Argonne Training Program on Extreme-Scale Computing (ATPESC)

Quick Start on ATPESC Computing Resources

JaeHyuk Kwack
Argonne National Laboratory

Date 07/31/2022
AVAILABLE RESOUCE FOR ATPESC

- ALCF Systems
  - KNL (Theta)
  - x86+K80 GPU (Cooley)
  - x86+A100 GPUs (thetaGPU)

- OLCF
  - IBM Power9+NVIDIA V100 GPU (Ascent)

- NERSC
  - x86 + NVIDIA A100 GPU (Perlmutter)

- Cloud resources
  - NVIDIA cloud GPU resources
  - Intel DevCloud
  - AMD Accelerator Cloud (AAC)
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>System</td>
<td>Cray XC40</td>
<td>HPE</td>
</tr>
<tr>
<td>Name</td>
<td>Theta</td>
<td>Polaris</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Aurora in 2022</td>
</tr>
<tr>
<td>Compute nodes</td>
<td>4,392</td>
<td>560</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>4608</td>
</tr>
<tr>
<td>Node architecture</td>
<td>Intel Knights Landing, 64 cores</td>
<td>AMD Milan + 4x NVIDIA A100 GPU</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Intel Xeon + Intel GPU</td>
</tr>
<tr>
<td></td>
<td>Intel Xeon + Intel GPU</td>
<td>2 x IBM POWER9 22 cores 6 x NVIDIA V100 GPUs</td>
</tr>
<tr>
<td></td>
<td>AMD CPU + AMD GPU</td>
<td></td>
</tr>
<tr>
<td>Processing Units</td>
<td>281,088 Cores</td>
<td>17,920 AMD Milan Cores + 2240 GPUs</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>202,752 POWER9 Cores + 27648 GPUs</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Memory per node,</td>
<td>192 DDR4 + 16 MCDRAM</td>
<td>512 DDR4 + 160 HBM2 + 1600 SSD</td>
</tr>
<tr>
<td>(gigabytes)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>512 DDR4 + 96 HBM2 + 1600 NVM</td>
<td></td>
</tr>
<tr>
<td>Peak performance,</td>
<td>11.69</td>
<td>44</td>
</tr>
<tr>
<td>(petaflops)</td>
<td>Exascale</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td></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

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: /grand/projects/ATPESC2022
      - CREATE A SUBDIRECTORY /grand/projects/ATPESC2022/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 \leftarrow what is currently loaded
  - avail
  - load
  - unload
  - switch\|swap
  - use \leftarrow 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 ATPESC2022

# 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 ATPESC2022" (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 ATPESC2022 –A ATPESC2022
- 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
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 -l
    NODES=`cat $COBALT_NODEFILE | wc -l`
    PROCS=$((NODES * 16))
    mpirun -n $PROCS myprog.exe
  ```
- Submit on 1 node/gpu for 30 minutes
  ```bash
  qsub -n 1 -t 30 -q training-gpu -A ATPESC2022 ./test.sh # For 1 node with 8 GPUs
  qsub -n 1 -t 30 -q single-gpu -A ATPESC2022 ./test.sh # For 1 GPU
  ```
- Submit on 1 node/gpu for 30 minutes for an interactive job
  ```bash
  qsub -I -n 1 -t 30 -q training-gpu -A ATPESC2022 # For 1 node with 8 GPUs
  qsub -I -n 1 -t 30 -q single-gpu -A ATPESC2022 # For 1 GPU
  ```
- 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)
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:
    ```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 ATPESC2022 ./test.sh
    ```
  - Refer to online user guide for more info
ALCF References

• Sample files (Theta, ThetaGPU, Cooley)
  – /grand/projects/ATPESC2022/EXAMPLES/track-0-getting-started/GettingStarted

• Online docs
  – https://www.alcf.anl.gov/support-center
  – Getting Started Presentations (slides and videos)
    • Theta and Cooley
    • https://www.alcf.anl.gov/workshops/2019-getting-started-videos
  – 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

- **Project name:** ATPESC2022

- **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: showres)

- **Queue**
  - Theta: ATPESC2022, ThetaGPU: training-gpu, or single-gpu, Cooley: training (check showres) or default for running without reservation
ATPESC Resources

---

**OLCF – Ascent**

- **Ascent User Guide** [https://docs.olcf.ornl.gov/systems/ascent_user_guide.html](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](https://github.com/olcf-tutorials)

- See documents in your Argonne Folder for additional information

- **For other questions, email:** help@olcf.ornl.gov
NERSC – Perlmutter (HPE Cray XE)

- 1536 GPU-accelerated nodes, each with
  - 64 AMD Milan cores with 4 NVIDIA A100 GPUs
  - 256 GB DDR4 memory
  - 160 GB HBM2 memory on 4 GPUs
- 3072 AMD Milan (64-core) nodes, each with
  - 128 physical cores
  - 512 GB memory

- Reference

https://docs.nersc.gov/systems/perlmutter/
Cloud resources for Tools track

- NVIDIA Cloud GPU resources
  - Attendees interested in the NVIDIA tools hands-on session will need an NVIDIA developer account. They can sign up here ahead of time: https://developer.nvidia.com/
  - On the day of the tools track, we will screenshare this landing page URL and the accompanying access code: https://courses.nvidia.com/dli-event/

- Intel DevCloud (TBD)

- 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
Handson 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/projects/ATPESC2022  # 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

- $ export CRAYPE_LINK_TYPE=dynamic  # For dynamic linking
- $ module swap PrgEnv-intel PrgEnv-crays; module swap PrgEnv-crays PrgEnv-intel  # To avoid a bug
- $ cc -o hellomi hellomi.c  # Build the example
- $ make clean; make  # Another way to build the example
- $ aprun -n 4 ./hellomi  # 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 ATPESC2022 -q ATPESC2022

# Start an interactive job mode

$ qsub -l n 1 -t 30 -A ATPESC2022 -q ATPESC2022

$ cd /grand/projects/ATPESC2022/usr

$ cd {your_username}/GettingStarted/theta/

$ aprun -n 4 ./hellompi

jkwack@thetamom3:/grand/projects/ATPESC2022/usr/jkwack/GettingStarted/theta> aprun -n 4 ./hellompi
0: Hello!
1: Hello!
2: Hello!
3: Hello!

Application 27120510 resources: utime ~0s, stime ~2s, Rss ~10244, inblocks ~2590, 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/projects/ATPESC2022  # 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

jkwack@theta-pusn1:/grand/projects/ATPESC2022/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 -l -n 1 -t 30 -A ATPESC2022 -q single-gpu
  # Start an interactive job mode

```bash
jkwack@thetagusnl:/grand/projects/ATPESC2022/usr/jkwack/GettingStarted/thetaGPU$ qsub -l -n 1 -t 30 -A ATPESC2022 -q single-gpu
Job routed to queue "single-gpu".
WARNING: Filesystem attribute not set for this job submission.
This job will be set to request all filesystems. In the event
of a filesystem outage, this job may be put on hold unnecessarily.
Setting attrs to: {'filesystems': 'home,grand,eagle,theta-fs0'}
Wait for job 100906966 to start...
Opening interactive session to thetagpu06-gpu0
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-121-generic x86_64)

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

System information as of Sun 31 Jul 2022 02:09:05 AM CDT

  System load: 1.66  Users logged in: 0
  Usage of /: 1.5% of 1.72TB  IPv4 address for docker0: 172.17.0.1
  Memory usage: 2%  IPv4 address for enp22s0: 10.230.2.194
  Swap usage: 0%  IPv4 address for infiniband0: 172.23.2.194
  Processes: 3262  IPv4 address for infiniband0: 172.22.2.194

Last login: Sun Jul 31 01:22:28 2022 from thetagpusln1.mcp

Currently Loaded Modules:
  1) openmpi/openmpi-4.0.5  2) Core/StdEnv
```
Hands-on exercise: ThetaGPU

- $ cd /grand/projects/ATPESC2022/usr/{your_username}/GettingStarted/thetaGPU/

- $ mpirun -n 4 ./hellomi

- $ nvidia-smi

In the image:
- The script is executed to demonstrate the use of ThetaGPU.
- The output of `nvidia-smi` is shown, indicating the status of the NVIDIA device.
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 -l -n 1 -t 30 -A ATPESC2022 -q training  # Start an interactive job mode
- $ cd /grand/projects/ATPESC2022/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

```bash
[jkwack@cc043 ~]$ cd /grand/projects/ATPESC2022/usr/jkwack/GettingStarted/cooley/
[jkwack@cc043 cooley]$ make clean; make
/bin/rm -f *.error *.output *.cobaltlog hellompi
which mpicc
/software/libraries/mpi/openmpi-2.1.5/intel/bin/mpicc
mpicc -g -O0 -o hellompi hellompi.c
[jkwack@cc043 cooley]$ mpirun -n 4 ./hellompi
0: Hello!
1: Hello!
2: Hello!
3: Hello!
```
Theta Memory Modes - IPM and DDR
Selected at node boot time

- **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
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/