prefer memory allocation to HBM: `numactl -p 1`
 - Example: `aprun -n 128 -N 64 -d 1 -j 1 --cc depth numactl -m 1 <app> <app_args>`

Theta - Now That Your Job Is Queued

<https://www.alcf.anl.gov/user-guides/cobalt-job-control>

- Check status of submitted jobs with `qstat`
- Format of output can be customized with `--header`

> `qstat --header JobID:User:WallTime:Nodes:State:Queue`

JobID	User	WallTime	Nodes	State	Queue
<hr/>					
123456	user1	24:00:00	4000	running	default
123457	user2	03:00:00	2048	queued	default
123458	user3	03:00:00	128	running	default
123459	user1	00:30:00	1	running	debug-cache-quad
123460	user4	00:30:00	64	queued	training

Theta - Now That Your Job Is Queued

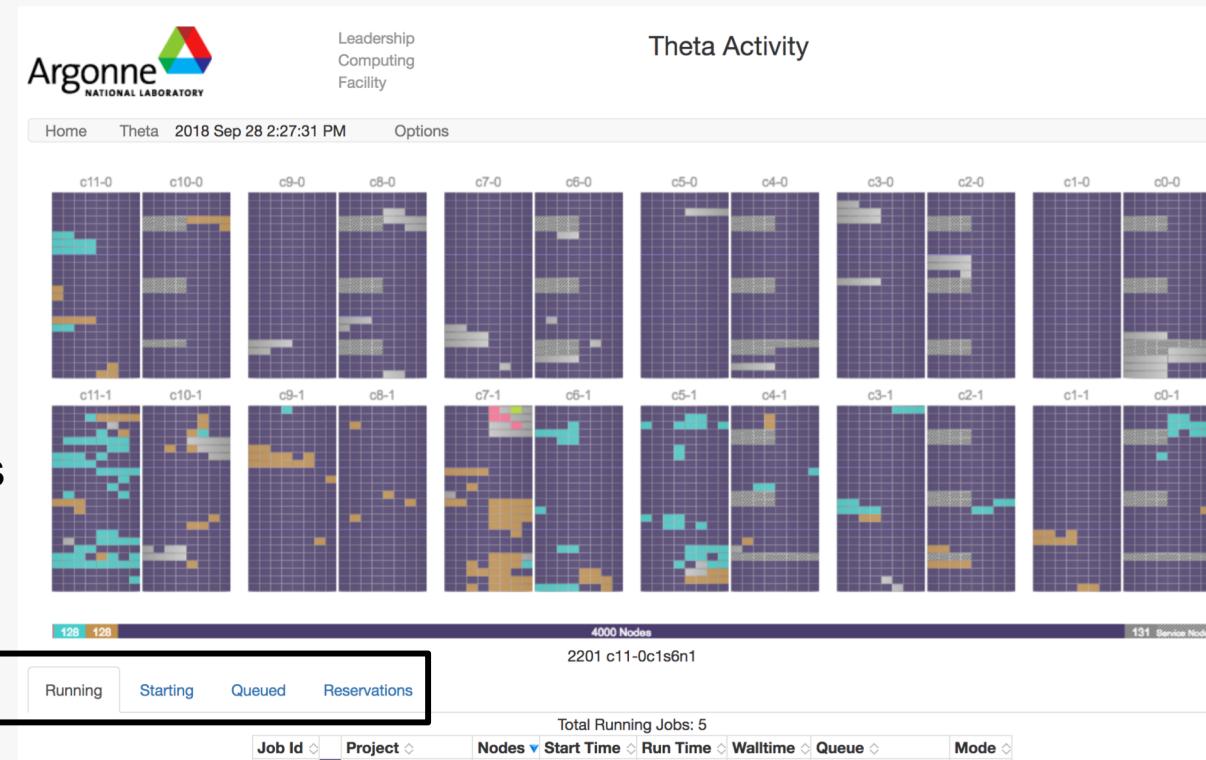
<https://www.alcf.anl.gov/user-guides/cobalt-job-control>

- Additional qstat queries
 - Show more job details: `qstat -f <jobid>`
 - Show all job details: `qstat -fl <jobid>`
 - Show all jobs from user: `qstat -u <user>`
 - Show information about queues: `qstat -Q`
- Delete job from queue: `qdel <jobid>`
- Alter properties of queued job
 - Change walltime: `qalter -t <new_time> <jobid>`
 - Change number of nodes: `qalter -n <new_number_of_nodes> <jobid>`
 - Change queue: `qmove <new_queue> <jobid>`

Theta - Checking Status of Job

<https://status.alcf.anl.gov/theta/activity>

Running
Starting
Queued
Reservations



Theta - Cobalt Files For Submitted Job

<https://www.alcf.anl.gov/user-guides/cobalt-job-control>

- Cobalt will create three files per job
 - Prefix defaults to jobid if not set with qsub's `-O` option
- Cobalt log file: `<prefix_name>.cobaltlog`
 - Create when job is submitted, additional info written while job runs
 - Contains submission information from qsub, runjob, and environment
- Job stderr file: `<prefix_name>.error`
 - Created at start of job
 - Contains job startup information and any output sent to standard error
- Job stdout file: `<prefix_name>.output`
 - Created at start of job
 - Contains content sent to standard output

Theta - Queues

<https://www.alcf.anl.gov/user-guides/job-scheduling-policy-xc40-systems>

- Jobs are routed to single `default` queue
 - Nodes allocated to job will be rebooted (if needed) for requested memory mode
 - Best to always specify memory mode (e.g. `--attrs mcdrum=cache:numa=quad`)
 - Pad requested walltime by ~30 minutes to account for possible rebooting
 - Don't delete job if remains in "starting" for several minutes
- Wall-clock limits are function of number of requested nodes
 - minimum allocation: 128 nodes, maximum walltime 3 hours
 - capability jobs: >= 802 nodes, maximum walltime 24 hours
 - [Check website for current policies](#)
- Two 16-node debug queues available
 - `debug-cache-quad`
 - `debug-flat-quad`

Theta - Queues

<https://www.alcf.anl.gov/user-guides/job-scheduling-policy-xc40-systems>

<https://www.alcf.anl.gov/user-guides/running-jobs-xc40>

- Jobs smaller than 128 nodes in default queue will be allocated 128 nodes
- Consider ensembles instead
 - Bundling multiple smaller jobs into single submission (e.g. multiple apruns)
 - Use a workflow system such as Balsam: <https://www.alcf.anl.gov/balsam>
- For long sequences of jobs, chain them together with dependencies
 - Dependent jobs inherit score boost from previous successful job in chain.

ANY QUESTIONS?



Cooley

<https://www.alcf.anl.gov/user-guides/cooley>

- Cooley serves as an analysis and visualization resource for projects
- x86+GPU system with 126 compute nodes
- 293 TF peak performance
- Each compute node has
 - two 2.4 GHz Intel Haswell processors (6 cores per CPU, 12 cores total)
 - one NVIDIA Tesla K80 (with two GPUs)
 - 384 GB CPU RAM, 12 GB RAM per GPU
 - 345 GB local scratch space
- FDR Infiniband interconnect
- Mira's GPFS and Theta's Lustre project filesystems mounted



Cooley - Softenv

<https://www.alcf.anl.gov/user-guides/cooley>

- Cooley uses softenv instead of modules
- Keys are read at login time to set environment variables
 - Mira, Cetus, Vesta: `~/.soft`
 - Cooley: `~/.soft.cooley`
- To get started:

```
# Select latest version of mvapich2 with GNU compilers  
+mvapich2  
@default
```

the end - do not put any keys after @default

- After edits to `.soft.cooley`, type `resoft` or log out and back in again
- Type `softenv` to see list of all available keys



Cooley - Compilers

<https://www.alcf.anl.gov/user-guides/cooley>

- Choose compiler via softenv keys
- Non-MPI compilers
 - GNU: [+gcc-8.2.0](#) (gcc, g++, gfortran)
 - Intel: [+intel-composer-xe](#) (icc, ifort)
 - Clang: [@clang](#) (clang)
- MPI Compiler wrappers
 - mvapich (mpicc, mpicxx, mpifort)
 - GNU: [+mvapich2](#)
 - Intel: [+mvapich2-intel](#)
 - Clang: [@mvapich2-clang](#) (no mpifort)
 - mpich (mpicc, mpicxx, mpif77, mpif90)
 - GNU: [+mpich2-1.4.1p1](#)
 - Intel: [+mpich2-1.4.1p1-intel](#)



Cooley - Job Script

<https://www.alcf.anl.gov/user-guides/cooley>

- Job script similar to Theta except mpirun instead of aprun
 - Example test.sh

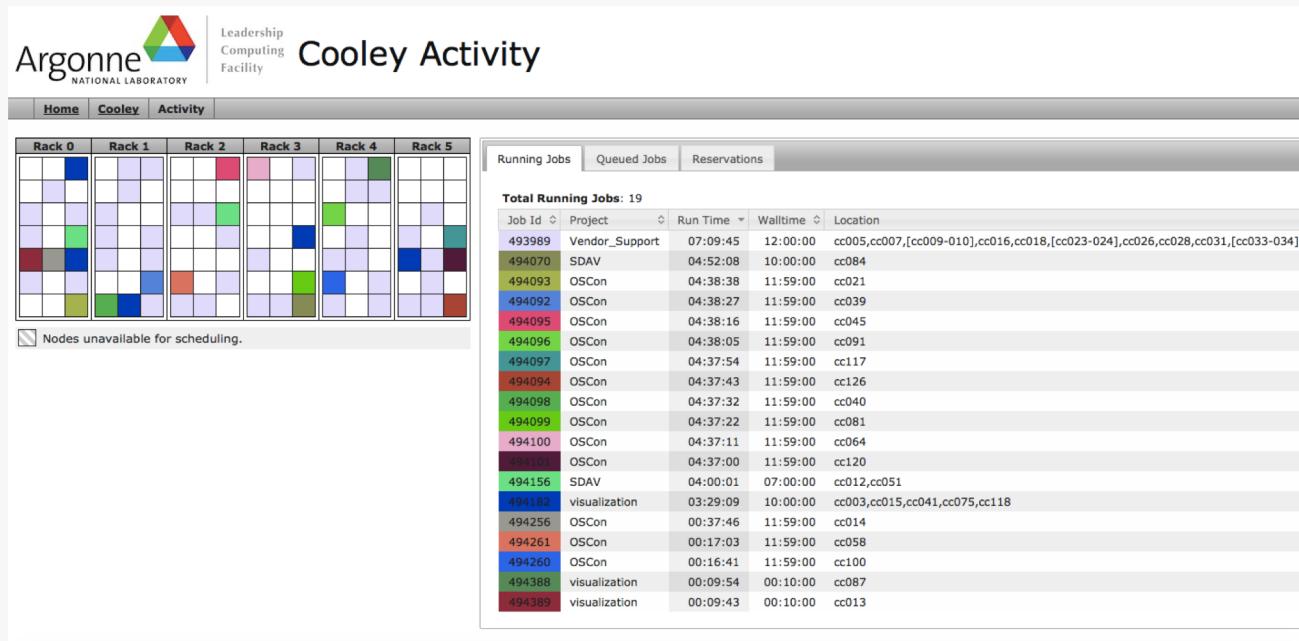
```
#!/bin/sh
NODES=`cat $COBALT_NODEFILE | wc -l`
PROCS=$((NODES * 12))
mpirun -f $COBALT_NODEFILE -n $PROCS myprog.exe
```

- Submit executable script on 5 nodes for 10 minutes
 - `qsub -q training -n 5 -t 10 -A SDL_Workshop ./test.sh`
- Queues
 - `default` for large/long jobs, `debug` for short/small jobs
 - `pubnet` to get public network visibility
 - `nox11` queues to suppress X server for CUDA jobs



Cooley - Checking Status of Job

<https://status.alcf.anl.gov/cooley/activity>



ANY QUESTIONS?



Why Hasn't My Job Started?

- There is a reservation which delays your job from starting
 - List all reservations currently in place: [showres](#)
- Job on Theta is in "starting" state; nodes being rebooted into memory mode requested.
- There are no available nodes for the requested queue
 - Nodes may be down, busy running other jobs, draining next job, or reserved
 - Check queue status: [qstat](#)
 - Check machine status: <http://status.alcf.anl.gov>
 - Check "ALCF Weekly Updates" for training, reservation, and maintenance notices
- List status of nodes on Theta & Cooley: [nodelist](#)

Theta - Core Files and Debugging

<https://www.alcf.anl.gov/user-guides/debugging-profiling>

- Abnormal Termination Processing (ATP)
 - Set environment variable ATP_ENABLED=1 in job script before aprun
 - Upon failure, generate merged stack backtrace tree in atpMergedBT.dot file
 - View output file with stat-view after loading with module load stat
- Notes on linking your application
 - PrgEnv-cray links everything necessary by default
 - PrgEnv-intel requires –WI,–T/opt/cray/pe/cce/default/cce/x86-64/lib/2.23.1.cce.Id
- Other debugging tools
 - You can generate STAT snapshots asynchronously
 - Full-featured debugging with DDT

When Things Go Wrong Running...

<https://www.alcf.anl.gov/user-support>

- Examine core files
- Best to save all three files generated by cobalt
 - <prefix_name>.cobaltlog, <prefix_name>.error, and <prefix_name>.output
- Retain important information
 - Jobid, machine name, copy/location of all files, exact error message
- Contact us
 - Your ALCF contact
 - Email: support@alcf.anl.gov
 - Call the ALCF Help Desk
 - Hours: Monday-Friday, 9am-5pm CT
 - Phone: 630-252-3111 or 866-508-9181 (toll-free, US only)



HAPPY COMPUTING!

HANDS-ON SESSION

Hands-on session

- On Theta or Cooley, copy examples from /projects/SDL_Workshop/examples
- This week's workshop has its own project and queue names
 - Theta: [-A SDL_Workshop -q training](#)
 - Cooley: [-A SDL_Workshop -q training](#)
- Theta/Cooley: compile & submit
 - [make](#)
 - [qsub submit.sh](#)

Hands-on session - Theta Compilation Example

- Create directory, copy files, compile and submit
 - > mkdir -p /projects/SDL_Workshop/\$USER
 - > cp -r /projects/SDL_Workshop/examples /projects/SDL_Workshop/\$USER
 - > cd /projects/SDL_Workshop/\$USER/examples/theta/compilation
 - > make
- Submit job and check output
 - > qsub submit.sh
 - > qstat -u \$USER
 - > cat <JobID>.output
- qsub echos a cobalt JobID to the screen. In the absence of a -o argument, three files are created (say JobID was 123456):
123456.cobaltlog, 123456.error, 123456.output (replaced by hellompi.output with -o)

Hands-on session - Theta Examples

- Example of an OpenMP job submission
 - Change to directory, compile, and submit

```
> cd /projects/SDL_Workshop/$USER/examples/theta/omp
> make
> qsub submit.sh
```
 - Note, remember that thread affinity is controlled by aprun settings (see slides 14-18 for reference)
- Example of a Python job submission
 - Change to directory, compile, and submit

```
> cd /projects/SDL_Workshop/$USER/examples/theta/python
> qsub submit.sh
```
 - Note, examine submit.sh script for loading python environment on Theta

Hands-on session - Cooley Examples

- Example of an OpenMP job submission
 - Change to directory, compile, and submit

```
> cd /projects/SDL_Workshop/$USER/examples/cooley/omp
> make
> qsub submit.sh
```
 - Remember to edit your `~/.soft.cooley` file and add compiler & MPI keys. Note, `@default` should be the last line in your file.
- Example of a Python job submission
 - Edit your `~/.soft.cooley` and add “`+anaconda`” before `@default`
 - Update your environment to include python paths

```
> resoft
```
 - Change to directory, compile, and submit

```
> cd /projects/SDL_Workshop/$USER/examples/theta/python
> qsub submit.sh
```