October 10-12, 2023

ALCF Hands-on HPC Workshop

From Polaris to Aurora

Overview of Hardware and Software

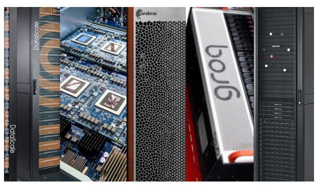
Brian Homerding
Performance Engineer
Argonne Leadership Computing Facility (ALCF)

October 10th, 2023

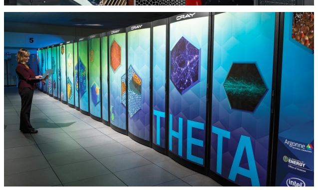
ALCF Systems

- ❑ Aurora (CPU+GPU)
  - ❑ Theoretical peak performance: > 2 Exaflops DP
  - ❑ > 10,000 nodes: 2x 4th Gen Intel XEON Max Series + 6x Data Center GPU Max Series

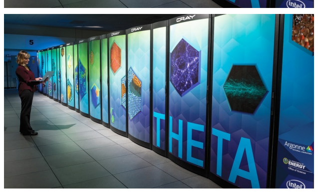
- ❑ Polaris (CPU+GPU)
  - ❑ Top500: Rmax 25.82 PFlop/s, Rpeak 34.16 PFlop/s
  - ❑ 560 nodes: 1x AMD EPYC Milan 7543P + 4x NVIDIA A100



- ❑ ALCF AI Testbed (various AI accelerators)
  - ❑ Available for allocation requests (DD):
    Cerebras CS-2, SambaNova DataScale, Graphcore Bow Pod64
  - ❑ Access Forthcoming: Groq, Habana Gaudi



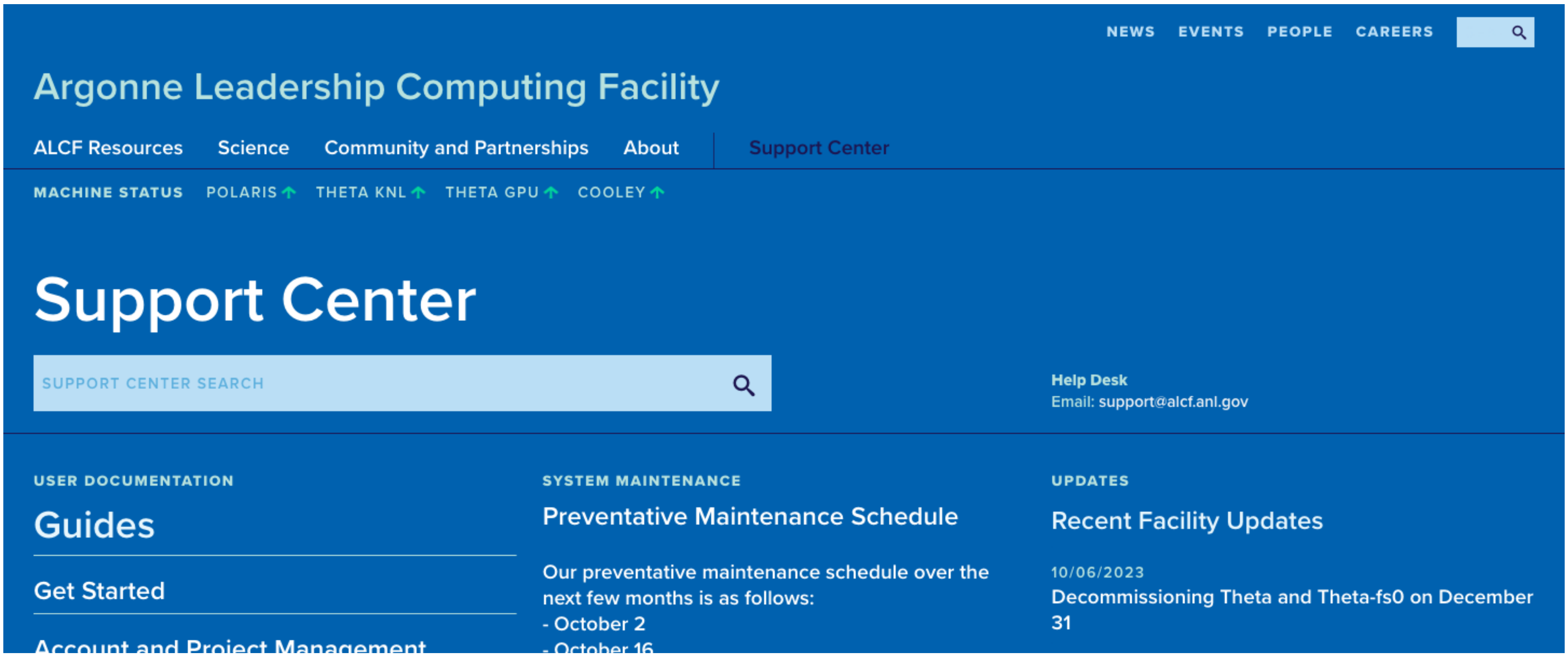
- ❑ ThetaGPU (CPU+GPU)
  - ❑ GPU-accelerated computing pathfinder, Rpeak 3.9 PFlop/s
  - ❑ 42 nodes: 2x AMD EPYC Rome 7742 + 8x NVIDIA A100



- ❑ Theta (CPU)
  - ❑ Top500: Rmax 6.92 PFlop/s, Rpeak 11.66 PFlop/s
  - ❑ 4392 nodes: 1x Intel Xeon Phi 7230 (KNL)

Getting Started on ALCF Systems

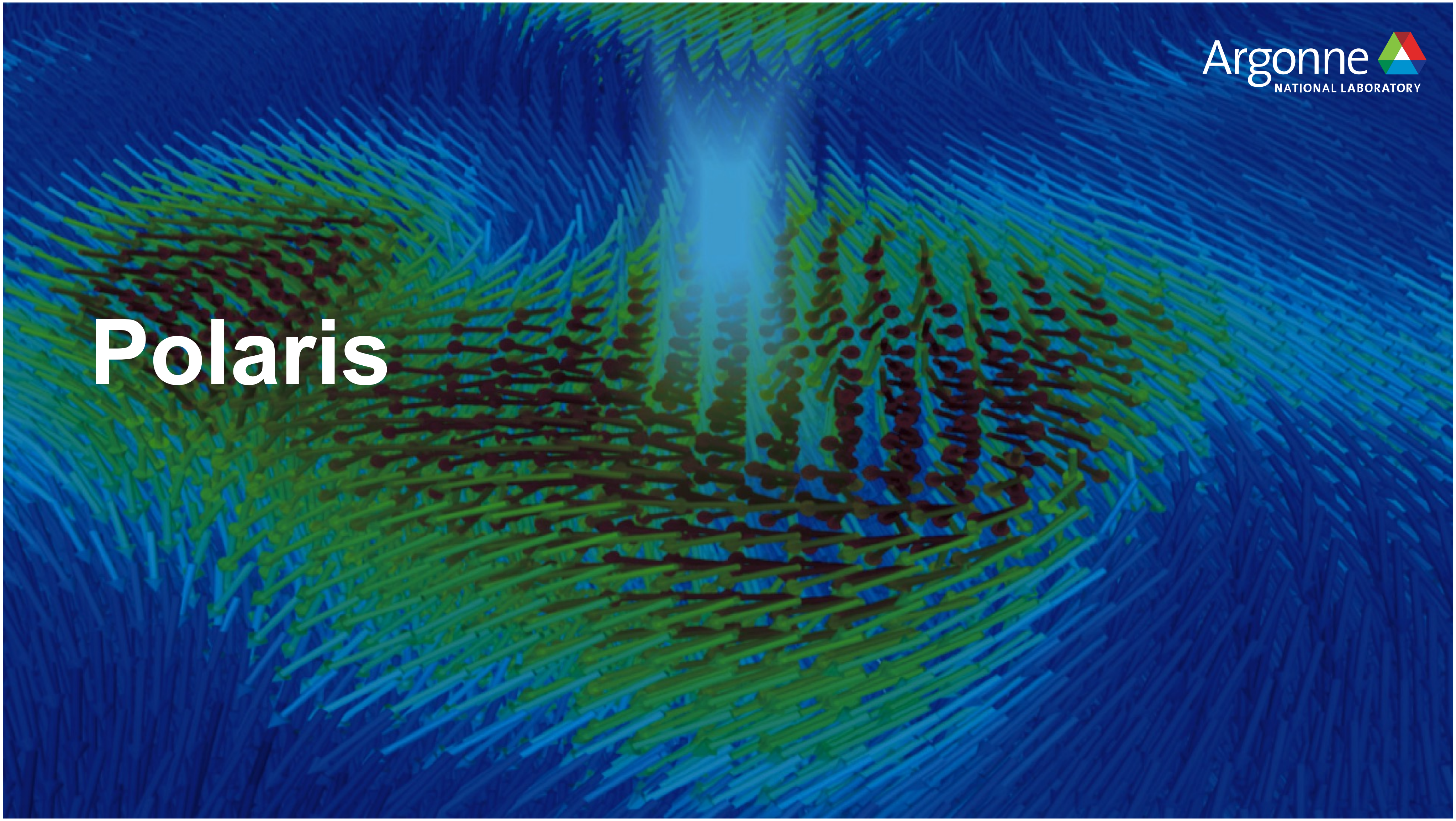
❑ ALCF guides and information: https://www.alcf.anl.gov/support-center



Getting Started at the ALCF Hands-on HPC Workshop

- Connect and login:
  - `ssh <your_ALCF_username>@<ALCF_system_name>.alcf.anl.gov`
- Workshop materials:
  - https://github.com/argonne-lcf/ALCF_Hands_on_HPC_Workshop
- Slack: # announcements, # q-and-a, # *-breakout
- Workshop project information
  - Project name: `fallwkshp23`
  - Available queues:
    - Single node: `fallws23single`
    - Scaling up to 128 nodes: `fallws23scaling`
  - Project storage location: `/lus/eagle/projects/fallwkshp23/`

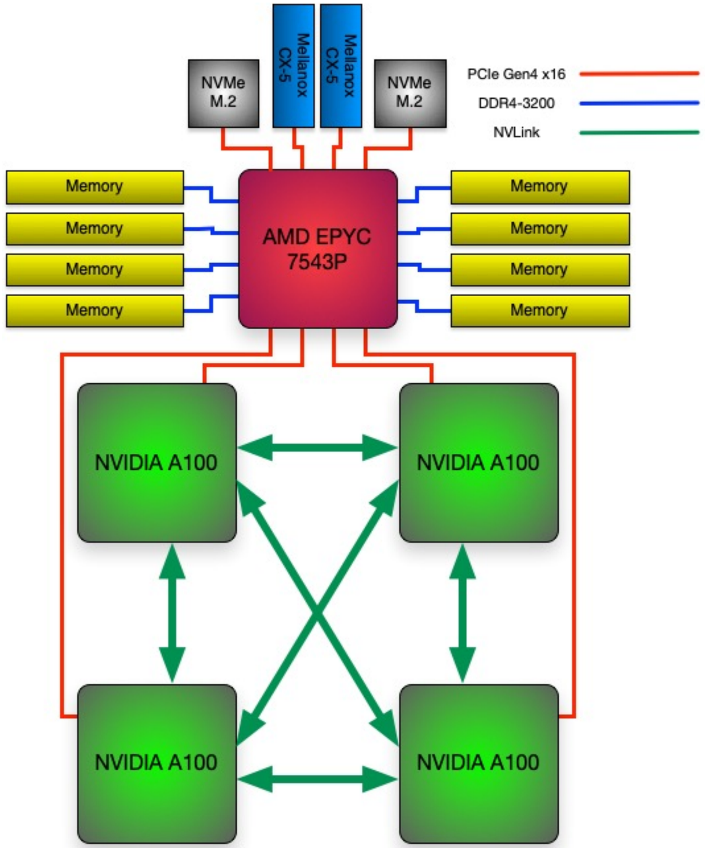
Polaris



Hardware

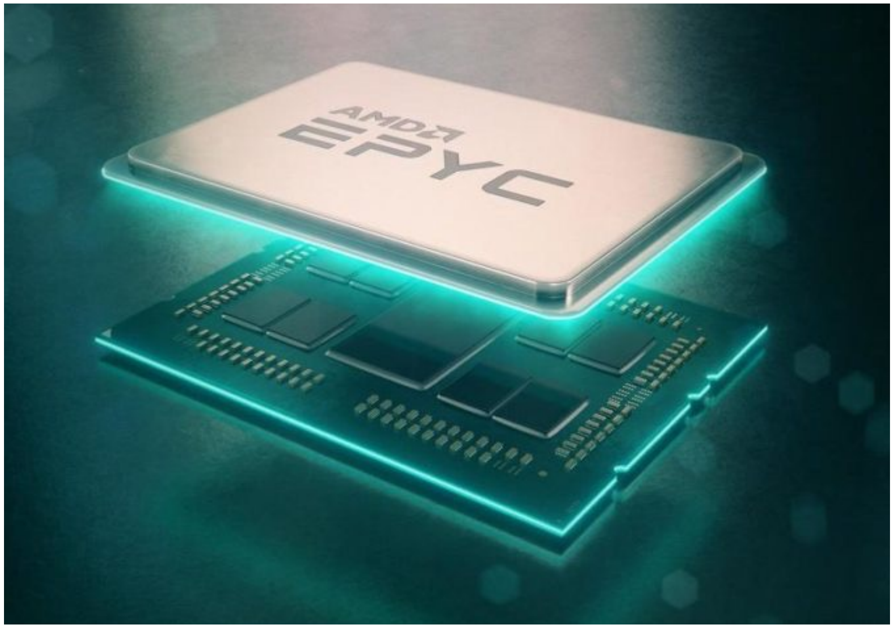
Polaris Single Node Configuration

# 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 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



Single AMD EPYC “MILAN” 7543P CPU Specs

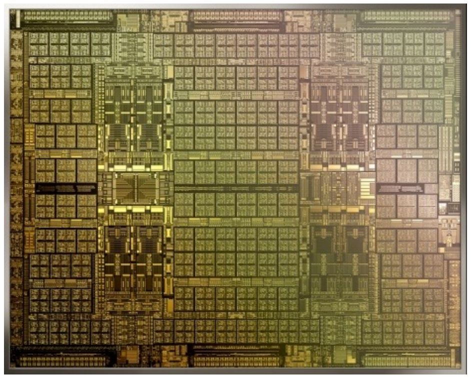
Base Frequency	2.8 GHz
Max Boost Clk	3.7 GHz
# of Zen3 Cores	32
# of Threads	64
Total DDR4 Memory	512 GB
# of Memory Channels	8
DDR4 Memory BW	204.8 GB/s
Total Shared L3 Cache	256 MB
L2 Cache per Core	512 KB
L1 Cache per Core	32 KB
PCIe Gen 4	128 lanes (8 ports)
PCIe Gen4 BW	64 GB/s
TDP	225 W



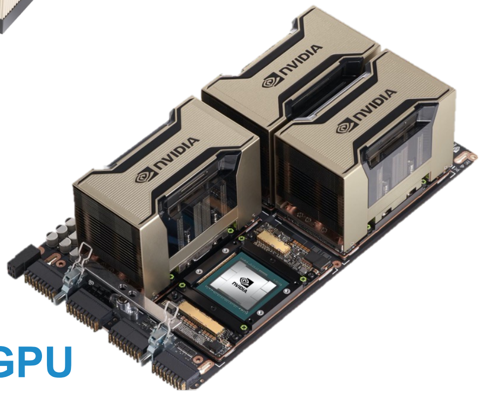
NVIDIA HGX A100 Specs

	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

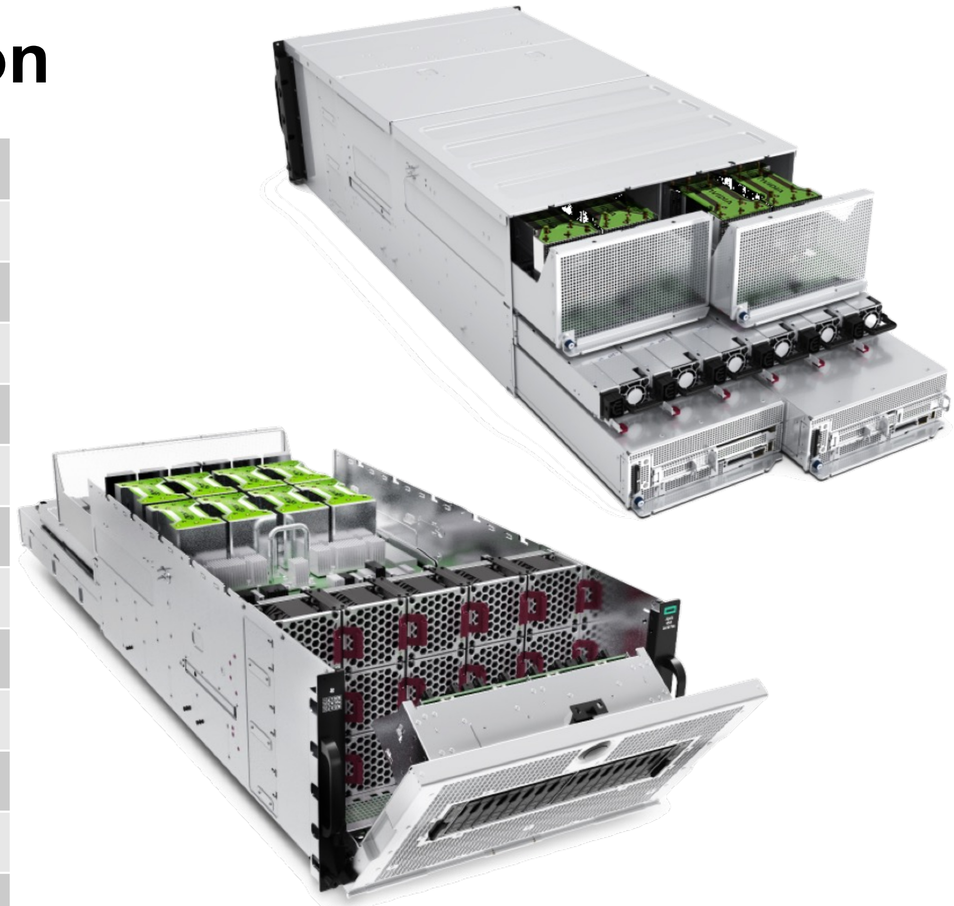
Node Local Storage

- 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
- Each volume will be mounted as an ext4/xfs volume that is user accessible
- Users access SSD via standard POSIX APIs
- Data is destroyed when the job ends so any data users wish to keep must be moved to Grand or Eagle

Argonne NATIONAL LABORATORY

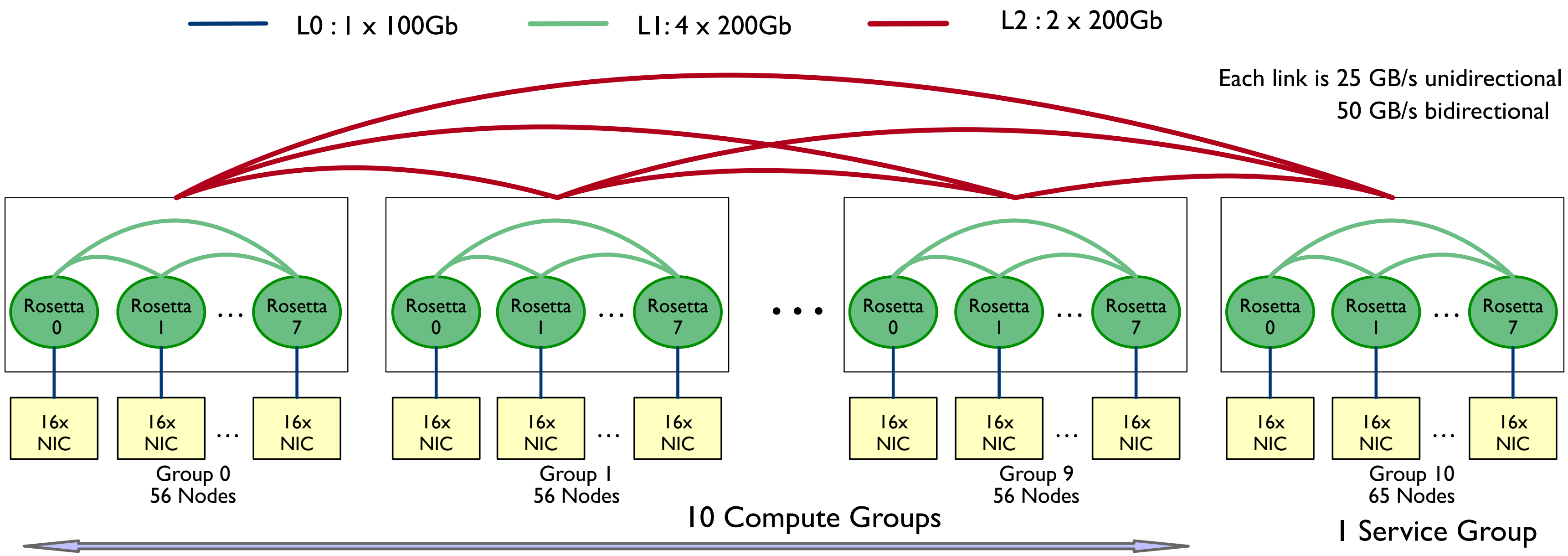
Polaris System Configuration

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



Apollo 6500 Gen10+

Slingshot Configuration

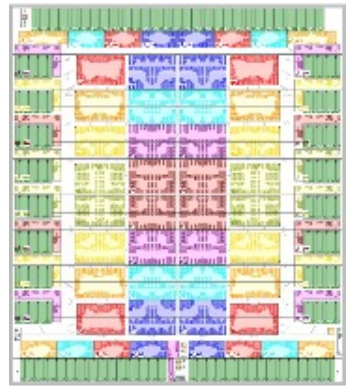


- 11 Total dragonfly groups, 10 compute groups and 1 non-compute group
- 2 links/arc between each group
- 4 links/arc within each group (between switches of a group)
- 1 link from each NIC (100Gb with SS10, 200Gb when upgraded to SS11)

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

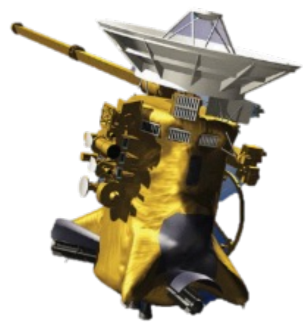
SS-10 (100Gb)
Injection: ~14 TB/s
Bisection: ~24 TB/s

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

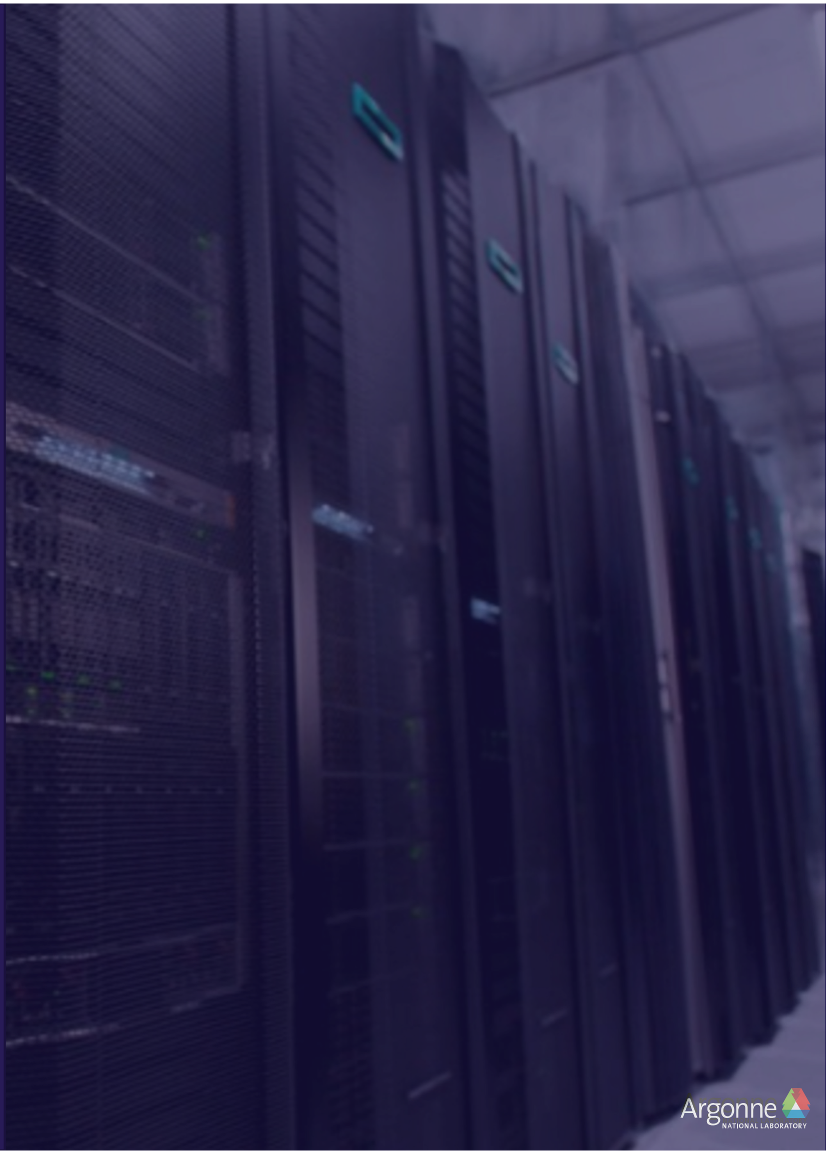
Slingshot 11 upgrade

- ❑ Polaris's interconnect is being upgraded from Slingshot 10 to Slingshot 11
- ❑ Phased Rollout
  - ❑ October 16: 30% of system
  - ❑ October 30: 60% of system
  - ❑ November 13: complete system
- ❑ Will impact maximum job size
- ❑ https://www.alcf.anl.gov/support-center/facility-updates/polaris-upgrade-slingshot-10-slingshot-11

Storage

Polaris is 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

Filesystem

- Polaris has a shared home filesystem

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

- Main project storage
  - `/lus/grand/projects`

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

Programming Environment

- HPE Cray PE for Polaris
  - HPE Cray MPI support for PGI offload to A100 for Multi-NIC and Multi-GPU support
  - Full Rome and Milan support

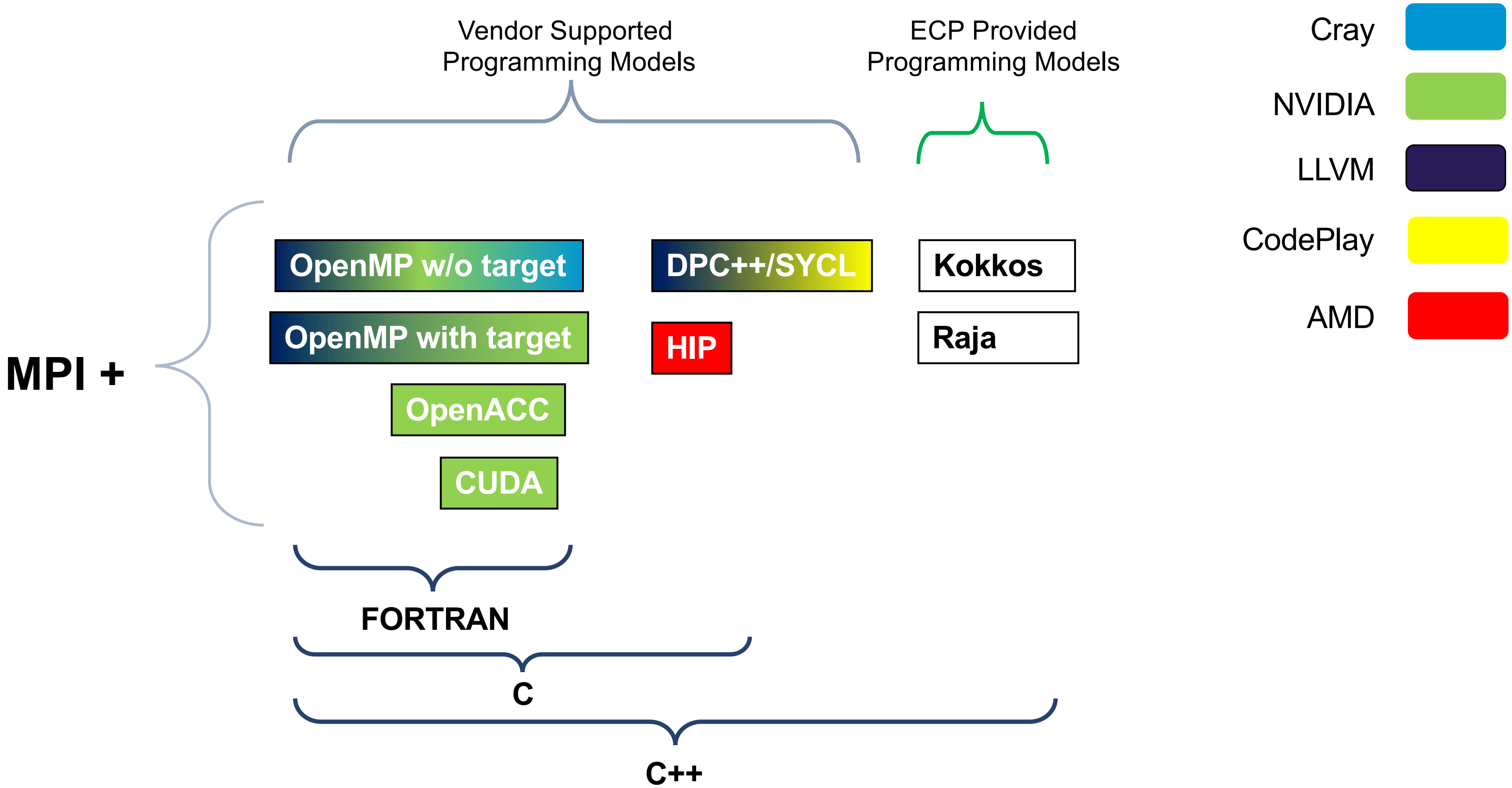
- NVIDIA HPC SDK will provide primary support for programming A100

- SYCL/Data Parallel C++ provided via
  - CodePlay computecpp compiler with Nvidia support
  - LLVM via Intel DPC++ branch which supports offload to Nvidia GPUs as well as Intel GPUs

Modules

```
alcf@polaris-login-04:~> module list
Currently Loaded Modules:
  1) craype-x86-rome          5) nvhpc/21.9          9) cray-pmi/6.1.2       13) PrgEnv-nvhpc/8.3.3
  2) libfabric/1.11.0.4.125   6) craype/2.7.15      10) cray-pmi-lib/6.0.17  14) craype-accel-nvidia80
  3) craype-network-ofi       7) cray-dsmml/0.2.2   11) cray-pals/1.1.7
  4) perftools-base/22.05.0   8) cray-mpich/8.1.16  12) cray-libpals/1.1.7



alcf@polaris-login-04:~> module avail
-------------------- /opt/cray/pe/lmod/modulefiles/mpi/nvidia/20/ofi/1.0/cray-mpich/8.0 --------------------
   cray-hdf5-parallel/1.12.1.3    cray-parallel-netcdf/1.12.2.3
-------------------------- /opt/cray/pe/lmod/modulefiles/comnet/nvidia/20/ofi/1.0 --------------------------
   cray-mpich-abi/8.1.16    cray-mpich/8.1.16 (L)
------------------------------ /opt/cray/pe/lmod/modulefiles/compiler/nvidia/20 -----------------------------
   cray-hdf5/1.12.1.3
-------------------------------- /opt/cray/pe/lmod/modulefiles/mix_compilers --------------------------------
   gcc-mixed/11.2.0        nvhpc-mixed/23.1    nvidia-mixed/21.9 (D)    nvidia-mixed/23.3
   nvhpc-mixed/21.9 (D)    nvhpc-mixed/23.3    nvidia-mixed/23.1
------------------------------ /opt/cray/pe/lmod/modulefiles/perftools/22.05.0 ------------------------------
   perftools         perftools-lite-events    perftools-lite-hbm      perftools-preload
{...}
```

Programming Models



Compilers

- Cray Programming Environment provides wrappers for building MPI enabled application
  - `cc` – C compiler
  - `CC` – C++ compiler
  - `ftn` – Fortran compiler

- The wrappers provide options to understand the underlying invocation
  - `--craype-verbose` – prints the underlying compiler invocation
  - `--cray-print-opts=libs` – prints library information
  - `--cray-print-opts=cflags` – prints include information

- The opts prints are useful for build scripts
  - `CRAY_LIB=$(cc --cray-print-opts=libs)`
  - `CRAY_CFLAGS=$(cc --cray-print-opts=cflags)`

Compilers

- Beyond the default `PrgEnv-nvhpc` environment. Several additional compilers are available with varying support for programming models.
  - GNU:
    - GNU compilers. Useful for mixing with nvhpc compilers
  - LLVM:
    - Open source LLVM compiler. Support for CUDA and OpenMP offload
  - Cray:
    - Cray Compiling Environment (CCE)
  - oneAPI Toolkit:
    - Intel oneAPI compiler and Codeplay plugins for NVIDIA GPUs

Scheduler – PBS Professional

- Primary commands
  - `qsub`
    - Request resources and start your script on the head node
    - `-A` - Allocation
    - `-l` - Options
  - `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 unneeded requests

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.

Polaris Queues, Projects, and Allocations

- There are several production queues for submitting jobs to Polaris
  - `debug`, `prod`, …
  - Workshop reservation available queues:
    - Single node: `fallws23single`
    - Scaling up to 128 nodes: `fallws23scaling`
- Projects have an approved amount of disk space.
    - `alcf@polaris-login-04:~> myprojectquotas`

```
Name                         Type     Filesystem        Used          Quota          Grace
==========================================================================================
fallwkshp23               Project     grand               4k             1T              -
```

  - Workshop project storage location: `/lus/eagle/projects/fallwkshp23/`
- Node hour allocations on approved systems.
  - `alcf@polaris-login-04:~> sbank`

```
Allocation  Suballocation  Start       End         Resource  Project       Jobs  Charged  Available Balance
----------  -------------  ----------  ----------  --------  ------------  ----  -------  -----------------
10953       10821          2023-08-08  2023-11-09  polaris   fallwkshp23     12      1.3            2,998.7
```


Argonne NATIONAL LABORATORY

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

MPI Environment Variables

- `MPICH_GPU_SUPPORT_ENABLED`
  — Enable MPI operations with communication buffers on GPU-attached memory regions
- `MPICH_OFI_NIC_VERBOSE`
  — Print verbose information about NIC selection
- `MPICH_OFI_NIC_POLICY`
  — Selects the rank-to-NIC assignment policy (BLOCK, ROUND-ROBIN, NUMA, GPU, USER)
- `MPICH_OFI_NIC_MAPPING`
  — Specifies the rank-to-NIC mapping on each node

Affinity Example – Submission Script

- https://github.com/argonne-lcf/GettingStarted/tree/master/Examples/Polaris/affinity

```
#!/bin/sh
#PBS -l select=1:system=polaris
#PBS -l place=scatter
#PBS -l walltime=0:30:00
#PBS -q debug
#PBS -A <PROJECT>
#PBS -l filesystems=home:grand:eagle

cd ${PBS_O_WORKDIR}
# MPI example w/ 16 MPI ranks per node spread evenly across cores
NNODES=`wc -l < $PBS_NODEFILE`
NRANKS_PER_NODE=16
NDEPTH=4
NTHREADS=1
NTOTRANKS=$(( NNODES * NRANKS_PER_NODE ))
echo "NUM_OF_NODES= ${NNODES} TOTAL_NUM_RANKS= ${NTOTRANKS}
   RANKS_PER_NODE= ${NRANKS_PER_NODE} THREADS_PER_RANK= ${NTHREADS}"

mpiexec -n ${NTOTRANKS} --ppn ${NRANKS_PER_NODE}
   --depth=4 --cpu-bind depth ./hello_affinity
```

Affinity Example – Output

- https://github.com/argonne-lcf/GettingStarted/tree/master/Examples/Polaris/affinity

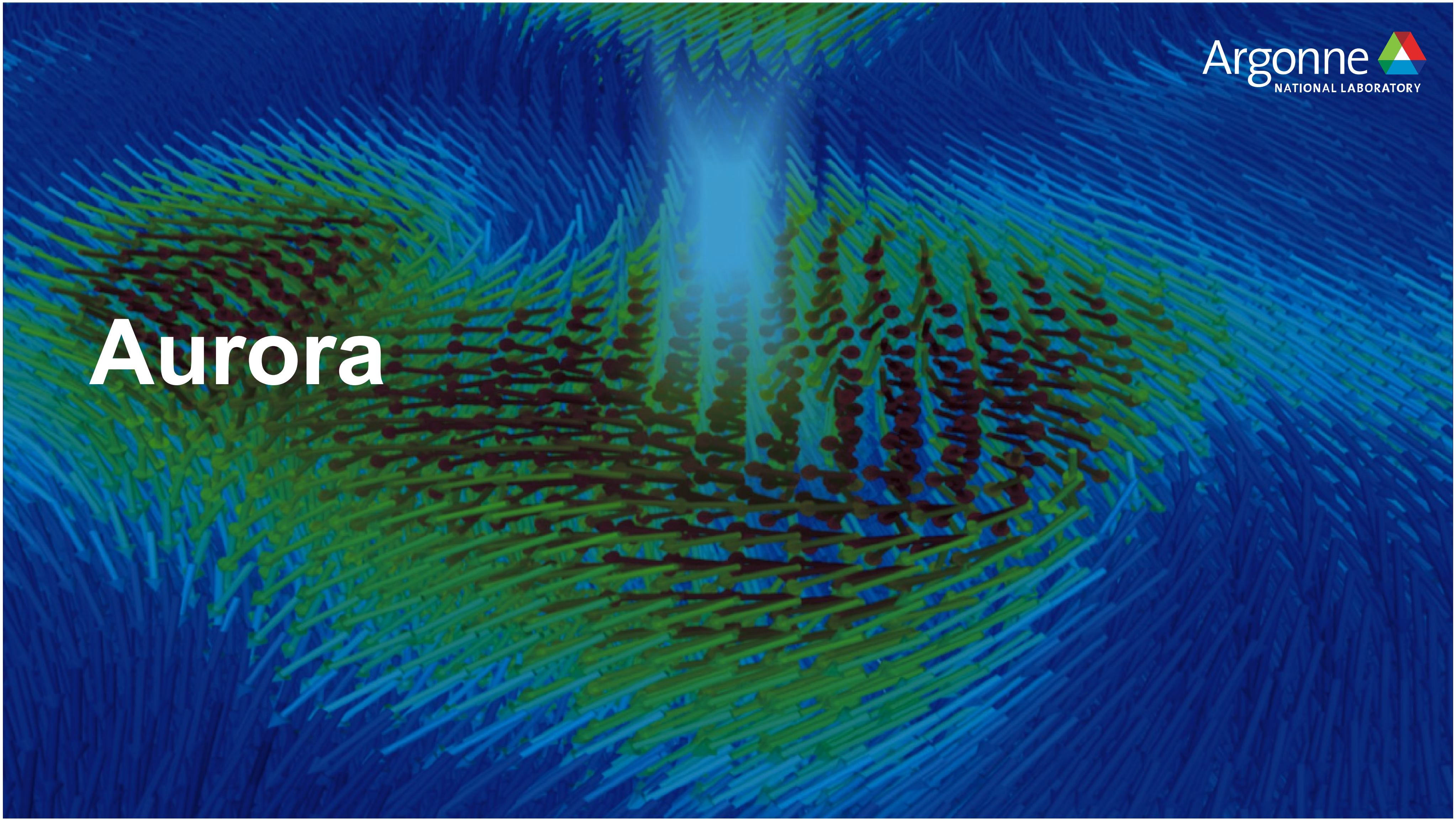
```
$ qsub -l select=2,walltime=0:10:00 -l filesystems=home:grand:eagle
       -A <PROJECT> ./submit.sh


NUM_OF_NODES= 2 TOTAL_NUM_RANKS= 32 RANKS_PER_NODE= 16 THREADS_PER_RANK= 1

To affinity and beyond!! nname= x3007c0s13b0n0 rnk= 0 list_cores= (0-3)
To affinity and beyond!! nname= x3007c0s13b0n0 rnk= 1 list_cores= (4-7)
...
To affinity and beyond!! nname= x3007c0s13b0n0 rnk= 15 list_cores= (60-63)
To affinity and beyond!! nname= x3007c0s13b1n0 rnk= 16 list_cores= (0-3)
...
To affinity and beyond!! nname= x3007c0s13b1n0 rnk= 31 list_cores= (60-63)
```

Polaris Debuggers

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

Polaris Profilers

- ❑Profilers
  - ❑ PAT (Performance Analysis Tool)
    - ❑Whole program performance analysis
  - ❑ NVIDIA® Nsight™
    - ❑System-wide performance analysis tool
  - ❑ TAU (Tuning and Analysis Utilities)
    - ❑Portable profiling and tracing toolkit
  - ❑ THAPI (Tracing Heterogeneous APIs)
    - ❑Tracing infrastructure for heterogeneous computing applications
  - ❑ HPCToolkit
    - ❑Integrated suite of tools for measurement and analysis of program performance



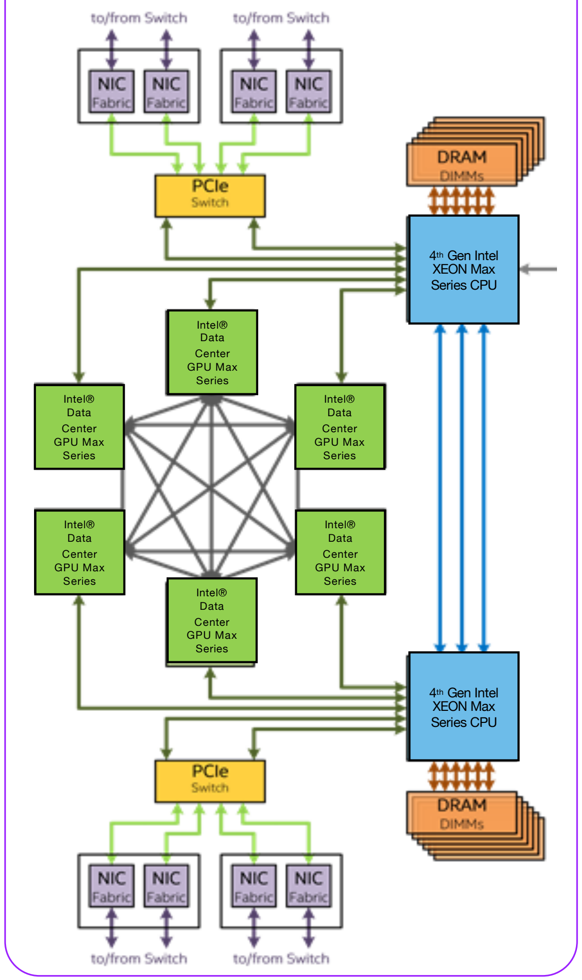
Aurora

Hardware


Argonne
NATIONAL LABORATORY
U.S. DEPARTMENT OF
ENERGY
intel
Hewlett Packard
Enterprise

Aurora Compute Node

- Six Intel® Data Center GPU Max Series
  - All to all connection
- Two 4th Gen Intel XEON Max Series CPUs with:
  - HBM memory
  - DDR memory
- Unified Memory Architecture across CPUs and GPUs
- 8 Slingshot Fabric endpoints



Polaris to Aurora – Compute Node Hardware

- CPUs:
  - 1x AMD EPYC 7543P CPU -> 2x 4th Gen Intel XEON Max Series CPUs
- GPUs:
  - 4x NVIDIA A100 GPUs -> 6x Intel® Data Center GPU Max Series
- Slingshot fabric endpoints:
  - 2x NICs -> 8x NICs

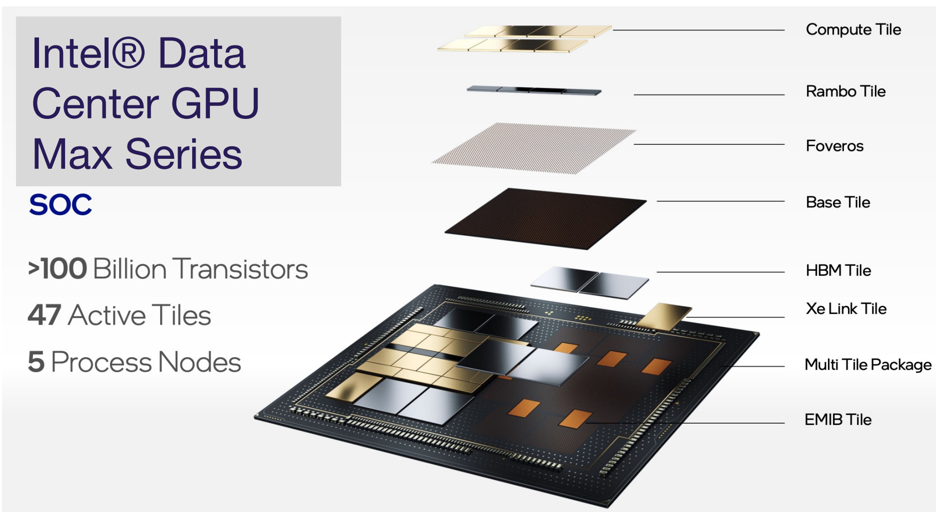
Intel® Data Center GPU Max Series

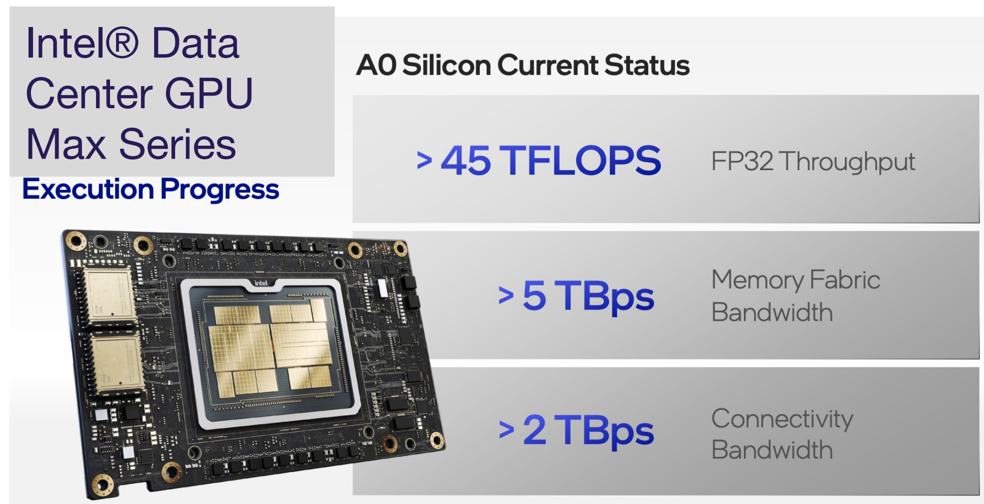
Intel provided an introduction to the Intel® Data Center GPU Max Series at an Intel Architecture Day event

- https://www.intel.com/content/www/us/en/newsroom/resources/press-kit-architecture-day-2021.html

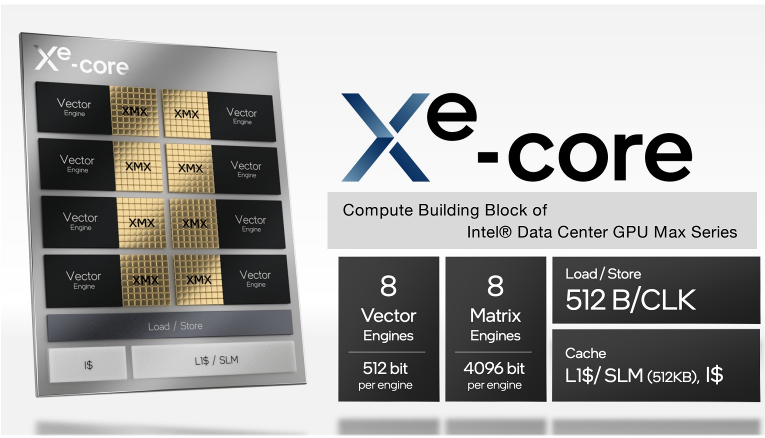
Also presented at Hot Chips

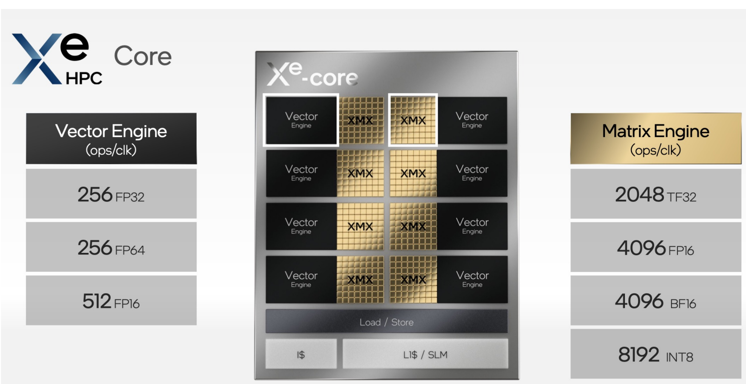
- https://hc33.hotchips.org/assets/program/conference/day2/hc2021_pvc_final.pdf

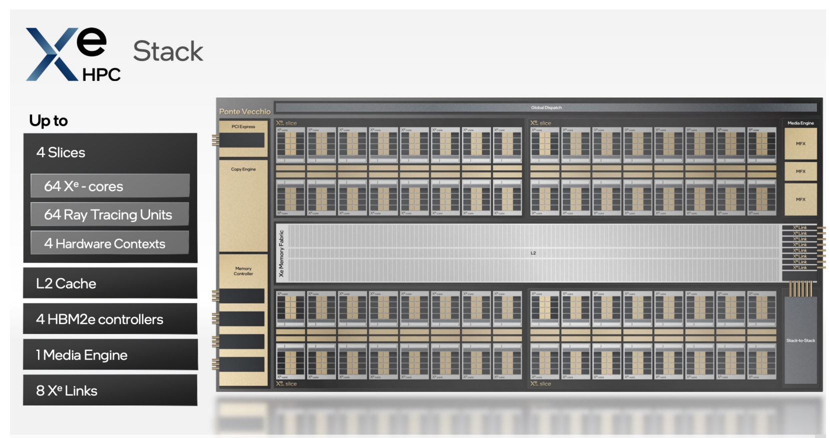




Intel® Data Center GPU Max Series Architectural Components







Argonne NATIONAL LABORATORY

Aurora

Leadership Computing Facility
Exascale Supercomputer

Peak Performance
≧ 2 Exaflops DP

Intel GPU
Intel® Data Center GPU Max Series

Intel Xeon Processor
4th Gen Intel XEON Max Series CPU with High Bandwidth Memory

Platform
HPE Cray-Ex

Compute Node
Two 4th Gen Intel XEON Max Series CPUs
Six Intel® Data Center GPU Max Series
Node Unified Memory Architecture
Eight fabric endpoints

GPU Architecture
Intel® Data Center GPU Max Series architecture
High Bandwidth Memory Stacks

Node Performance
>130 TF

System Size
>10,000 nodes

Aggregate System Memory
>10 PB aggregate System Memory

System Interconnect
HPE Slingshot 11
Dragonfly topology with adaptive routing

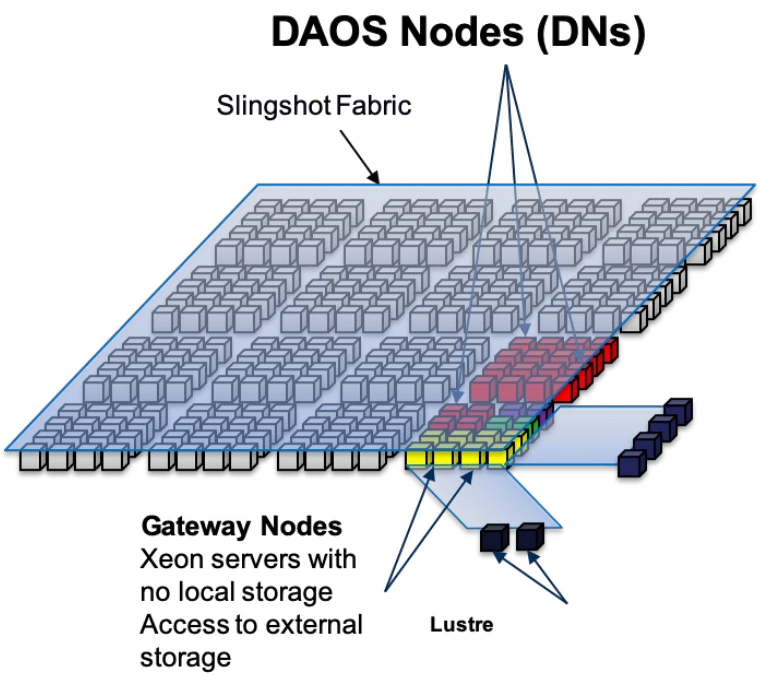
Network Switch
25.6 Tb/s per switch (64 200 Gb/s ports)
Links with 25 GB/s per direction

High-Performance Storage
220 PB
≧25 TB/s DAOS bandwidth

Software Environment
- C/C++
- Fortran
- SYCL/DPC++
- OpenMP offload
- Kokkos
- RAJA
- Intel Performance Tools

Distributed Asynchronous Object Store (DAOS)

- Primary storage system for Aurora
- Offers high performance in bandwidth and IO operations
  - 230 PB capacity
  - ≥ 25 TB/s
- Provides a flexible storage API that enables new I/O paradigms
- Provides compatibility with existing I/O models such as POSIX, MPI-IO and HDF5
- Open source storage solution



Software

Available Aurora Programming Models

- Aurora applications may use:
  - DPC++/SYCL
  - OpenMP
  - Kokkos
  - Raja
  - OpenCL

- Experimental
  - HIP – *running GAMESS, CP2K, libCEED*

- Not available on Aurora:
  - CUDA
  - OpenACC

Polaris to Aurora – Programming Models

- OpenACC -> OpenMP
  - Both pragma based directive programming models

- CUDA -> DPC++/SYCL, Kokkos, RAJA, OpenMP
  - Intel DPC++ Compatibility Tool: assisted migration of CUDA to DPC++
  - https://www.intel.com/content/www/us/en/docs/dpcpp-compatibility-tool/get-started-guide/2023-0/overview.html

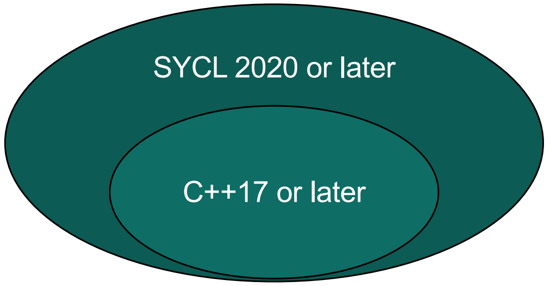
oneAPI

- Industry specification from Intel (https://www.oneapi.com/spec/)
  - Language and libraries to target programming across diverse architectures (DPC++, APIs, low level interface)
- Intel oneAPI products and toolkits (https://software.intel.com/ONEAPI)
  - Languages
    - Fortran (w/ OpenMP 5+)
    - C/C++ (w/ OpenMP 5+)
    - DPC++
    - Python
  - Libraries
    - oneAPI MKL (oneMKL)
    - oneAPI Deep Neural Network Library (oneDNN)
    - oneAPI Data Analytics Library (oneDAL)
    - MPI
  - Tools
    - Intel Advisor
    - Intel VTune
    - Intel Inspector

https://software.intel.com/oneapi

DPC++ (Data Parallel C++) and SYCL

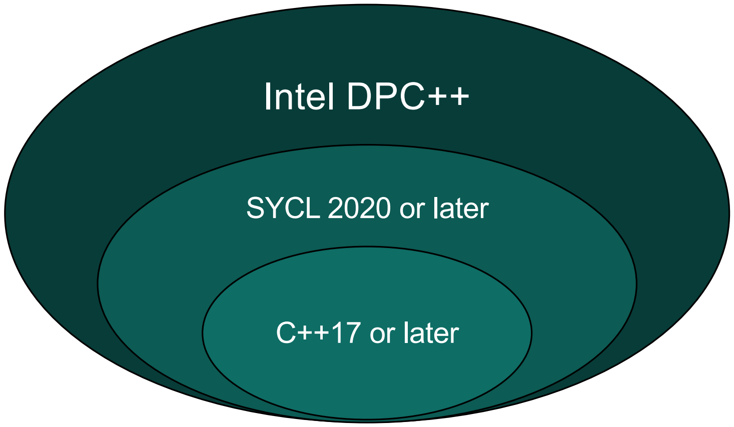
- SYCL
  - Standard developed by Khronos and announced in 2014
  - The latest SYCL specification (SYCL 2020) was released in 2021
  - SYCL is a C++ based abstraction layer (standard C++17)
  - Builds on OpenCL **concepts** (but single-source)
  - *SYCL is designed to be as close to standard C++ as possible*



DPC++ (Data Parallel C++) and SYCL

- SYCL
  - Standard developed by Khronos and announced in 2014
  - The latest SYCL specification (SYCL 2020) was released in 2021
  - SYCL is a C++ based abstraction layer (standard C++17)
  - Builds on OpenCL **concepts** (but single-source)
  - *SYCL is designed to be as close to standard C++ as possible*

- DPC++
  - Part of Intel oneAPI specification and Intel's implementation of SYCL
  - Intel extension of SYCL to support new innovative features
  - Open source and available on GitHub
  - Contains a Plugin Interface (PI) to allow DPC++ to run on multiple devices



OpenMP

- OpenMP is a widely supported and utilized programming model
- OpenMP 5 constructs will provide directives based programming model for Intel GPUs
- Available for C, C++, and Fortran and optimized for Aurora
- Current OpenMP 5.1 spec supports offloading to an accelerator/GPU
  - Support started with OpenMP 4
- OpenMP with offload support offers a potential path to developing performance portable applications
- Multiple compilers and vendors providing OpenMP implementations
- Community has a consensus what is the “most common” subset of OpenMP features to be supported on devices.
  - OpenMP features inappropriate to GPUs are often not implemented

Intel Fortran for Aurora

- ❑ Fortran 2008
- ❑ OpenMP 5
- ❑ New compiler—LLVM backend
  - ❑ Strong Intel history of optimizing Fortran compilers
- ❑ Beta available today in oneAPI toolkits

https://software.intel.com/content/www/us/en/develop/tools/oneapi/components/fortran-compiler.html

Intel MKL – Math Kernel Library

- Highly tuned algorithms
  - FFT
  - Linear algebra (BLAS, LAPACK)
  - Sparse linear algebra
  - Statistical functions
  - Vector math
  - Random number generators

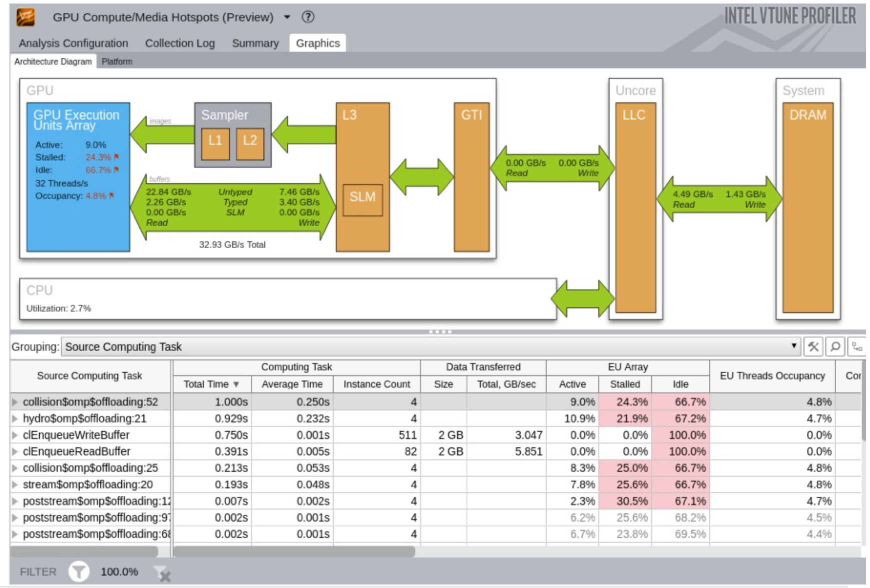
- Optimized for every Intel platform

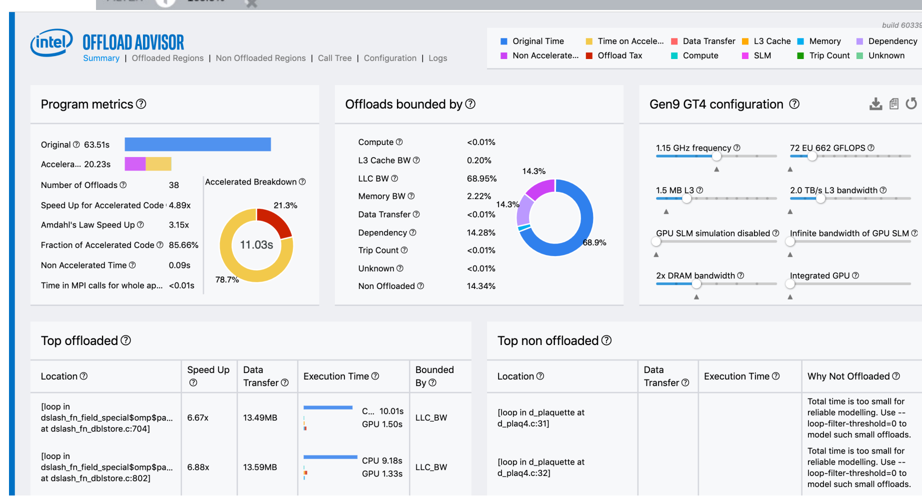
- oneAPI MKL (oneMKL)
  - https://software.intel.com/en-us/oneapi/mkl

Latest oneAPI toolkits include DPC++ support and C/Fortran OpenMP offload

Intel VTune and Advisor

- ❑ Vtune Profiler
  - ❑ Widely used performance analysis tool
  - ❑ Supports analysis on Intel GPUs
- ❑ Advisor
  - ❑ Provides roofline analysis
  - ❑ Offload analysis will identify components for profitable offload
    - ❑ Measure performance and behavior of original code
    - ❑ Model specific accelerator performance to determine offload opportunities
    - ❑ Considers overhead from data transfer and kernel launch





Information and Help

- User documentation is available at 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