Getting Started on ALCF Systems

Simulation, Data, and Learning Workshop
October 4, 2022
Adrian Pope (on behalf of ALCF)

ALCF Systems



- **Polaris** (CPU+GPU)
  - Top500: Rmax 25.82 PFlop/s, Rpeak 34.16 PFlop/s
  - 560 nodes: 1x AMD EPYC Milan 7543P + 4x NVIDIA A100
- **Theta** (CPU)
  - Top500: Rmax 6.92 PFlop/s, Rpeak 11.66 PFlop/s
  - 4392 nodes: 1x Intel Xeon Phi 7230 (KNL)
- **ThetaGPU** (CPU+GPU)
  - GPU-accelerated computing pathfinder, Rpeak 3.9 PFlop/s
  - 24 nodes: 2x AMD EPYC Rome 7742 + 8x NVIDIA A100
- **Cooley** (CPU+GPU)
  - Visualization + Data Analysis, Rpeak 0.3 PFlop/s
  - 126 nodes: 2x Intel Haswell E5-2620 + 1x NVIDIA Tesla K80
- **AI Testbed** (various AI accelerators)
  - Available for Allocation Requests (DD): Cerebras CS-2, SambaNova DataScale
  - Access Forthcoming: Graphcore MK-1, Groq, Habana Gaudi

Polaris

https://www.alcf.anl.gov/polaris

- ALCF's latest computational resource
  - Provides on-ramp to Aurora
- Generally available ALCF resource
  - INCITE, ALCC, DD

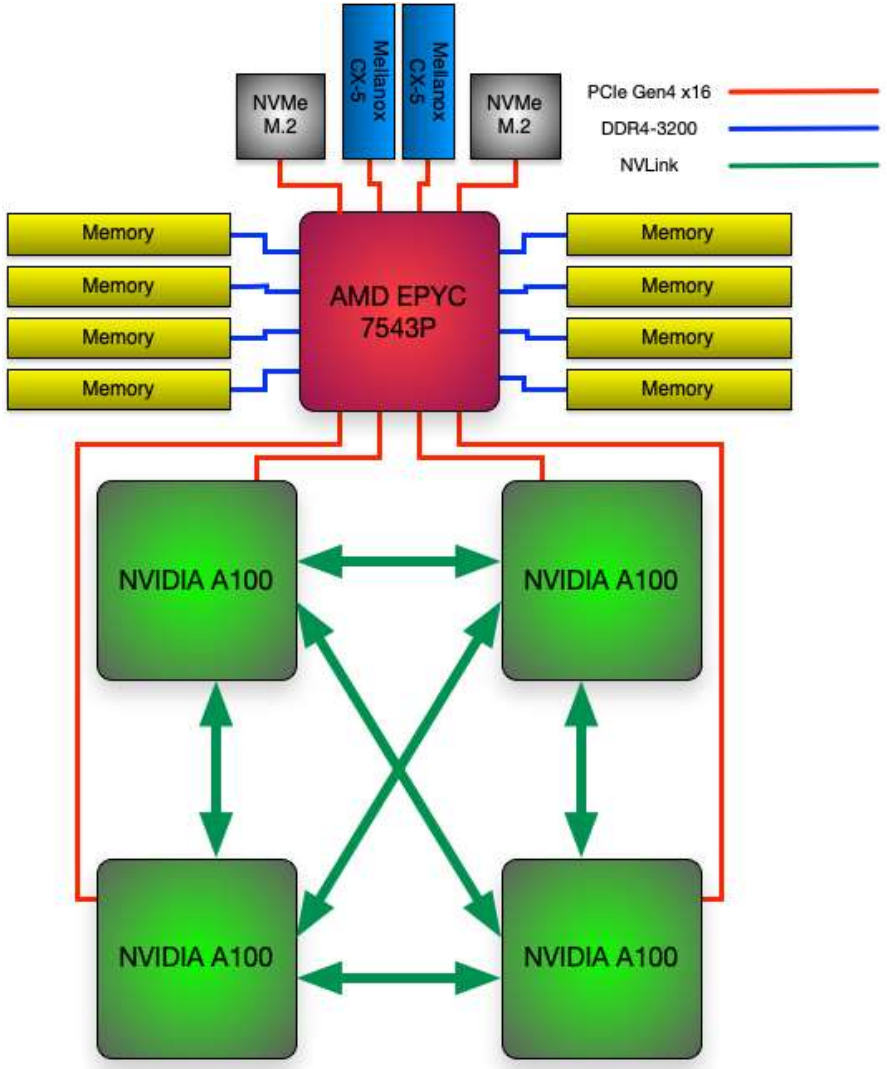
14	**Polaris** - Apollo 6500, AMD EPYC 7532 32C 2.4GHz, NVIDIA A100 SXM4 40 GB, Slingshot-10, HPE DOE/SC/Argonne National Laboratory United States	256,592	25.81	34.16

Top500 June 2022

Hardware

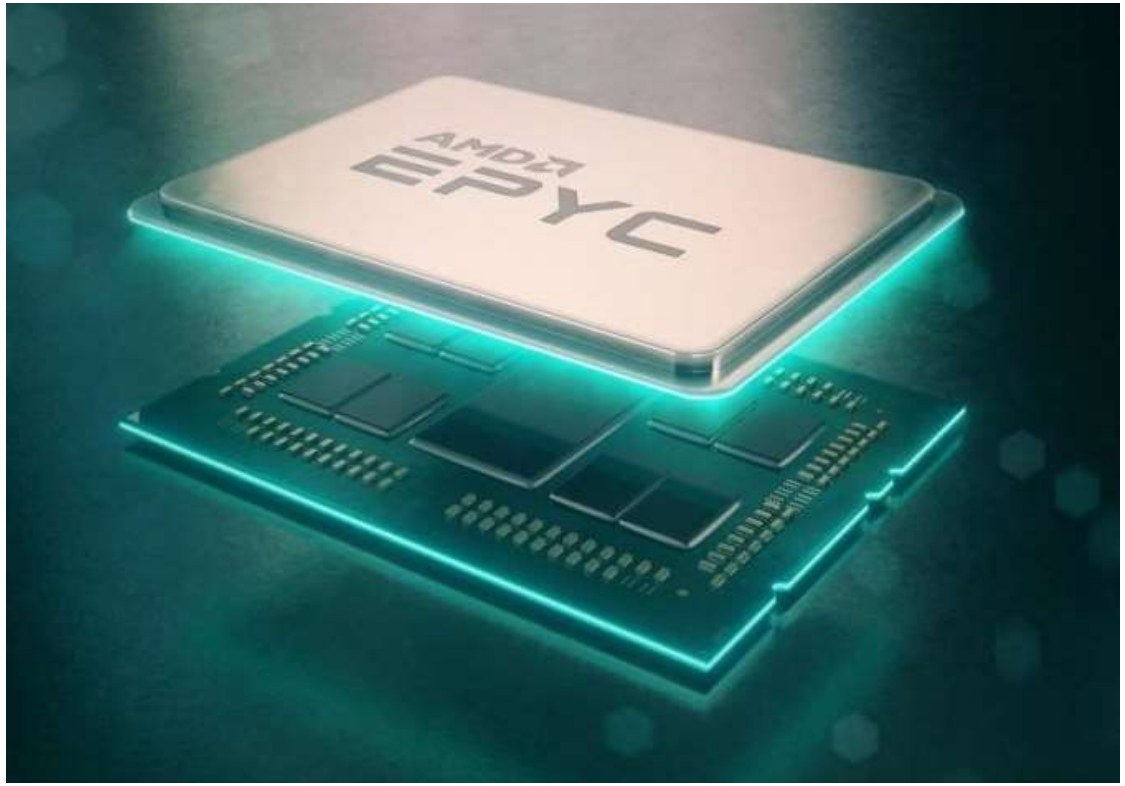
Polaris Single Node Configuration

# 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



Single AMD EPYC “MILAN” 7543P CPU Specs

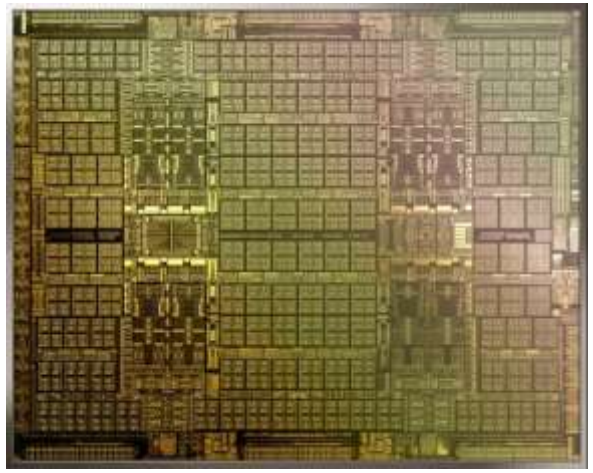
Base Frequency	2.8 GHz
Max Boost Clk	3.7 GHz
# of Zen3 Cores	32
# of Threads	64
Total DDR4 Memory	512 GB
# of Memory Channels	8
DDR4 Memory BW	204.8 GB/s
Total Shared L3 Cache	256 MB
L2 Cache per Core	512 KB
L1 Cache per Core	32 KB
PCIe Gen 4	128 lanes (8 ports)
PCIe Gen4 BW	64 GB/s
TDP	225 W



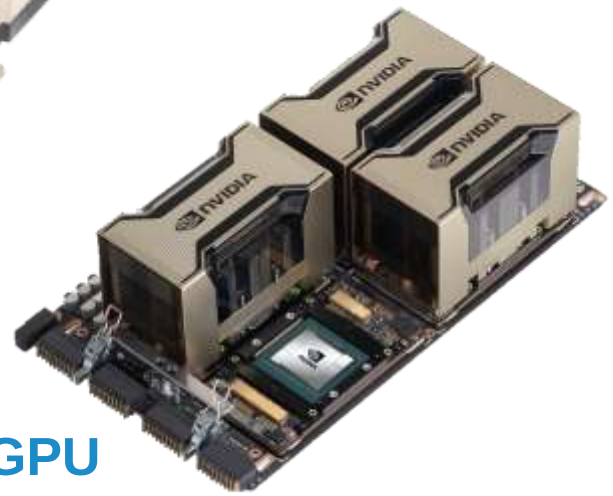
NVIDIA HGX A100 Specs

	A100 PCIe	HGX
FP64	9.7 TF	38.8 TF
FP64 Tensor Core	19.5 TF	78 TF
FP32	19.5 TF	78 TF
BF16 Tensor Core	312 TF	1.3 PF
FP16 Tensor Core	312 TF	1.3 PF
INT8 Tensor Core	624 TOPS	2496 TOPS
GPU Memory	40 GB HBM2	160 GB HBM2
GPU Memory BW	1.6 TB/s	6.4 TB/s
Interconnect	PCIe Gen4 64 GB/s	NVLink 600 GB/s
Max TDP Power	250W	400W

Ampere 7nm



A100 PCIe



HGX A100 4-GPU

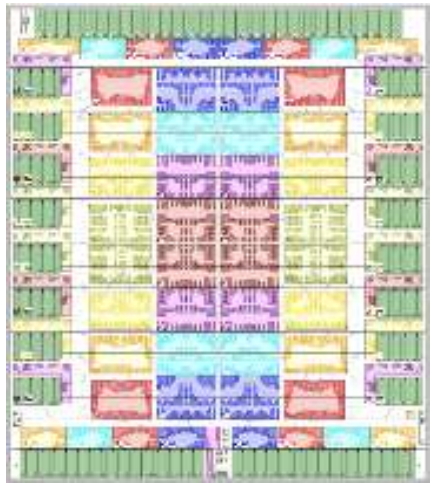
Node Local Storage

- Each compute node has two NVMe SSDs
  - 1.6 TB each / 3.2 TB total
- Similar to Theta, ALCF provides no specific software for using SSDs
- RAID0 volume that is user accessible
- Users access SSD via standard POSIX APIs
  - /local/scratch
- Data is destroyed when the job ends so any data users wish to keep must be moved to Grand or Eagle

Slingshot Interconnect

Rosetta Switch

- Multiple QoS levels
- Aggressive adaptive routing
- Advanced congestion control
- Very low average and tail latency
- High performance multicast and reduction



64 ports x 200 Gbps

SS-10 (100Gb)
Injection: ~14 TB/s
Bisection: ~24 TB/s

SS-11 (200Gb)
Injection: ~28 TB/s
Bisection: ~24 TB/s

Mellanox ConnectX NIC

Slingshot 10

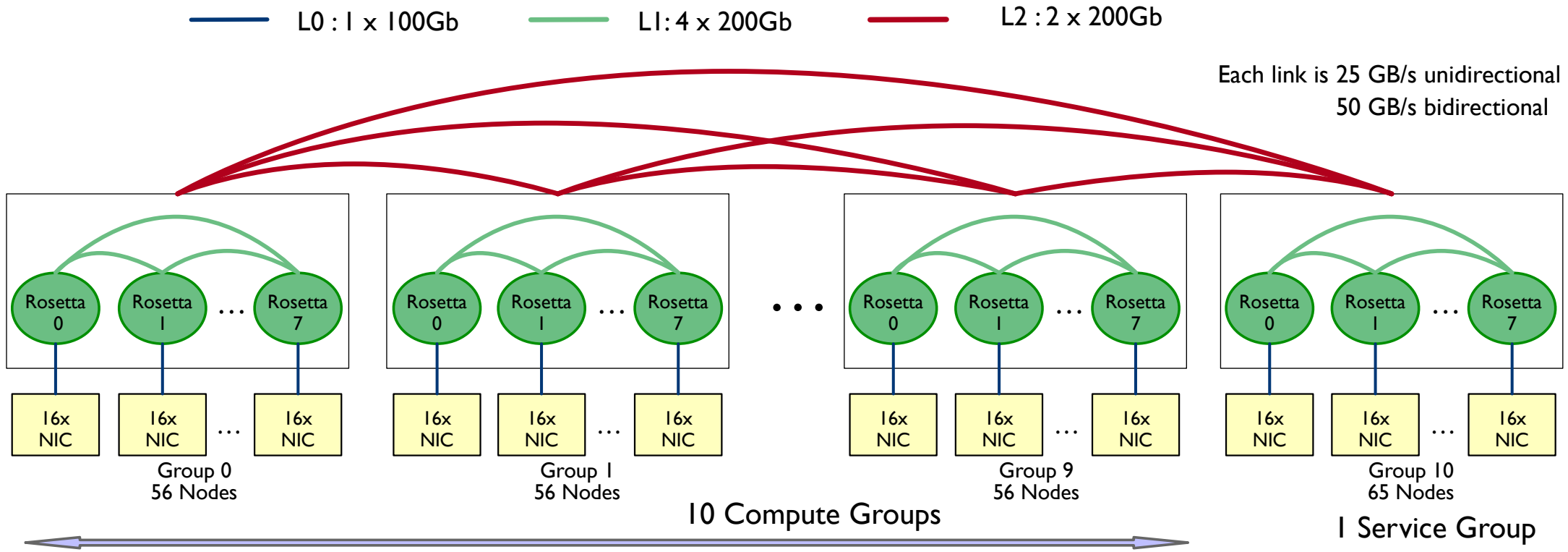
- HPE Cray MPI stack
- Ethernet functionality
- RDMA offload

Cassini NIC

Slingshot 11

- MPI hardware tag matching
- MPI progress engine
- One-sided operations
- Collectives
- 2X injection bandwidth

Slingshot Configuration



- 11 Total dragonfly groups, 10 compute groups and 1 non-compute group
- 2 links/arc between each group
- 4 links/arc within each group (between switches of a group)
- 1 link from each NIC (100Gb with SS10, 200Gb when upgraded to SS11)

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)	13 TB/s (28 TB/s)
Total GPU DP Tensor Core Flops	44 PF
Total Power	1.8 MW

Apollo 6500 Gen10+

Storage

Polaris will be connected to existing ALCF storage resources

- Grand – Global/Center-wide file system providing main project storage
  - 100 PB @ 650 GB/s
  - Accessed via Lustre LNET routers using Polaris gateway nodes
- Eagle – Community file system providing project storage that can be shared externally via Globus sharing
  - 100 PB @ 650 GB/s
  - Accessed via Lustre LNET routers using Polaris gateway nodes
- Gateway nodes can provide >1 TB/s
- Home – shared home file system for convenience not for performance or bulk storage

Software


2021 ARGONNE LEADERSHIP COMPUTING FACILITY
Argonne NATIONAL LABORATORY
Science

Login

- Requires ALCF account
  - https://accounts.alcf.anl.gov
  - Project: ? (INCITE, ALCC, DD = Director’s Discretionary)
  - Resource: polaris

- ssh -A username@polaris.alcf.anl.gov
  - Use your ALCF username
  - Password is pin + one-time-password

Filesystem

- Polaris shared common `/home` with other ALCF systems

- The Eagle and Grand filesystems available and mounted
  - `/lus/grand`
  - `/lus/eagle`

- Main project storage
  - `/lus/grand/projects`
  - `/lus/grand/projects/$PROJECT` ←

- Community project storage
  - `/lus/eagle/projects`

Modules

- Polaris provides modules (Lmod) as a convenient way to access HPE or ALCF provided libraries
- `module list`
  - Shows the currently loaded modules
- `module load <module>`
  - Loads a module into your environment
  - Only effects the current shell
- `module unload <module>`
  - Removes module from your environment
  - Only effects the current shell
- `module avail`
  - Lists all available modules
- `module use <path>`
  - Adds an alternate path to the module search path

```
x3109c0s25b1n0:~ # module list
Currently Loaded Modules:
1) craype-x86-rome
2) libfabric/1.11.0.4.125
3) craype-network-ofi
4) perftools-base/22.05.0
5) nvhpc/21.9
6) craype/2.7.15
7) cray-dsmml/0.2.2
8) cray-mpich/8.1.16
9) cray-pmi/6.1.2
10)cray-pmi-lib/6.0.17
11)cray-pals/1.1.7
12)cray-libpals/1.1.7
13)PrgEnv-nvhpc/8.3.3
14)craype-accel-nvidia80
```

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

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 -lwalltime=01:00:00
#PBS -lselect=4
#PBS -lsystem=polaris
#PBS -lfilesystems=home:eagle

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>
```

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

Scheduler – PBS Professional

- Resource requests and placement
  - Job wide options
    - `-l walltime=06:00:00`
  - Resource selection
    - `-l select=[<N>:]<chunk>[+[<N>:]<chunk> ...]`
  - Simple example with system selection (128 compute nodes on Polaris)
    - `-l select=128:system=polaris`

- Useful definitions
  - chunk
    - Set of resources allocated as a unit to a job
  - vnode
    - Virtual node. Abstract object representing a usable part of an execution host
  - ncpus
    - On Polaris this is equal to a hardware thread. Polaris has a single socket with 32 cores, each with 2 threads resulting in ncpus=64
  - ngpus
    - Number of GPUs. Generally will be four on Polaris. Could potentially be higher if using *Multi Instance GPU (MIG)* mode.

Some useful commands for working with PBS

- `qsub2pbs` (Available on Theta and Cooley)
  - Translates Cobalt qsub command to PBS qsub command
  - Pass it a Cobalt command line and get a pbs one

- `pbsnodes`
  - Provides information about the current state of nodes

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

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

MPI Control

- man mpi_intro
  - Lists various environment variables to control CrayMPI

Python

The Cray PE provides python (cray-python module) with several builtin modules

- `numpy`
- `scipy`
- `pandas`
- `mpi4py`

Debuggers & Profilers

- Debuggers
  - STAT (Stack Trace Analysis Tool)
    - Stack tracing at scale
  - gdb4hpc
    - Parallelized gdb for HPC
  - CUDA-GDB
    - NVIDIA tool for debugging CUDA
  - gdb

- Profilers
  - PAT (Performance Analysis Tool)
    - Whole program performance analysis
  - NVIDIA® Nsight™
    - GPU performance analysis tool

Information and Help

- User documentation will be added to the ALCF support center
  - https://www.alcf.anl.gov/support-center
- Additional information about Polaris
  - https://www.alcf.anl.gov/polaris
- Getting help for ALCF resources
  - support@alcf.anl.gov