Quick Start on ATPESC Computing Resources

JaeHyuk Kwack
Argonne National Laboratory

Date 08/01/2021
AVAILABLE RESOURCE FOR ATPESC

- ALCF Systems
  - KNL (Theta)
  - x86+K80 GPU (Cooley)
  - x86+A100 GPUs (thetaGPU)
- OLCF
  - IBM Power9+NVIDIA V100 GPU (Ascent)
- NERSC
  - KNL+Haswell (Cori)
- Intel DevCloud
  - Intel Gen9 GPUs
- AMD Accelerator Cloud (AAC)
  - AMD MI-100 GPUs
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).
## Leadership Computing Facility System

<table>
<thead>
<tr>
<th></th>
<th>Argonne LCF</th>
<th>Oak Ridge LCF</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System</strong></td>
<td>Cray XC40</td>
<td>Cray</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Theta</td>
<td>Aurora in 2022</td>
</tr>
<tr>
<td><strong>Compute nodes</strong></td>
<td>4,392</td>
<td>4608</td>
</tr>
<tr>
<td><strong>Node architecture</strong></td>
<td>Intel Knights Landing, 64 cores</td>
<td>Intel Xeon + Intel GPU</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Processing Units</strong></td>
<td>281,088 Cores</td>
<td>202,752 POWER9 Cores + 27648 GPUs</td>
</tr>
<tr>
<td><strong>Memory per node, (gigabytes)</strong></td>
<td>192 DDR4 + 16 MCDRAM</td>
<td>-</td>
</tr>
<tr>
<td><strong>Peak performance, (petaflops)</strong></td>
<td>11.69</td>
<td>Exascale</td>
</tr>
</tbody>
</table>
ALCF Systems

- **Theta - Cray XC40**
  - 4,392 nodes / 281,088 cores

- **ThetaGPU – NVIDIA DGX A100**
  - 24 DGX A100 nodes, each with
    - Two AMD Rome 64-core processors
    - Eight NVIDIA A100 GPUs with 40 GB HBM 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
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

- **GPFS**
  - Home directories (/home) are in /gpfs/mira-home
    - Default quota 50GiB
    - Your home directory is backed up

- **Lustre**
  - Project directory locations (/grand) in /lus/grand/projects
    - Theta, ThetaGPU, Cooley: /grand/ATPESC2021
      - CREATE A SUBDIRECTORY /grand/ATPESC2021/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 (Theta, ThetaGPU ONLY)

- 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 ATPESC2021

# 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 ATPESC2021” (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 ATPESC2021 -A ATPESC2021`
- 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`
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
Machine status web page

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:

  ```bash
  $ 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
  ``

- 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:
    ```bash
    #!/bin/bash
    NODES=`cat $COBALT_NODEFILE \| wc -l`
    PROCS=$((NODES * 16))
    mpirun -n $PROCS myprog.exe
    ```
  - Submit on 1 nodes for 30 minutes
    qsub -n 1 -t 30 -q training -A ATPESC2021 ./test.sh
  - Submit on 1 nodes for 30 minutes for an interactive job
    qsub -l -n 1 -t 30 -q training -A ATPESC2021
  - Refer to online user guide for more info
    - [https://www.alcf.anl.gov/support-center/theta-gpu-nodes](https://www.alcf.anl.gov/support-center/theta-gpu-nodes)
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
Similar to **modules** package

Keys are read at login time to set environment variables like PATH.

Cooley: `~/.soft.cooley`

To get started:

```bash
# This key selects Intel compilers to be used by mpi wrappers
+mvapich2-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:
    ```sh
    #!/bin/sh
    NODES=`cat $COBALT_NODEFILE | wc -l`
    PROCS=$((NODES * 12))
    mpirun -f $COBALT_NODEFILE -n $PROCS myprog.exe
    ```
  - Submit on 5 nodes for 10 minutes
    ```sh
    qsub -n 5 -t 10 -q training -A ATPESC2021 ./test.sh
    ```
  - Refer to online user guide for more info
ALCF References

• Sample files (Theta, ThetaGPU, Cooley)
  – /grand/ATPESC2021/EXAMPLES/track-0-getting-started/GettingStarted

• Online docs
  – [https://www.alcf.anl.gov/support-center](https://www.alcf.anl.gov/support-center)
  – Getting Started Presentations (*slides and videos*)
    • Theta and Cooley
  – Debugging:
Cryptocard tips

- 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
ATPESC Resources

ALCF – Theta, ThetaGPU and Cooley

- **Project name:** ATPESC2021

- **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](mailto:slack!!) and [support@alcf.anl.gov](mailto:support@alcf.anl.gov)

- **Reservations:** Please check the details of the reservations directly on each machine (**command:** showres)

- **Queue:** Theta: ATPESC2021 ThetaGPU, Cooley: training (check showres) or default for running without reservation
ATPESC Resources

OLCF – Ascent

- 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 – Cori (Cray XC40)

- 9688 KNL nodes, each with
  - 68 physical cores
  - 96 GB DDR4 memory
  - 16 GB MCDRAM

- 2388 Haswell (16-core) nodes, each with
  - 32 physical cores
  - 128 GB memory

- Reference
  
  https://docs.nersc.gov/systems/cori/
ATPESC Resources

**Intel – DevCloud**

- Intel Gen9 GPU nodes
  - Intel HD Graphics 630 (GT2): an integrated GPU
    - 24 Execution Units (EUs) @ up to 1.15 GHz

- Request account
ATPESC Resources

AMD – Accelerator Cloud (AAC)

- Total GPUs per Node: up to 8
- GPU: AMD Instinct MI100 (32GB)
- GPU driver: ROCm 4.2.0
- CPU: AMD EPYC 7742 64-core processor
- CPU Clock Speed: 2.25 GHz
- Total CPU: 2
- System memory: 512 GB

- Up to 42 MI-100 GPUs are available for ATPESC
  - Job scheduler is not available; manual mapping from users to GPUs is required.
- See documents in your Argonne Folder for additional information
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
Hands-on exercise: Theta

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

- `$ module list` # See loaded modules

- `$ 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/ATPESC2021
  # Go to the project folder
- $ cd usr
  # Go to user space under project
- $ mkdir {your_username}
  # Create your space
- $ cd {your_username}

- $ cd GettingStarted/theta/

- $ more hellompi.c
  # See the example source
- $ more Makefile
  # An example of how to compile a code
- $ more submit.sh
  # An example of job script
Hands-on exercise: Theta

• $ 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 -l -n 1 -t 30 -A ATPESC2021 -q ATPESC2021

  # Start an interactive job mode

  Wait for job 536985 to start...
  Opening interactive session to 3834
  Currently Loaded Modulefiles:
  1) modules/3.2.11.4
  2) alps/6.6.59-7.0.2.1.3.77_g872a8562.ari
  3) nodedstat/2.3.89-7.0.2.1.2.68_g864515.ari
  4) sdb/3.3.812-7.0.2.1.2.85_gd6c4e58.ari
  5) uqreg/2.3.2-7.0.2.1.2.44_g8175d3d.ari
  6) ugni/6.0.14-0.7.0.2.1.2.69_ge78e5b0.ari
  7) gni-headers/5.0.12-0.7.0.2.1.2.27_g3b1768f.ari
  8) dmp/7.1.1-7.0.2.1.2.90_g38cf134.ari
  9) xpmem/2.2.20-7.0.2.1.2.67_g87eb960.ari
  10) lll/21.4.629-7.0.2.1.2.59_g8co6ef.ari

- $ cd /grand/ATPESC2021/usr
- $ cd {your_username}/GettingStarted/theta/
- $ aprun -n 4 ./helompi

jkwack@thetamom3:/gafs/mira-home/jkwack> cd /grand/ATPESC2021/usr
jkwack@thetamom3:/grand/ATPESC2021/usr> cd jkwack/GettingStarted/theta
jkwack@thetamom3:/grand/ATPESC2021/usr/jkwack/GettingStarted/theta> aprun -n 4 ./helompi
0: Hello!
1: Hello!
2: Hello!
3: Hello!
Application 24119210 resources: utime ~0s, stime ~1s, Rss ~7096, inblocks ~0, 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
Hands-on exercise: ThetaGPU

• $ source ~/.bashrc
• $ cd /grand/ATPESC2021
  # Go to the project folder
• $ cd usr/{your_username}
  # Go to your space under project
• $ cd GettingStarted/thetaGPU/
• $ more hellompi.c
  # See the example source
• $ more Makefile
  # An example of how to compile a code
• $ more submit.sh
  # An example of job script
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
Hands-on exercise: ThetaGPU

- `$ qsub -l -n 1 -t 30 -A ATPESC2021 -q training`  
  
  # Start an interactive job mode
Hands-on exercise: ThetaGPU

- $ cd /grand/ATPESC2021/usr/{your_username}/GettingStarted/thetaGPU/
- $ mpirun -n 4 ./hellompi

```
jkwack@thetaGPU19:~$ cd /grand/ATPESC2021/usr/jkwack/GettingStarted/thetaGPU/
jkwack@thetaGPU19:~$ mpirun -n 4 ./hellompi
0: Hello!
1: Hello!
2: Hello!
3: Hello!
```

- $ nvidia-smi

```
NVIDIA-SMI 450.142.00 Driver Version: 450.142.00 CUDA Version: 11.0

GPU Name  Temp  Perf  Memory Usage
 0 A10G-SXM-4GB  24C  P0  53W / 400W
 1 A10G-SXM-4GB  24C  P0  52W / 400W
 2 A10G-SXM-4GB  24C  P0  53W / 400W
 3 A10G-SXM-4GB  24C  P0  52W / 400W
 4 A10G-SXM-4GB  28C  P0  53W / 400W
 5 A10G-SXM-4GB  28C  P0  53W / 400W
 6 A10G-SXM-4GB  28C  P0  53W / 400W
 7 A10G-SXM-4GB  28C  P0  53W / 400W

Processes:
- No running processes found
```
Hands-on exercise: Cooley

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

- $ vi .soft.cooley
  # Update your environment

- $ cat .soft.cooley
  
  +mvapich2-intel
  +intel-composer-xe
  
  @default

- $ resoft  
  # Apply the updated environment

- $ which mpicc
  
  /soft/libraries/mpi/mvapich2/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 -l -n 1 -t 30 -A ATPESC2021 -q training`  # Start an interactive job mode

- `$ cd /grand/ATPESC2021/usr/{your_username}`  # Go to the project folder
- `$ cd GettingStarted/cooley/`  # Go to the example folder
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
Theta Memory Modes - IPM and DDR
Selected at node boot time

Cache

- **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

Flat

Hybrid
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