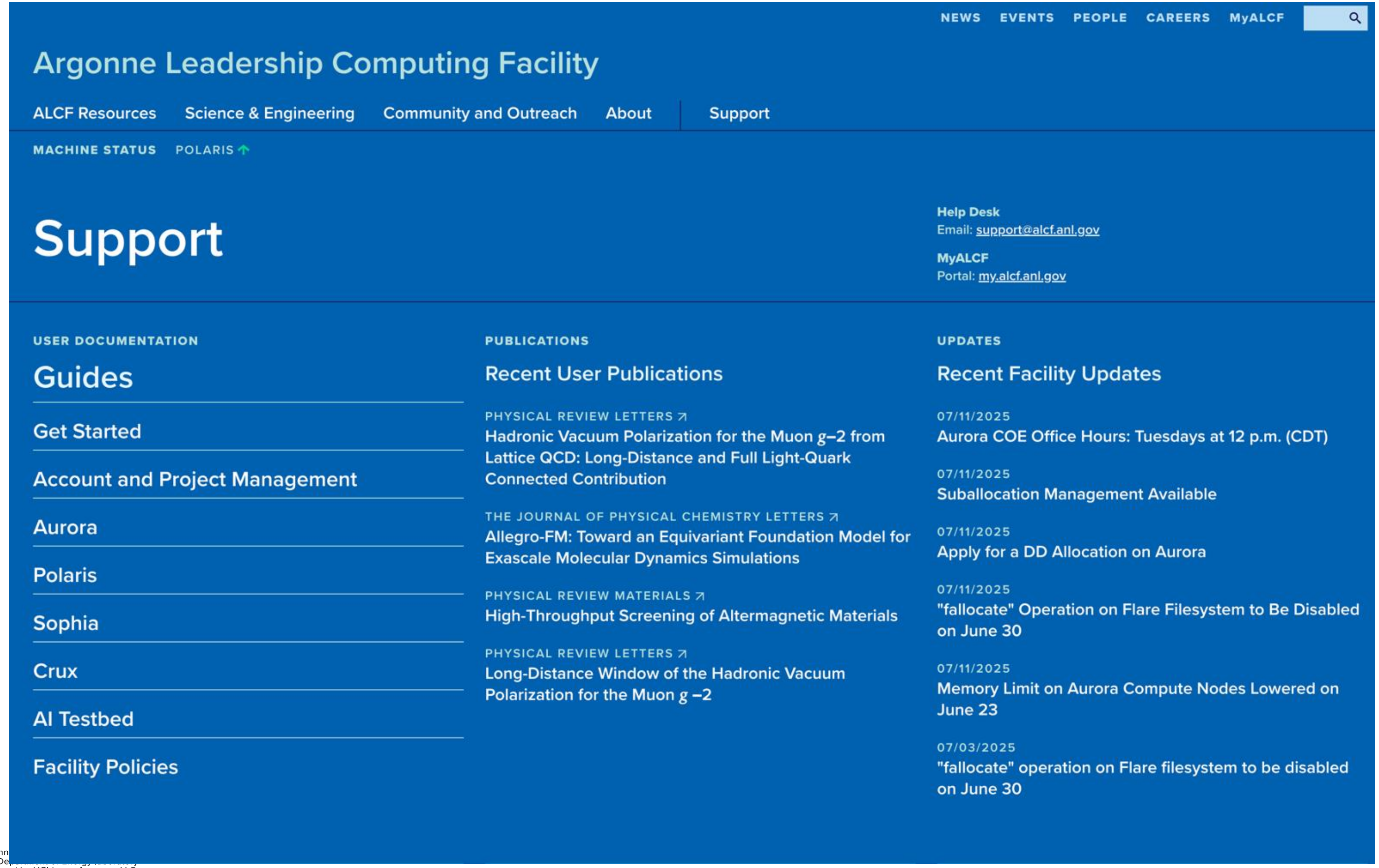
JULY 17, 2025

GETTING STARTED ON AURORA AND POLARIS BOOTCAMP

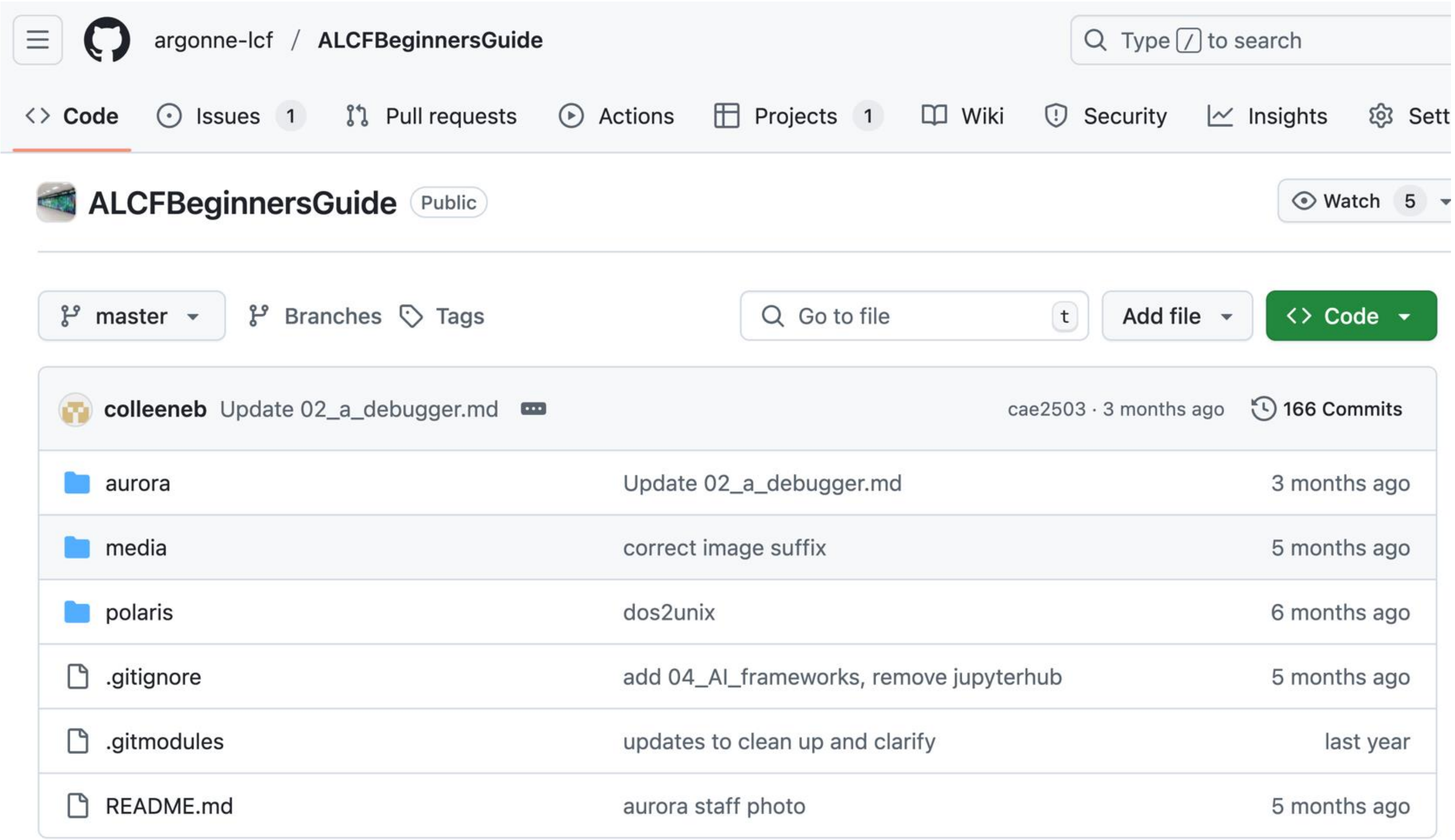
SOLOMON BEKELE
ARGONNE LEADERSHIP COMPUTING FACILITY

U.S. DEPARTMENT OF ENERGY
Argonne National Laboratory is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC.
Argonne NATIONAL LABORATORY

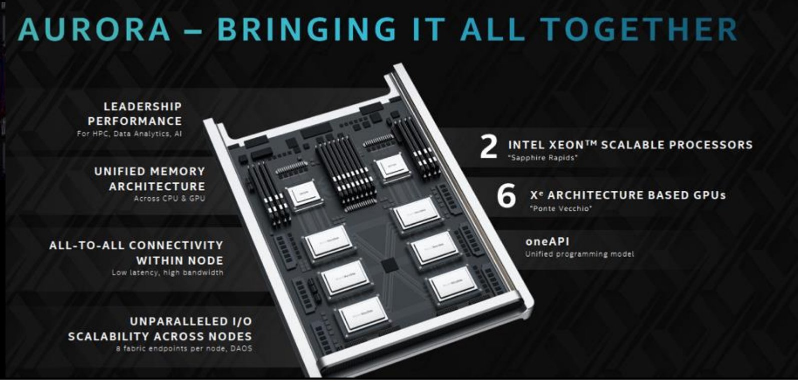
https://www.alcf.anl.gov/support-center



https://github.com/argonne-lcf/ALCFBeginnersGuide



AURORA



Peak Performance
≧ 2 Exaflops DP

Intel GPU
Intel® Data Center GPU Max Series 1550

Intel Xeon Processor
Intel® Xeon Max Series 9470C CPU with High Bandwidth Memory

Platform
HPE Cray-Ex

Compute Node
2x Intel® Xeon Max Series processors
6x Intel® Data Center GPU Max Series
8x Slingshot11 fabric endpoints

GPU Architecture
Intel XeHPC architecture
High Bandwidth Memory

Node Performance
>130 TF

System Size
166 Cabinets
10,624 Nodes
21,248 CPUs
63,744 GPUs

System Memory
1.36PB HBM CPU Capacity
10.9PB DDR5 Capacity
8.16PB HBM GPU Capacity

System Memory Bandwidth
30.58PB/s Peak HBM BW CPU
5.95PB/s Peak DDR5 BW
208.9PB/s Peak HBM BW GPU

High-Performance Storage
230PB
31TB/s DAOS bandwidth
1024 DAOS Nodes

System Interconnect
HPE Slingshot 11
Dragonfly topology with adaptive routing

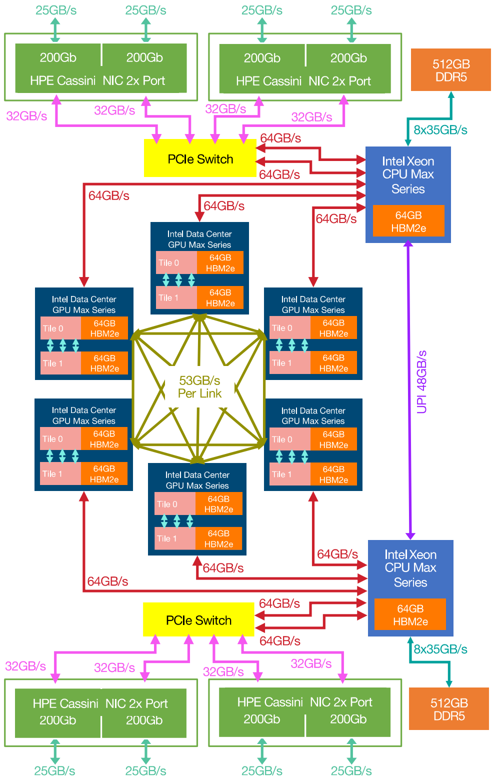
System Interconnect BW
Peak Injection BW 2.12PB/s
Peak Bisection BW 0.69PB/s

Network Switch
25.6 Tb/s per switch (64x 200 Gb/s ports)
Links with 25 GB/s per direction

Programming Environment
- C/C++, Fortran
- SYCL/DPC++
- OpenMP 5.0
- Kokkos, RAJA

AURORA EXASCALE COMPUTE BLADE

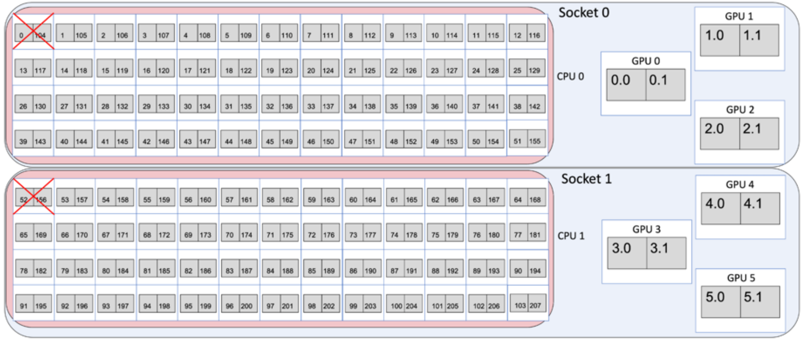
NODE CHARACTERISTICS	
6	GPU - Intel Data Center GPU Max Series (#)
2	CPU - Intel Xeon CPU Max Series (#)
768	GPU HBM Memory (GB)
19.66	Peak GPU HBM BW (TB/s)
128	CPU HBM Memory (GB)
2.87	Peak CPU HBM BW (TB/s)
1024	CPU DDR5 Memory (GB)
0.56	Peak CPU DDR5 BW (TB/s)
≧ 130	Peak Node DP FLOPS (TF)
200	Max Fabric Injection (GB/s)
8	NICs (#)



NODE TOPOLOGY: SOCKETS AND CORES

- 52 physical cores per socket
  - 104 logical CPUs per Socket (with SMT or Hyper-Threading enabled)
  - **Numbering starts from Thread 0 of Core 0 of Socket 0.**
  - CPU-bind
    - depth
    - list

```
CPU   0  → Socket 0, Core 0, Thread 0
CPU   1  → Socket 0, Core 1, Thread 0
...
CPU  51  → Socket 0, Core 51, Thread 0
CPU  52  → Socket 1, Core 0, Thread 0
...
CPU 103  → Socket 1, Core 51, Thread 0

=> Second hardware thread starts here

CPU 104  → Socket 0, Core 0, Thread 1
CPU 105  → Socket 0, Core 1, Thread 1
...
CPU 155  → Socket 0, Core 51, Thread 1
CPU 156  → Socket 1, Core 0, Thread 1
...
CPU 207  → Socket 1, Core 51, Thread 1
```

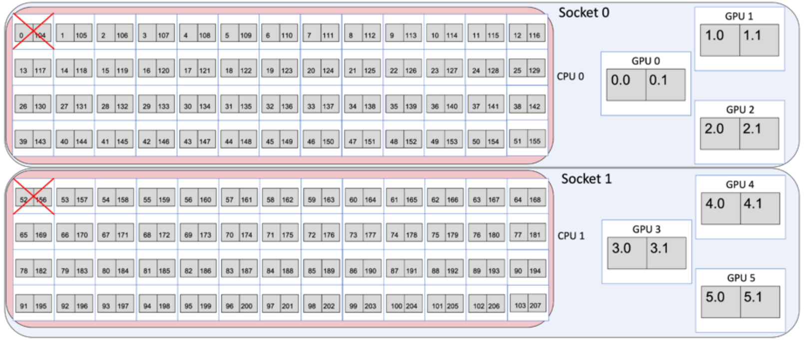


mpiexec -n 12 -ppn 12 --depth=8 --cpu-bind=depth ./app

NODE TOPOLOGY: SOCKETS AND CORES

- 52 physical cores per socket
  - 104 logical CPUs per Socket (with SMT or Hyper-Threading enabled)
  - **Numbering starts from Thread 0 of Core 0 of Socket 0.**
  - CPU-bind
    - depth
    - list

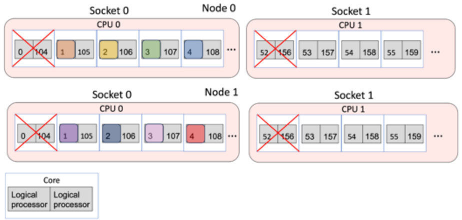
```
CPU   0  → Socket 0, Core 0, Thread 0
CPU   1  → Socket 0, Core 1, Thread 0
...
CPU  51  → Socket 0, Core 51, Thread 0
CPU  52  → Socket 1, Core 0, Thread 0
...
CPU 103  → Socket 1, Core 51, Thread 0

=> Second hardware thread starts here

CPU 104  → Socket 0, Core 0, Thread 1
CPU 105  → Socket 0, Core 1, Thread 1
...
CPU 155  → Socket 0, Core 51, Thread 1
CPU 156  → Socket 1, Core 0, Thread 1
...
CPU 207  → Socket 1, Core 51, Thread 1
```



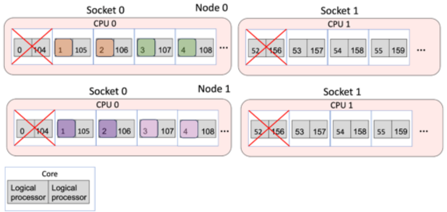

```
export CPU_BIND_SCHEME="--cpu-bind=list:1-8:9-16:17-24:25-32:33-40:41-48:53-60:61-68:69-76:77-84:85-92:93-100"
mpiexec -n 12 -ppn 12 ${CPU_BIND_SCHEME} ./app
```

MAPPING

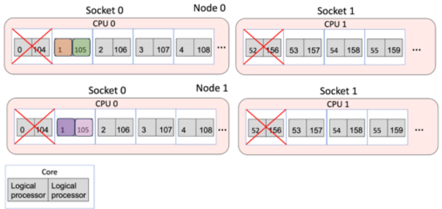
```
$ export OMP_NUM_THREADS=1
$ mpiexec -n 8 -ppn 4 --cpu-bind=list:1:2:3:4 ./app
```



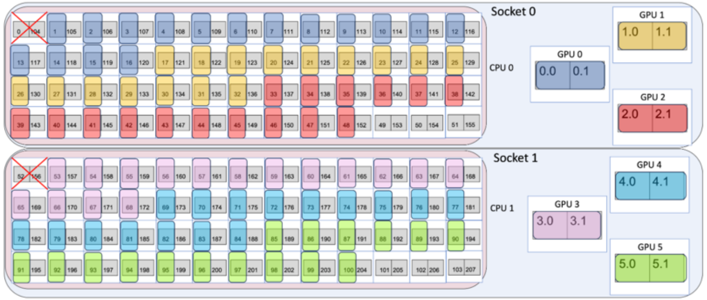

```
$ OMP_PLACES=threads OMP_NUM_THREADS=2
$ mpiexec -n 4 -ppn 2 --cpu-bind=list:1,2:3,4 ./app
```



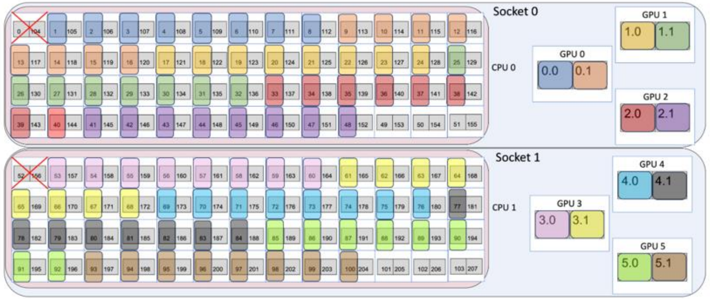

```
$ export OMP_NUM_THREADS=1
$ mpiexec -n 4 -ppn 2 --cpu-bind=list:1:105 ./app
```




```
$ export OMP_NUM_THREADS=1
$ export CPU_BIND_SCHEME="--cpu-bind=list:1-16:17-32:33-48:53-68:69-84:85-100"
$ mpiexec -n 6 -ppn 6 ${CPU_BIND_SCHEME} gpu_dev_compact.sh ./app
```




```
$ export OMP_NUM_THREADS=1
$ export CPU_BIND_SCHEME="--cpu-bind=list: 1-8:9-16:17-24:25-32:33-40:41-48:53-60:61-68:69-76:77-84:85-92:93-100"
$ mpiexec -n 12 -ppn 12 ${CPU_BIND_SCHEME} gpu_tile_compact.sh ./app
```

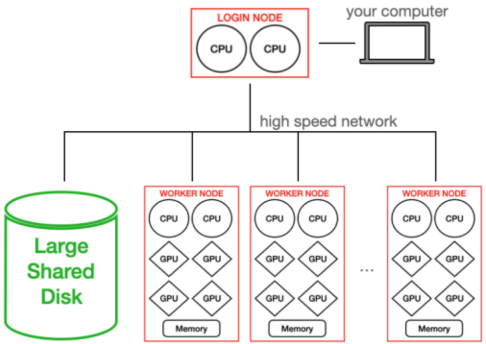


USING JOB SCHEDULER

Login: ssh <username>@aurora.alcf.anl.gov

PBS Job Script

```
#!/bin/bash -l
#PBS -l select=1
#PBS -l walltime=00:30:00
#PBS -q debug
#PBS -l filesystems=home:flare
#PBS -A alcf_training
#PBS -o logs/
#PBS -e logs/
./app_command
```



Interactive Job

```
$qsub -I -l select=1 -l walltime=00:30:00 -q debug -l filesystems=home:flare -A alcf_training
```

PLACEMENT

Aurora: **166 Racks**

Each rack: 8 chassis × 8 nodes = **64 nodes per rack**

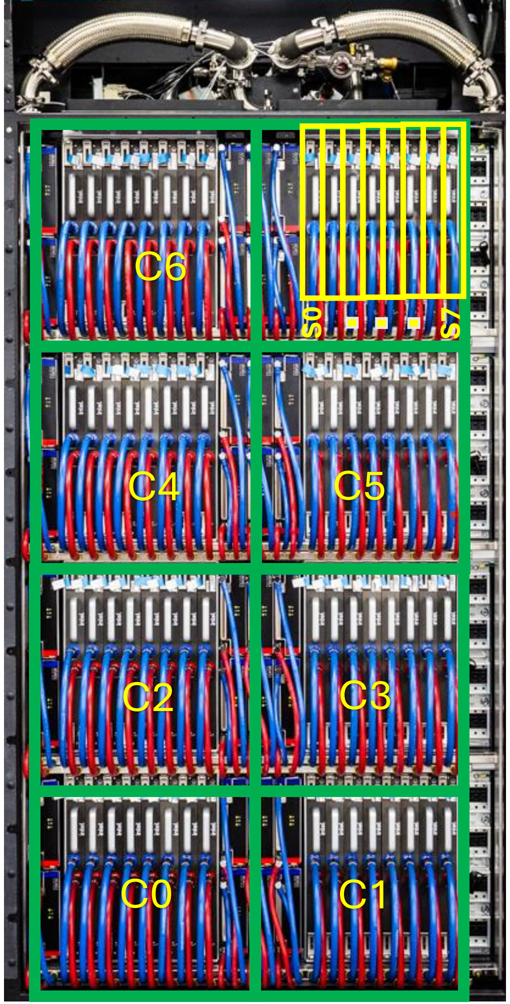
Node:x4109c0s0b0n0

Rack Identifier x4109, Chassis c0, Slot s0, Board b0, Node n0

tier0=**x4109 (with rack identifier)**

tier1=**x4109c0 (with rack and chassis identifier)**

```
#!/bin/bash -l
#PBS -l nodes:x4109c0s0b0n0
#PBS -l walltime=00:30:00
#PBS -q debug
#PBS -l filesystems=home:flare
#PBS -A <project-name>
#PBS -o logs/
#PBS -e logs/
./app_command
```



PLACEMENT

Aurora:**166 Racks**

Each rack: 8 chassis × 8 nodes = **64 nodes per rack**

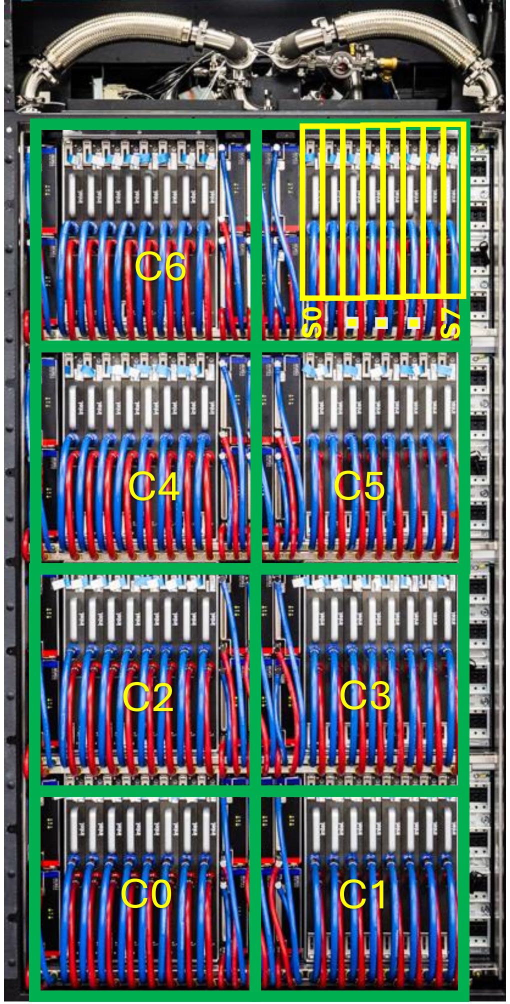
Node:x4109c0s0b0n0

Rack Identifier x4109, Chassis c0, Slot s0, Board b0, Node n0

tier0=**x4109 (with rack identifier)**

tier1=**x4109c0 (with rack and chassis identifier)**

```
#!/bin/bash -l
#PBS -l select=32
#PBS -l place=scatter:group=tier0
#PBS -l walltime=00:30:00
#PBS -q debug
#PBS -l filesystems=home:flare
#PBS -A <project-name>
#PBS -o logs/
#PBS -e logs/
./app_command
```



PLACEMENT

Aurora:**166 Racks**

Each rack: 8 chassis × 8 nodes = **64 nodes per rack**

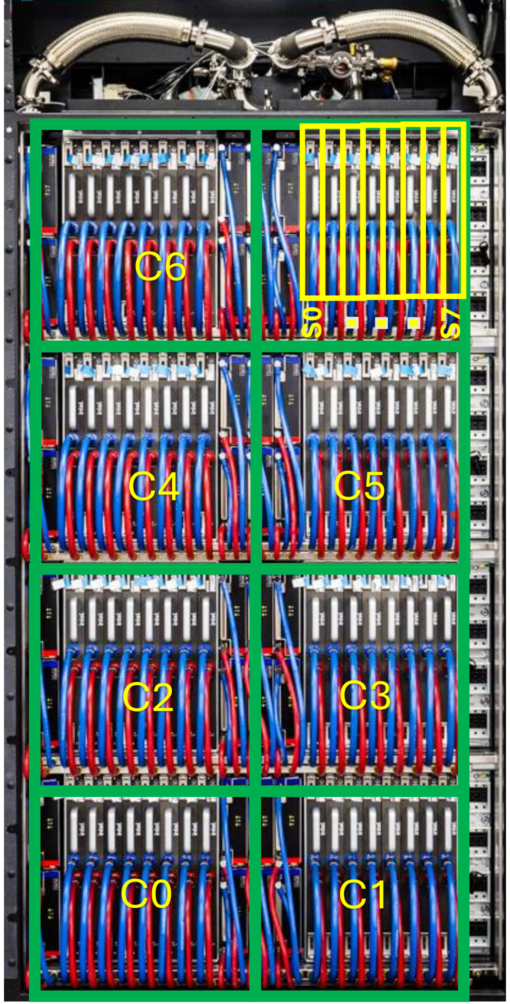
Node:x4109c0s0b0n0

Rack Identifier x4109, Chassis c0, Slot s0, Board b0, Node n0

tier0=**x4109 (with rack identifier)**

tier1=**x4109c0 (with rack and chassis identifier)**

```
#!/bin/bash -l
#PBS -l select=32:tier0=x4109
#PBS -l place=scatter:group=tier0
#PBS -l walltime=00:30:00
#PBS -q debug
#PBS -l filesystems=home:flare
#PBS -A <project-name>
#PBS -o logs/
#PBS -e logs/
./app_command
```



PLACEMENT

Aurora:**166 Racks**

Each rack: 8 chassis × 8 nodes = **64 nodes per rack**

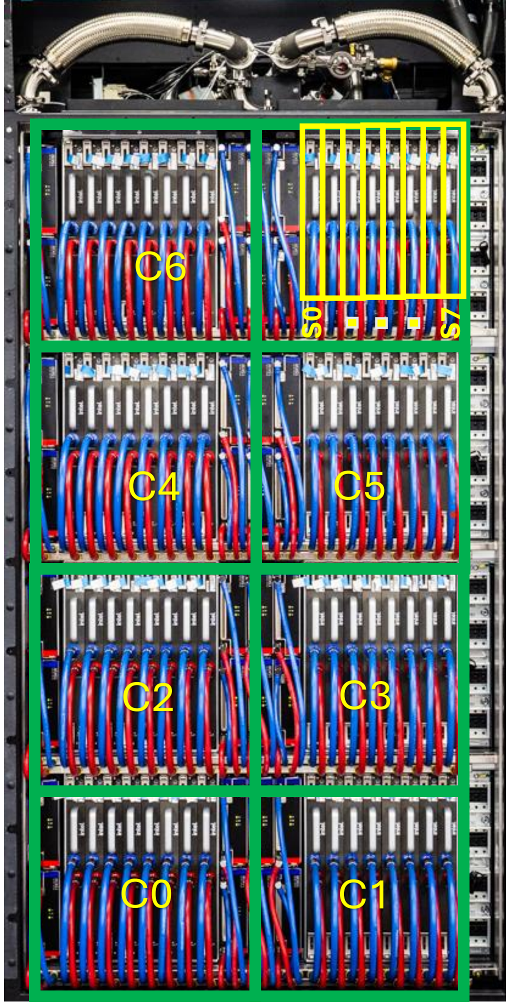
Node:x4109c0s0b0n0

Rack Identifier x4109, Chassis c0, Slot s0, Board b0, Node n0

tier0=**x4109 (with rack identifier)**

tier1=**x4109c0 (with rack and chassis identifier)**

```
#!/bin/bash -l
#PBS -l select=8
#PBS -l place=scatter:group=tier1
#PBS -l walltime=00:30:00
#PBS -q debug
#PBS -l filesystems=home:flare
#PBS -A <project-name>
#PBS -o logs/
#PBS -e logs/
./app_command.
```



PLACEMENT

Aurora:**166 Racks**

Each rack: 8 chassis × 8 nodes = **64 nodes per rack**

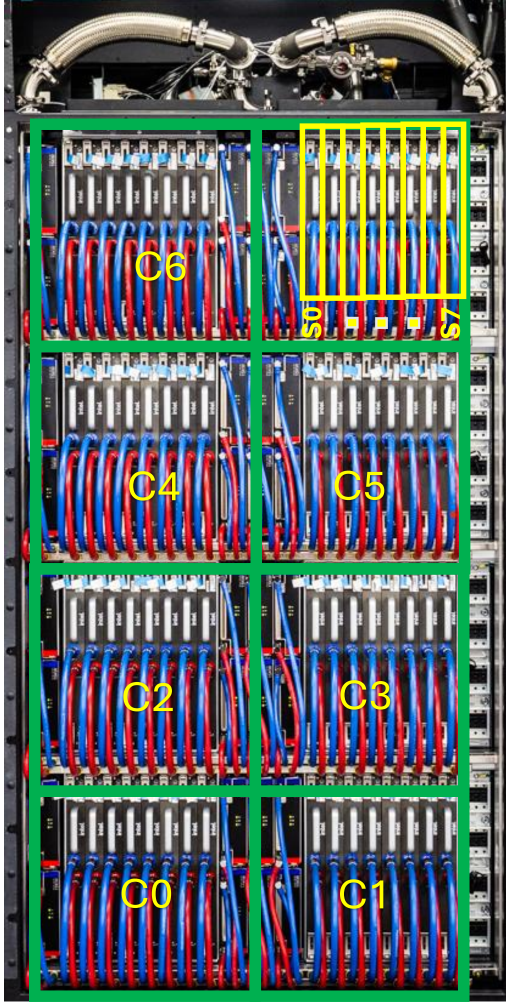
Node:x4109c0s0b0n0

Rack Identifier x4109, Chassis c0, Slot s0, Board b0, Node n0

tier0=**x4109 (with rack identifier)**

tier1=**x4109c0 (with rack and chassis identifier)**

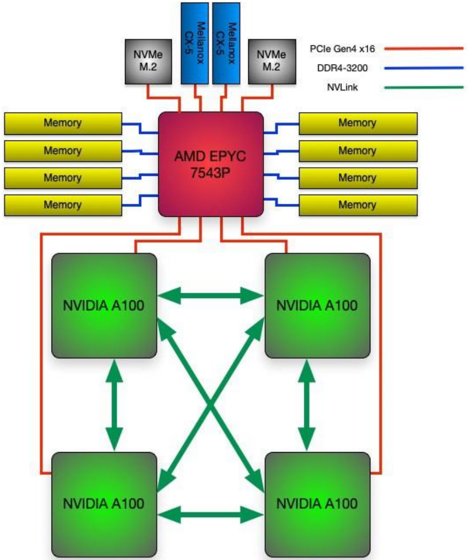
```
#!/bin/bash -l
#PBS -l select=8:tier1= x4109c0
#PBS -l place=scatter:group=tier1
#PBS -l walltime=00:30:00
#PBS -q debug
#PBS -l filesystems=home:flare
#PBS -A <project-name>
#PBS -o logs/
#PBS -e logs/
./app_command.
```



POLARIS

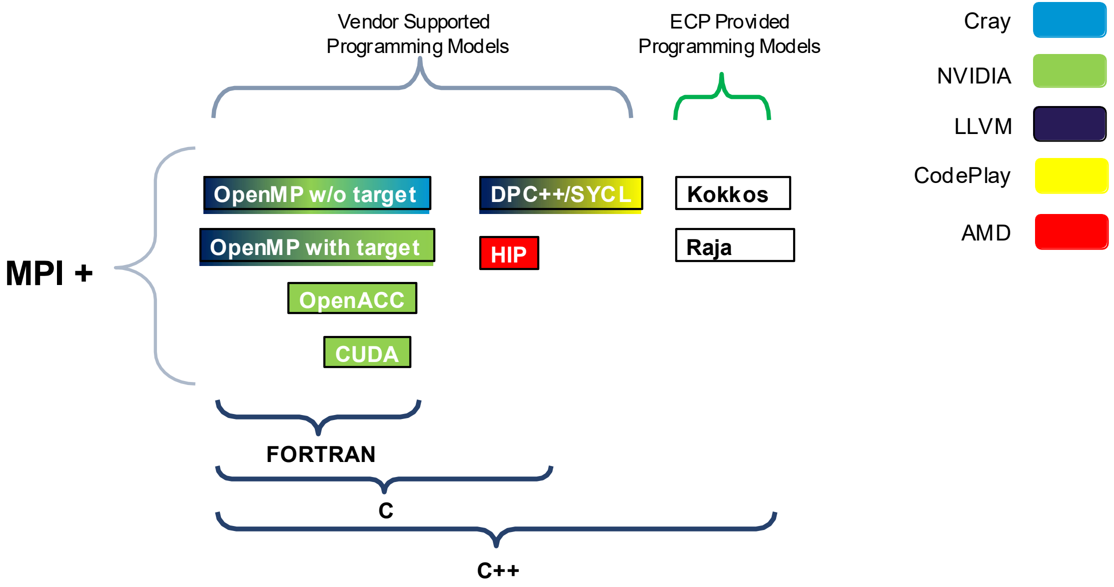
POLARIS NODE AND SYSTEM CONFIGURATION

# 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 Cassini NICs	2
Total Injection BW (w/ Cassini)	50 GB/s
PCIe Gen4 BW	64 GB/s
NVLink BW	600 GB/s
Total GPU DP Tensor Core Flops	78 TF



# 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
Total GPU DP Tensor Core Flops	44 PF
Total Power	1.8 MW

PROGRAMMING MODELS ON POLARIS



PROGRAMMING MODELS ON AURORA

- Aurora applications may use:
  - DPC++/SYCL
  - OpenMP
  - Kokkos
  - Raja
  - OpenCL
- Experimental
  - HIP – *running GAMESS, CP2K, libCEED*
- Not available on Aurora:
  - CUDA
  - OpenACC

TO HANDS-ON …

ACKNOWLEDGMENT

- This research used resources of the Argonne Leadership Computing Facility, a U.S. Department of Energy (DOE) Office of Science user facility at Argonne National Laboratory and is based on research supported by the U.S. DOE Office of Science-Advanced Scientific Computing Research Program, under Contract No. DE-AC02-06CH11357.
- We gratefully acknowledge the computing resources provided and operated by the Joint Laboratory for System Evaluation (JLSE) at Argonne National Laboratory.
- This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

Argonne
NATIONAL LABORATORY

U.S. DEPARTMENT OF
ENERGY