

**ISC**

High Performance

IMAGINE

TOMORROW

MAY 21 – 25, 2023 | HAMBURG, GERMANY

## Aurora Exascale Architecture

*Kalyan Kumaran ([kumaran@anl.gov](mailto:kumaran@anl.gov))*

*Director of Technology  
Argonne Leadership Computing Facility  
Argonne National Lab*

# About Argonne

**Argonne is a multidisciplinary science and engineering research center located outside Chicago.**

- Born out of the University of Chicago's work on the Manhattan Project in the 1940s.
- Managed by UChicago Argonne, LLC, for the U.S. Department of Energy's Office of Science.
- Works with universities, industry, and other national labs on questions and experiments too large for any one institution to do by itself.

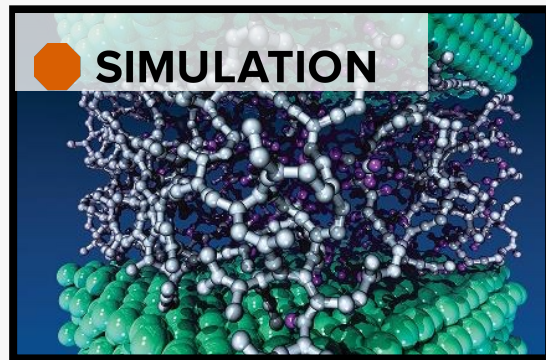
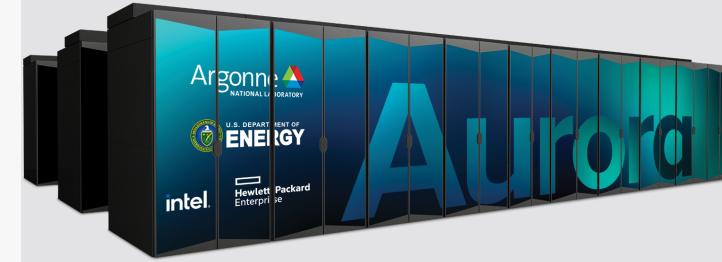


# Argonne Leadership Computing Facility



The Argonne Leadership Computing Facility provides world-class computing resources to the scientific community.

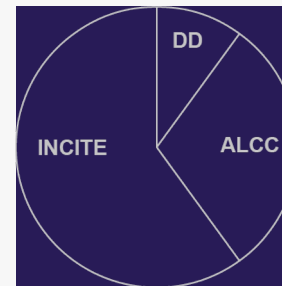
- Users pursue scientific challenges
- In-house experts to help maximize results
- Resources fully dedicated to open science



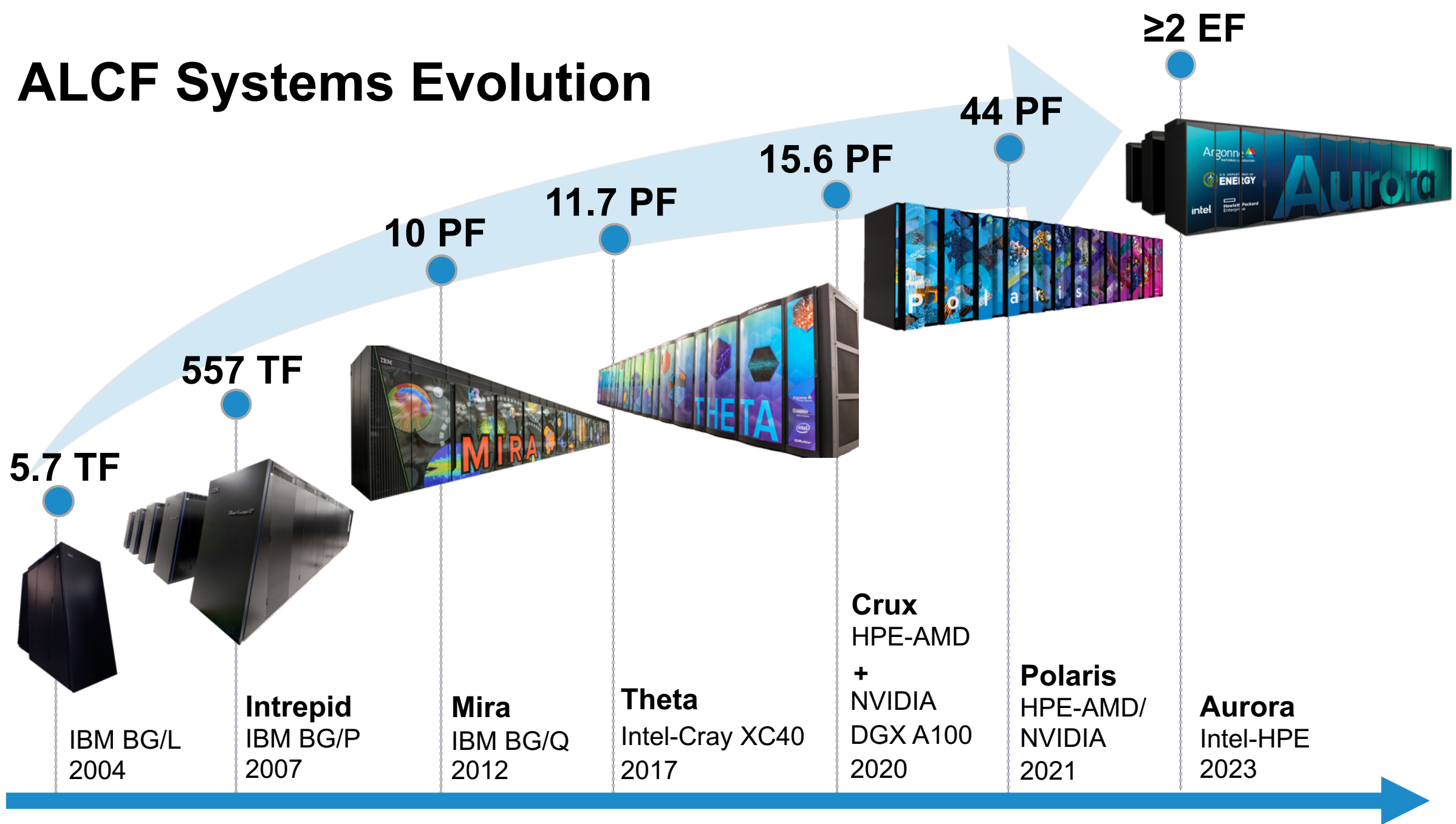
Architecture supports three types of computing

- Large-scale Simulation (PDEs, traditional HPC)
- Data Intensive Applications (scalable science pipelines)
- Deep Learning and Emerging Science AI (training and inferencing)

ALCF offers different pipelines based on your computational readiness. Apply to the allocation program that fits your needs.



# ALCF Systems Evolution







# Computing Resources

## Polaris

- HPE Apollo Gen10+
- AMD processors/NVIDIA GPUs
- 44 petaflops (double precision)
- NVIDIA GPU A100; HBM stack
- AMD EPYC Processor Milan
- 560 nodes

## Theta

### KNL NODES

- Intel-Cray XC40
- 11.7 petaflops
- 4,392 nodes
- 281,088 cores
- 843 TB (DDR4); 70 TB (HBM) of memory

### GPU NODES

- NVIDIA DGX A100
- 3.9 petaflops
- AMD EPYC 7742
- 24 nodes
- 24 TB of DDR4; 7.7 TB (HBM) of memory

## Cooley

- Cray/NVIDIA
- 126 nodes
- 1512 Intel Haswell CPU cores
- 126 NVIDIA Tesla K80 GPUs
- 48 TB RAM / 3 TB GPU

## Iota

- Intel/Cray XC40 architecture
- 117 teraflops
- 44 nodes
- 2,816 cores
- 12.3 TB of memory

## JLSE Experimental Testbeds

- 150 nodes
- Intel/AMD/IBM/Marvell/GPGPU
- EDR/100GbE/OPA
- Lustre/GPFS/DAOS

## Grand and Eagle (Storage)

Each system has:

- HPE ClusterStor E1000
- 100 petabytes of usable capacity
- 8,480 disk drives
- Lustre filesystem
  - 160 Object Storage Targets
  - 40 Metadata Targets
- HDR InfiniBand network
- 650 GB/s rate on data transfers

# ALCF AI Testbed

## Next-Generation AI-Accelerators

- Infrastructure of next-generation machines with hardware accelerators customized for artificial intelligence (AI) applications with a goal to integrate AI accelerators in existing and upcoming supercomputers
- Provides a platform to evaluate usability and performance of machine learning-based HPC science applications running on these accelerators.
- Promising results for diverse spectrum of science ranging from cancer, covid19, high-energy physics, biosciences, climate, among others.
- Close collaboration with AI accelerator vendors on their product developments and roadmaps



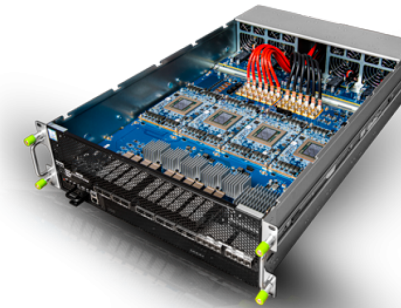
Cerebras (CS-2)



SambaNova



Graphcore



Habana

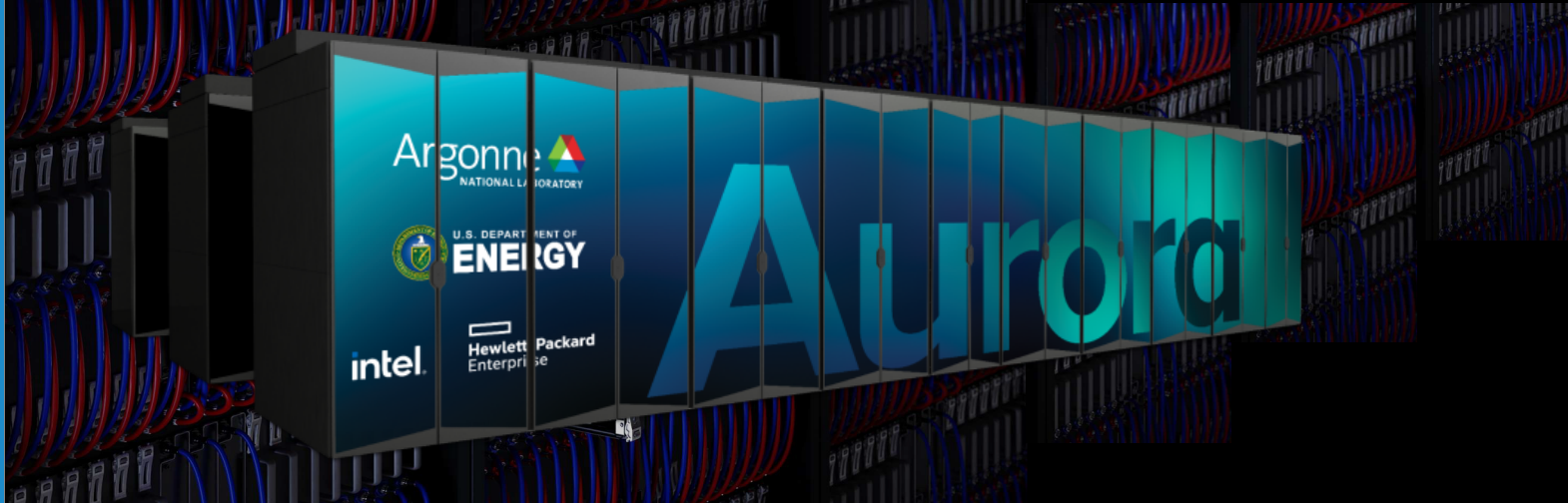


Groq

<https://www.alcf.anl.gov/alcf-ai-testbed>

# AURORA OVERVIEW





Argonne's upcoming exascale supercomputer will leverage several technological innovations to support machine learning and data science workloads alongside traditional modeling and simulation runs.

**Peak Performance**

≥ 2 Exaflops DP

**Intel GPU**

Intel® Data Center GPU Max Series

**Intel Xeon Processor**

Intel® Xeon Max Series 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

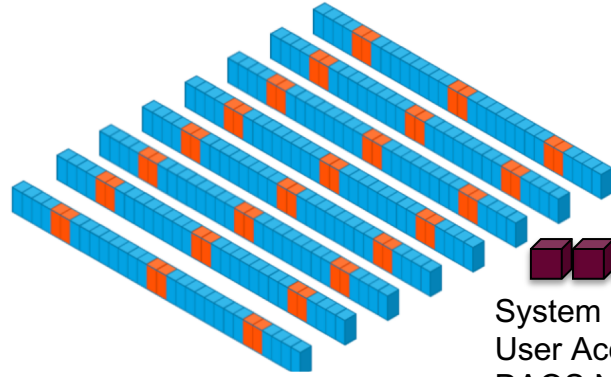
**Network Switch**

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

**Programming Environment**

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

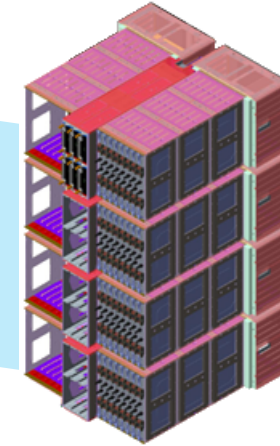
# Aurora High-level System Overview



## AURORA SYSTEM

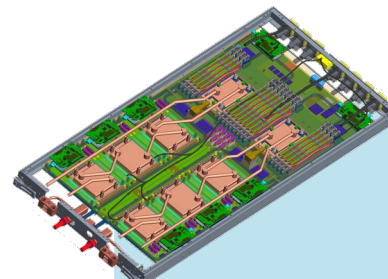
166 Compute racks  
10,624 Nodes  
GPU: 8.16 PB HBM  
CPU: 1.36 PB HBM, 10.9 PB DDR5  
DAOS: 64 racks, 1024 nodes  
230 PB (usable), 31 TB/s

System Service Nodes (SSNs)  
User Access Nodes (UANs)  
DAOS Nodes (DNs)  
Gateway Nodes (GNs)  
IOF service, scalable library loading  
DAOS <-> Lustre data mover



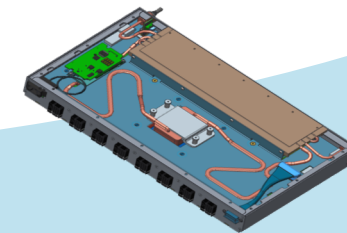
## COMPUTE RACK

64 Compute blades  
32 Switch blades  
GPU: 49.1 TB HBM  
CPU: 8.2 TB HBM, 64 TB DDR5



## COMPUTE BLADE

2 Intel Xeon Max Series w HBM  
6 Intel Data Center GPI Max Series  
GPU: 768 GB HBM  
CPU: 128 GB HBM, 1024 GB DDR5



## SWITCH BLADE

1 Slingshot switch  
64 ports  
Dragonfly topology

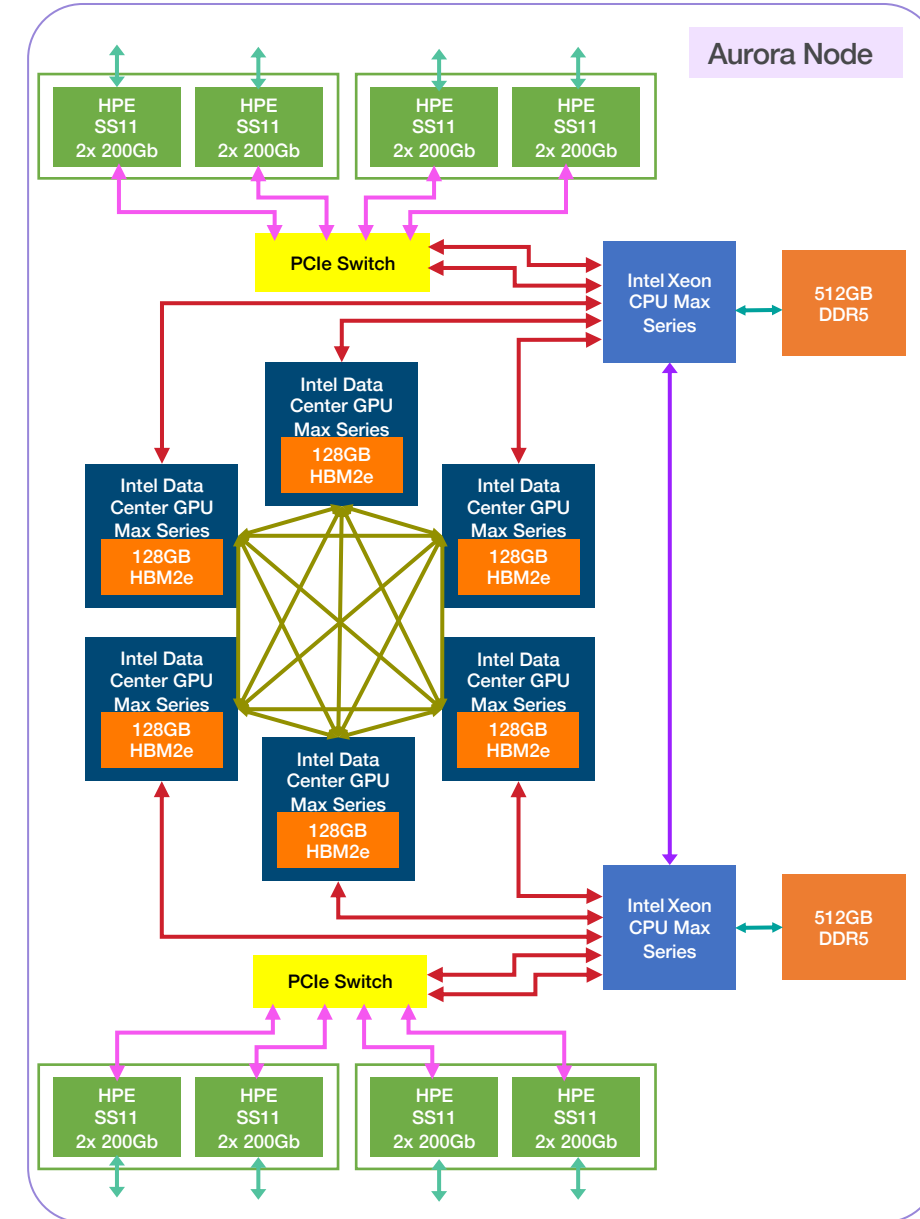
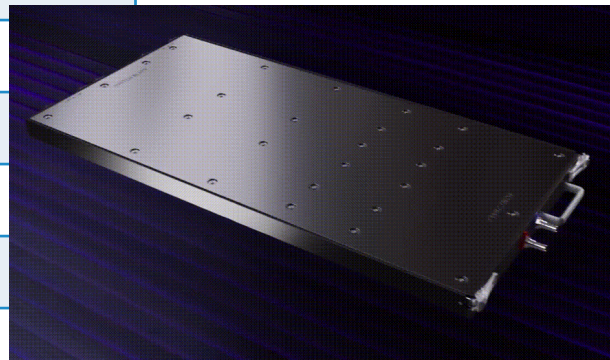
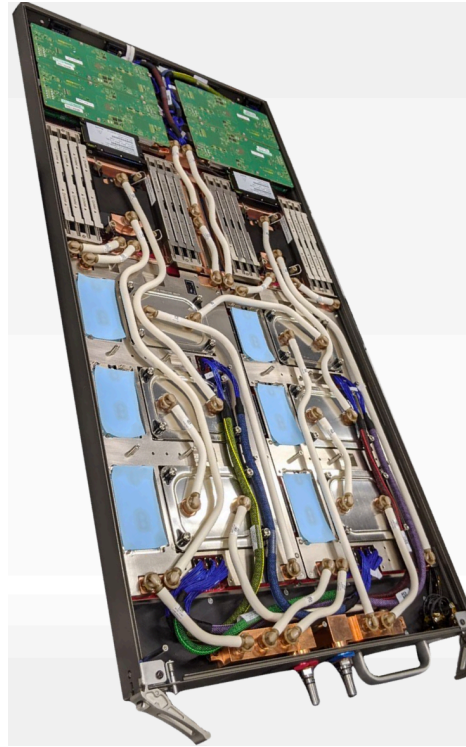
# AURORA NODE



# 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 (#)



# 4<sup>th</sup> Gen Intel® Xeon Max Series CPU with HBM (Sapphire Rapids)

XEON DESCRIPTION	
Vector Extension	AVX-512
Threads (#)	2
Total HBM Memory (GB)	64
Peak HBM Memory BW (TB/s)	1.43
Total DDR5 4400 Memory (GB)	512
Peak DDR5 4400 Memory BW (TB/s)	0.28

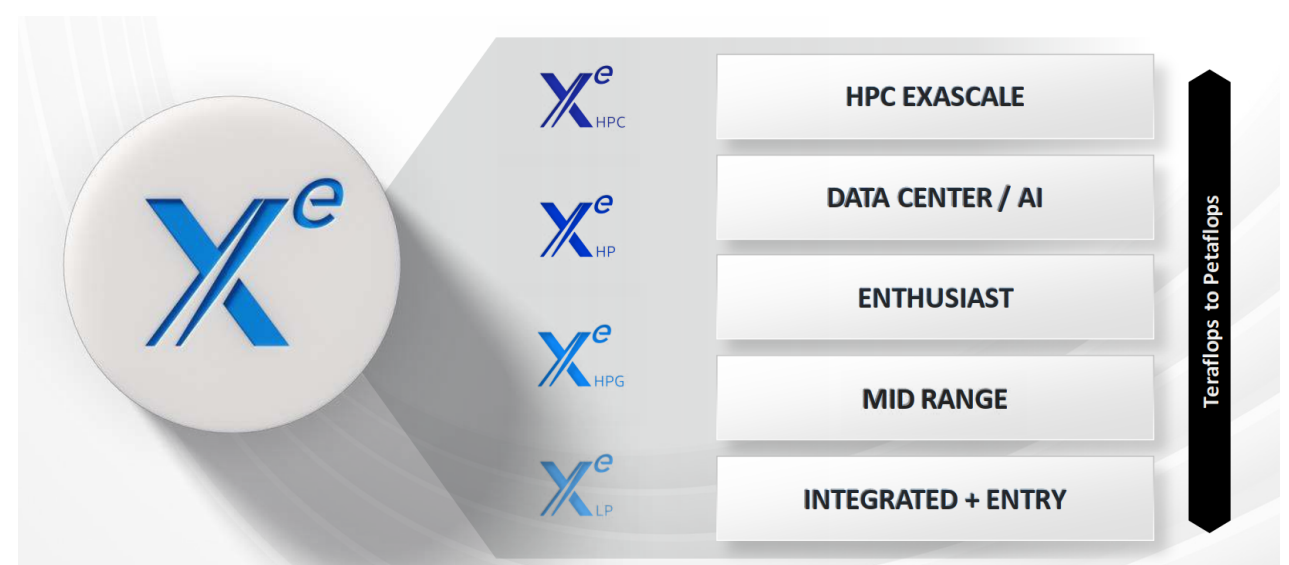
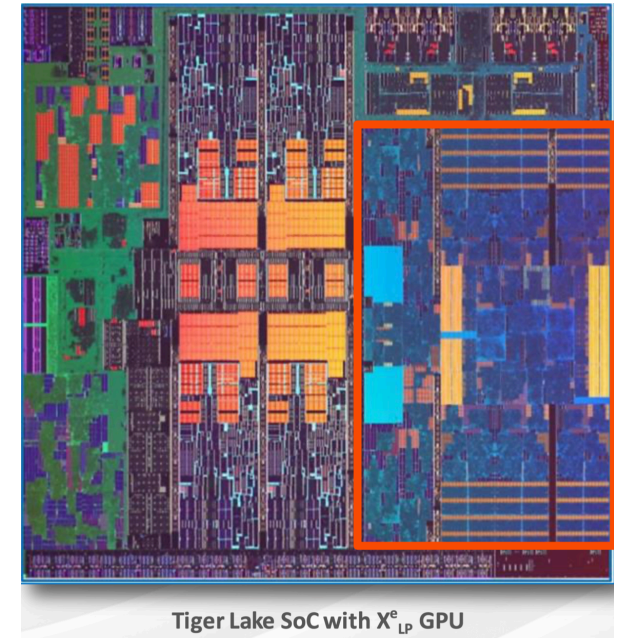


**SAPPHIRE RAPIDS**

- Breakthrough Technology**
  - DDR5: Increased Memory BW
  - PCIe 5: High Throughput
  - CXL 1.1: Next-gen IO
- Built-In AI Acceleration**
  - Intel® Advanced Matrix Extensions (AMX): Increased Deep Learning Inference and Training Performance
- Agility and Scalability**
  - Hardware Enhanced Security
  - Intel® Speed Select Technology
  - Broad Software Optimization
- NEW High Bandwidth Memory**
  - Significant performance increase for bandwidth-bound workloads

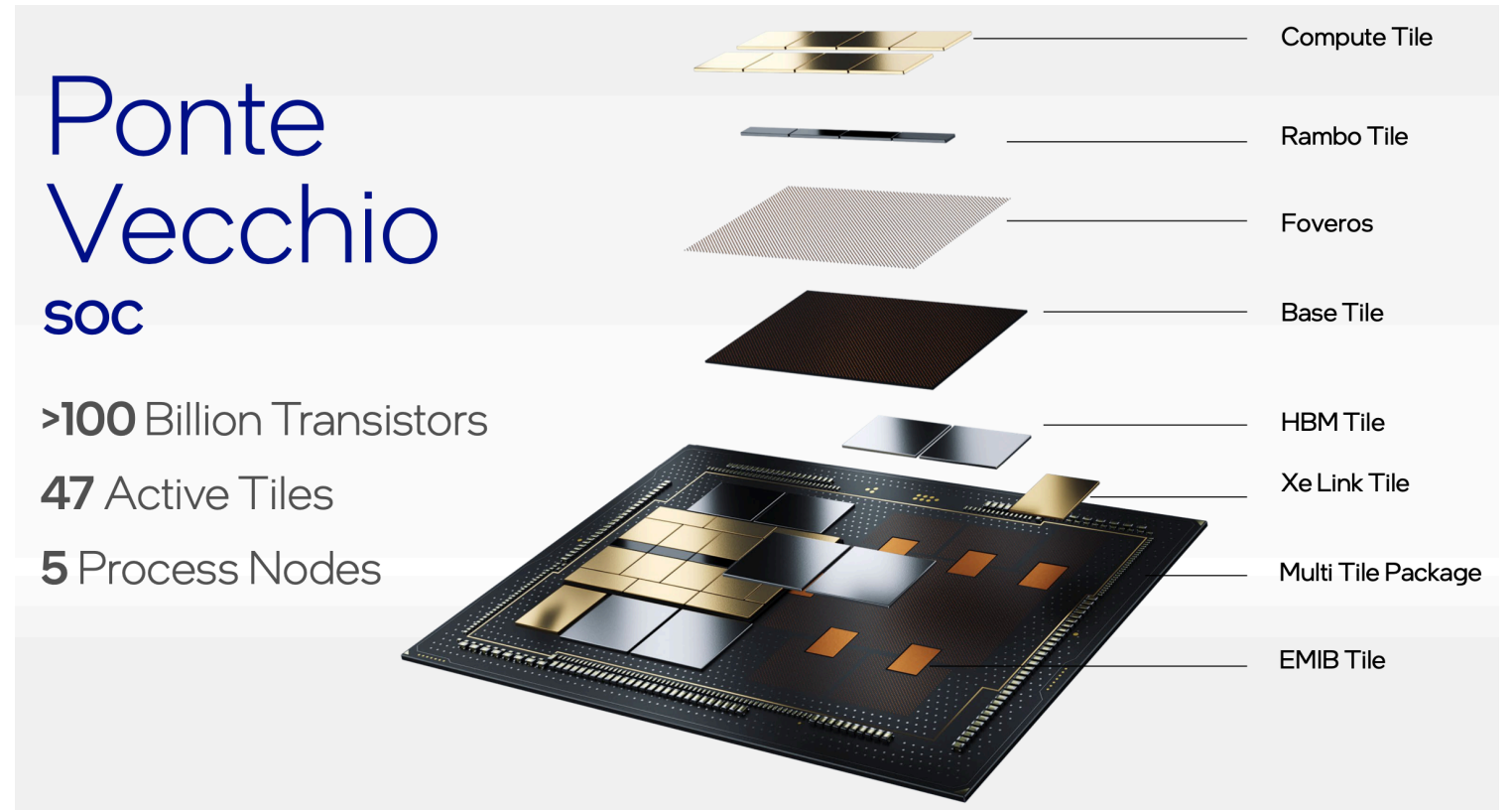
# Intel GPUs

- Intel has been building integrated GPUs for over a decade
- These have evolved into Xe architecture used in next gen GPUs
  - Xe LP
    - Platforms: Tiger Lake, Iris Xe Max
    - Integrated low power
  - Xe HP/HPG
    - DG2/Intel Arc GPU
    - Discrete & High power
  - Xe HPC
    - Ponte Vecchio
    - High performance computing



# Intel® Data Center GPU Max Series (Ponte Vecchio)

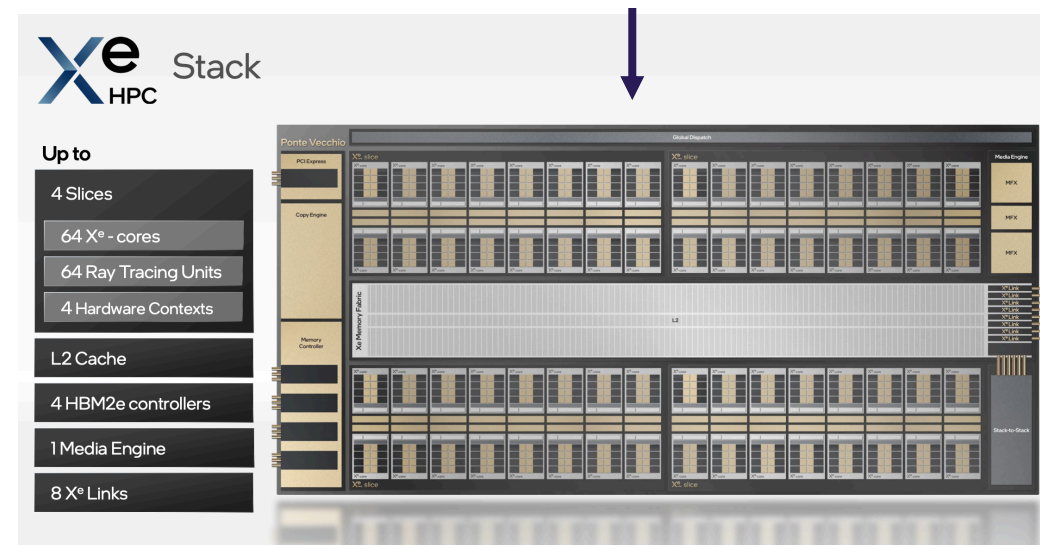
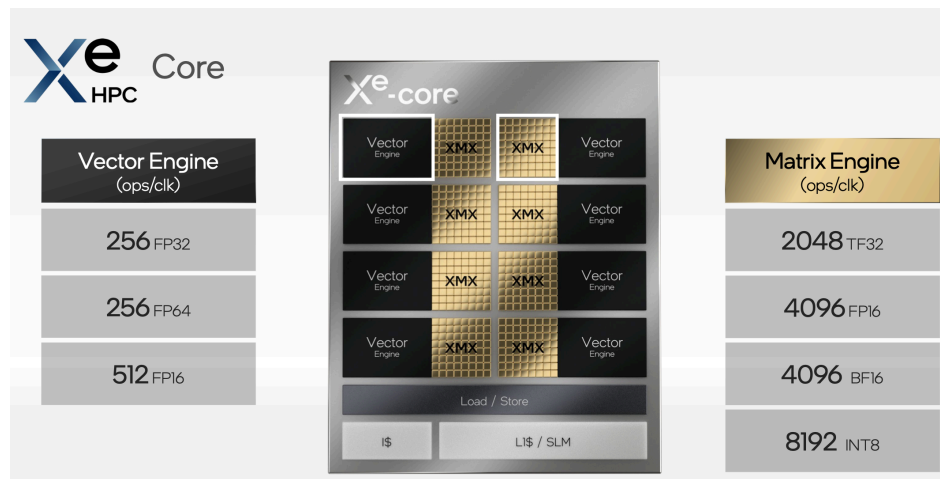
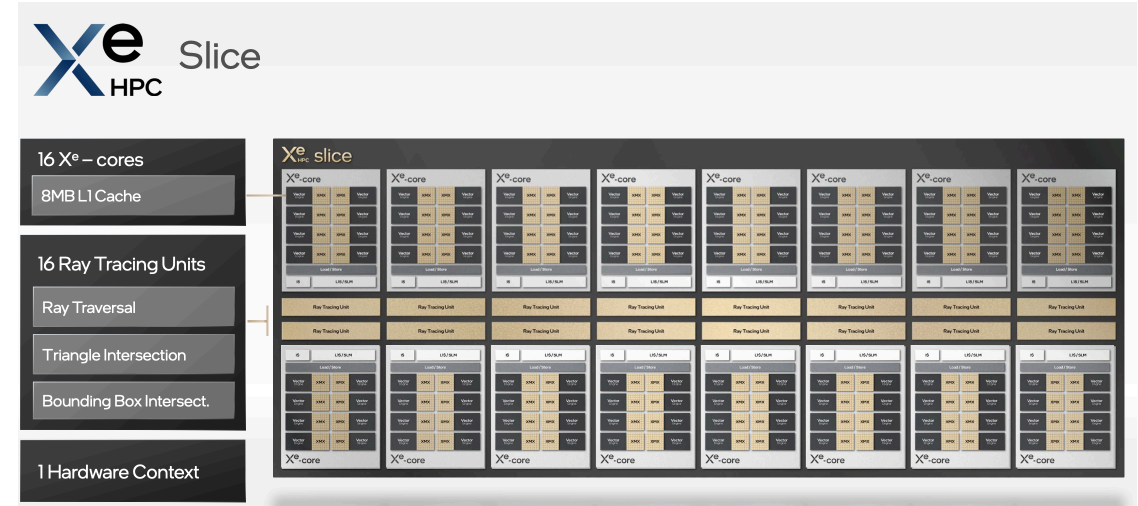
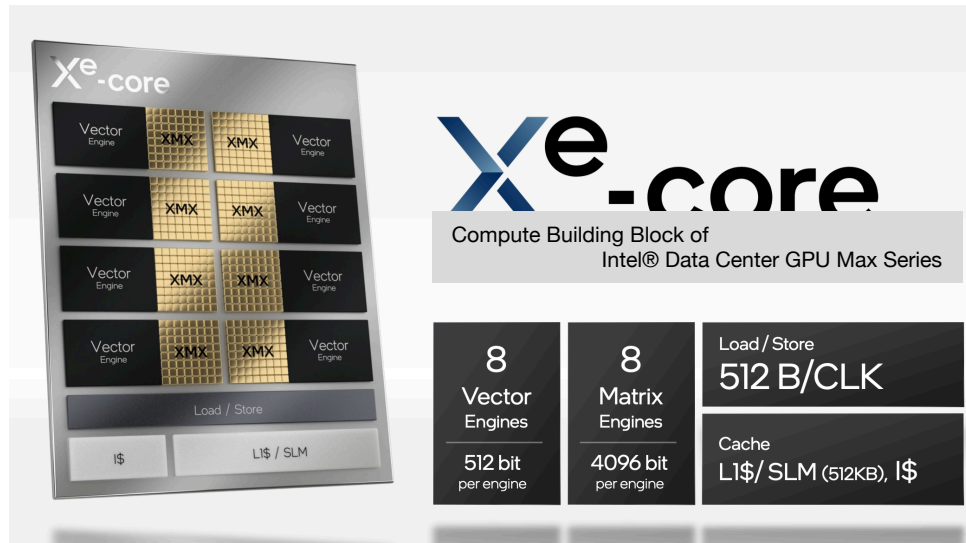
- Multi Tile architecture
- Compute Tile
  - Xe Cores
  - L1 Cache
- Base Tile
  - PCIe Gen5
  - HBM2e Main Memory
  - MDFI
  - EMIB
- Connectivity Tile
  - Xe link



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



# Intel® Data Center GPU Max Series Architectural Components



# AURORA FABRIC

# HPE Slingshot Interconnect

## Consistent, Repeatable Application Performance

- Advanced congestion control
- Fine grained adaptive routing
- Very low average and tail latency

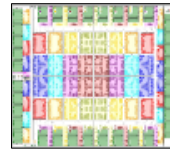
## Extremely Scalable RDMA Performance

- Connectionless protocol
- Fine grained flow control
- MPI HW tag matching & progress engine
- Dragonfly topology – 3 switch hops (typical)

## Native Ethernet

- Native IP – no encapsulation
- High-scale bandwidth integration to campus

## HPE Slingshot Switches - 64 ports @ 200 Gbps



HPE Switch ASIC

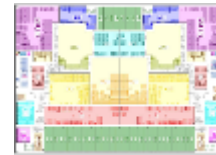


Rack switches

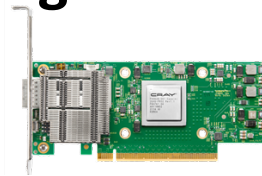


100% DLC Switches

## HPE Slingshot NICs - 200 Gbps



HPE NIC ASIC

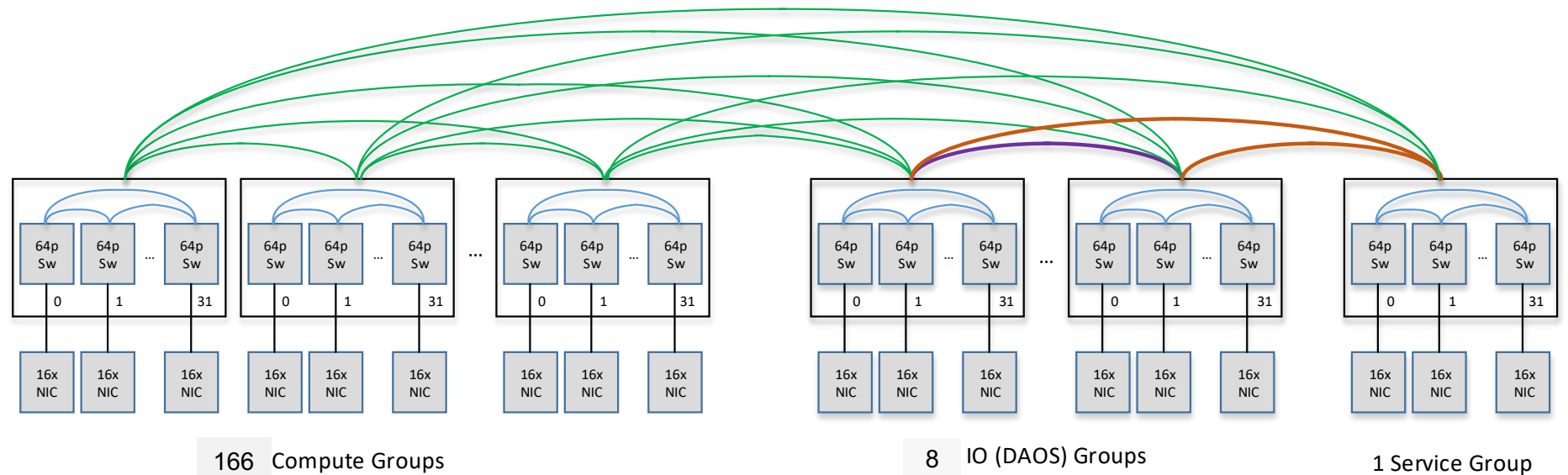


PCIe Adapters



100% DLC NIC Mezz

# Fabric



Each Link is 50GB/s bidirectional, 25GB/s unidirectional:

1 link per arc

2 links per arc

8 links per arc

24 links per arc

- 1-D Dragonfly Topology - 175 total groups (166 compute + 8 IO + 1 Service),
- All the global links are optical, all the local links in compute groups are electrical
- 2 global links between any two compute groups
- 24 links between any two IO groups, 8 links between the Service group and each IO group
- Total injection bandwidth: 2.12PB/s
- Total bisection bandwidth: 0.69PB/s



# AURORA STORAGE

# Aurora Storage Systems

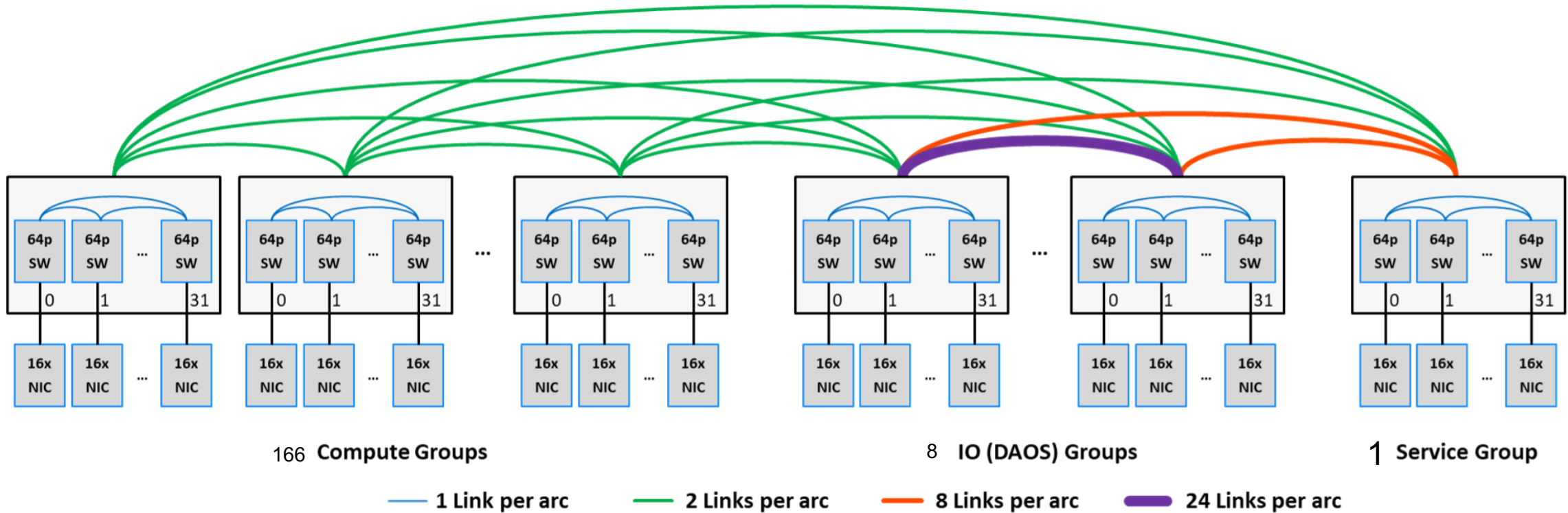


- DAOS provides Aurora's main "platform" high performance storage system
- Aurora leverages existing Lustre storage systems, Grand and Eagle, for center-wide data access and data sharing

System	Capacity	Performance
Aurora DAOS	230 PB @ EC16+2 <ul style="list-style-type: none"><li>▪ 250 PB NVMe</li><li>▪ 8 PB Optane PMEM</li></ul>	31 TB/s Read & Write
Eagle	100 PB @ RAID6 <ul style="list-style-type: none"><li>▪ 8480 HDD</li><li>▪ 40 Lustre MDT</li></ul>	> 650 GB/s Read & Write
Grand	100 PB @ RAID6 <ul style="list-style-type: none"><li>▪ 8480 HDD</li><li>▪ 40 Lustre MDT</li></ul>	> 650 GB/s Read & Write

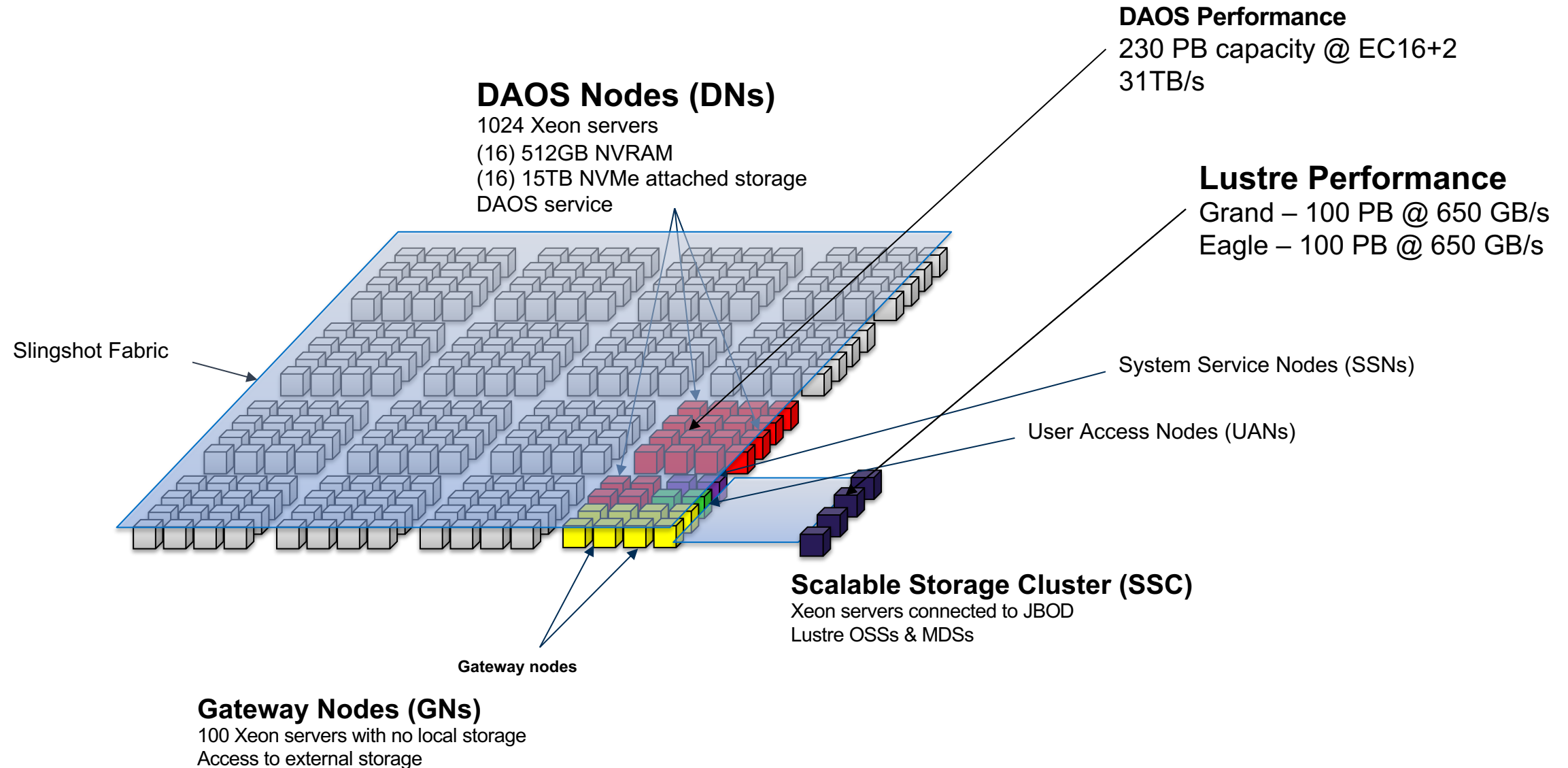


# Aurora Network Architecture



- Increased DAOS inter-group bandwidth
  - Support rebuilding and inter-server communication
  - Prevent DAOS server traffic interfering with application communication
- Increased bandwidth to service group
  - Support off-cluster access and data-movement

# Aurora Storage Overview





# DAOS Node Details

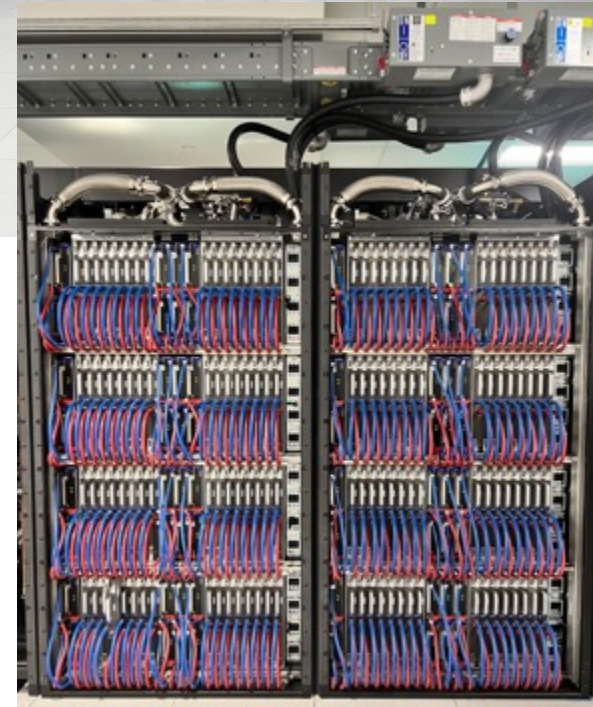
- Intel Coyote Pass System
  - (2) Xeon 5320 CPU (Ice Lake)
  - (16) 32GB DDR4 DIMMs
  - (16) 512GB Intel Optane Persistent Memory 200
  - (16) 15.3TB Samsung PM1733
  - (2) HPE Slingshot NIC
- 1024 Total Servers
  - Each node will run 2 DAOS engines
  - 2048 DAOS engines



# AURORA INSTALLATION



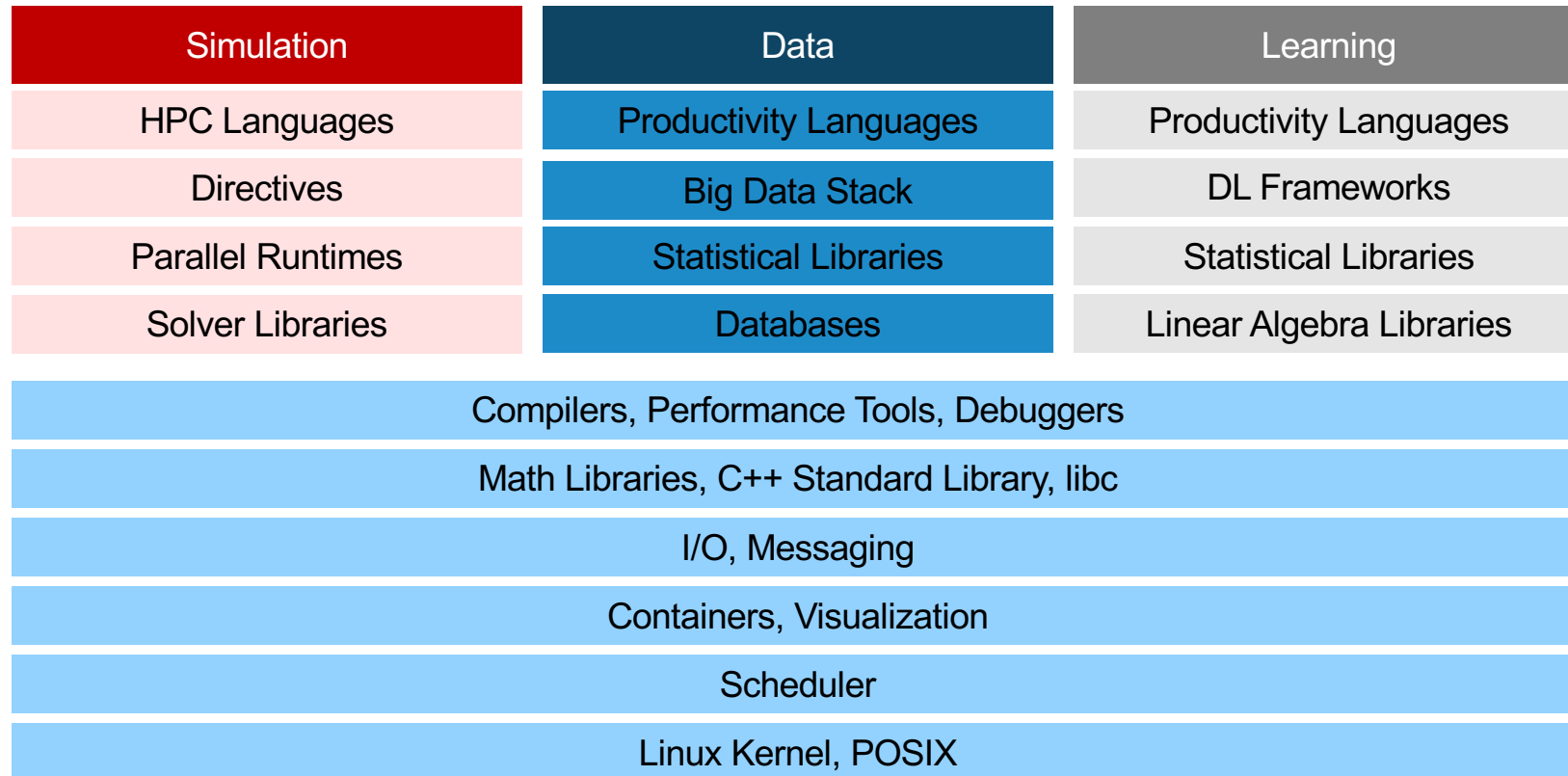
# Aurora Cabinets Installation at Argonne



# AURORA SOFTWARE STACK & PROGRAMMING MODELS



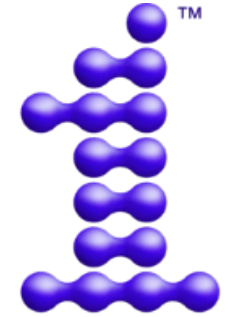
# Three Pillars of Aurora



# oneAPI

“**oneAPI** is a cross-industry, open, standards-based unified programming model that delivers a common developer experience across accelerator architectures—for faster application performance, more productivity, and greater innovation.”

-- oneapi.com



**oneAPI**

## Three Components

- Language
  - DPC++
- Libraries
  - oneMKL, oneDAL, ...
- Hardware Abstraction Layer
  - Level Zero (L0)

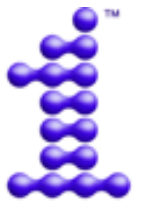
Set of specifications that any one can implement

Intel has their own implementations

<https://software.intel.com/ONEAPI>

Good documentation for understanding what will be on Aurora

# Aurora oneAPI Components



oneAPI

## Languages & Runtimes

DPC++ Compiler (CPU & GPU)

DPC++ Compatibility Tool

C/C++/Fortran OpenMP Offload Compiler  
(CPU & GPU)

Compiler/Compatibility IDE Plugins

Intel Distribution for Python

Parallel STL / oneDPL

oneTBB

oneCCL

Aurora MPICH

## Tools

Debugger

VTune

Inspector

Advisor

GT-PIN

## Math Libraries

oneDAL

oneMKL

oneDNN

Visualization

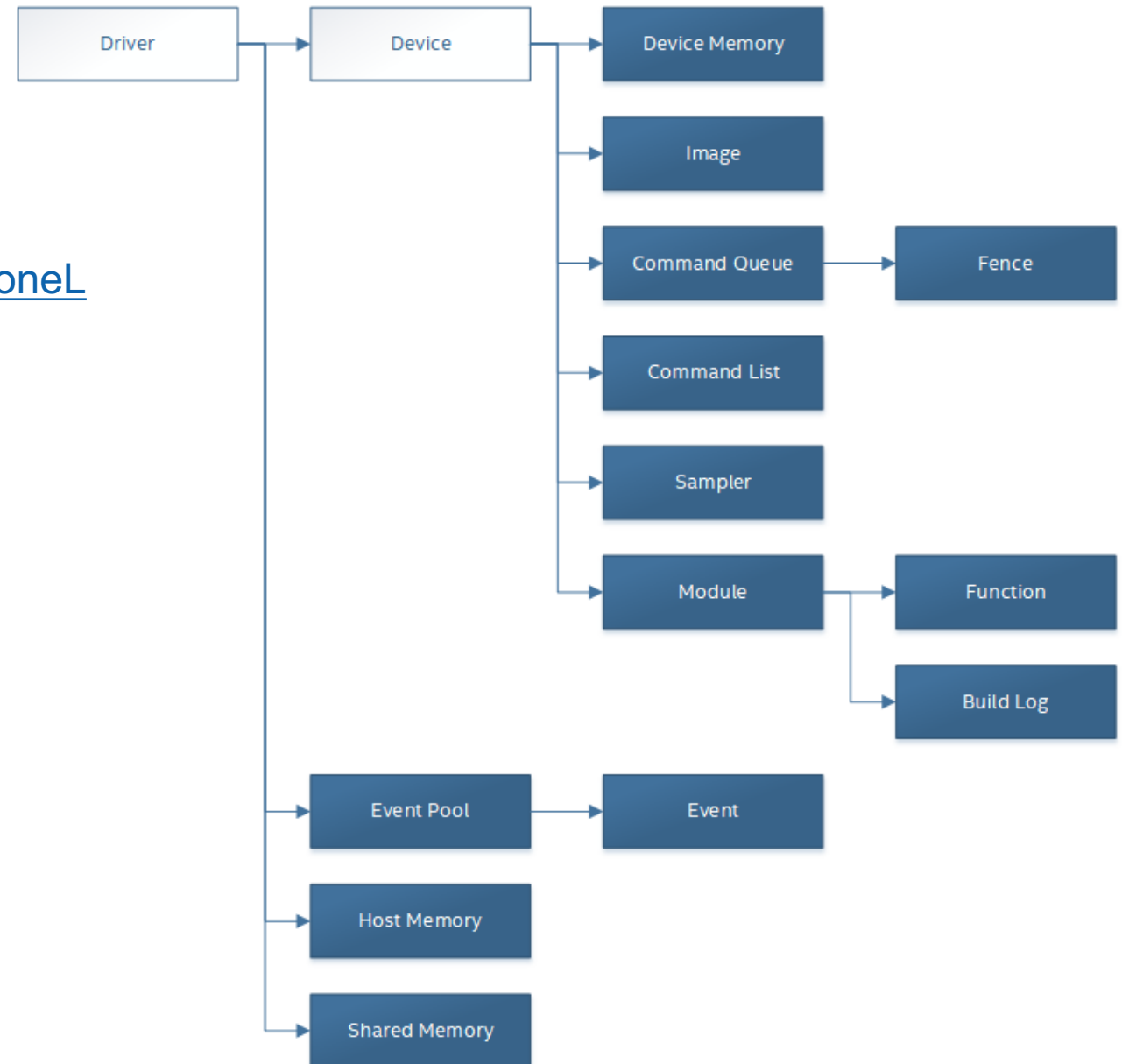
## Frameworks

PyTorch (Alpha)

TensorFlow  
(Alpha)

# Level Zero (L0)

- Low-level programming model for fine grained control of device
  - <https://spec.oneapi.com/versions/latest/oneL0/index.html>
- Management of:
  - Device memory
  - Synchronization
  - Command queue and command lists
  - And more

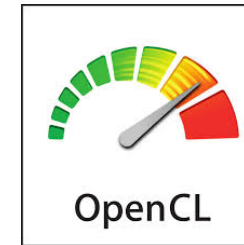


# Aurora Programming Models

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

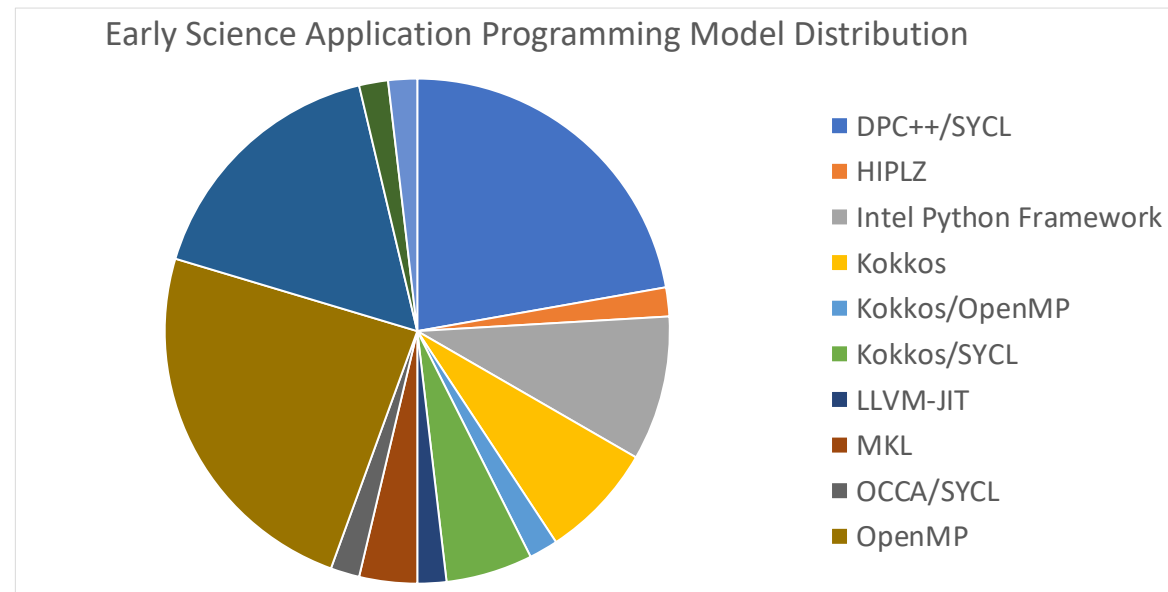


HIP



- Experimental
  - HIP
- Not available on Aurora:
  - CUDA
  - OpenACC

Early Science Application Programming Model Distribution



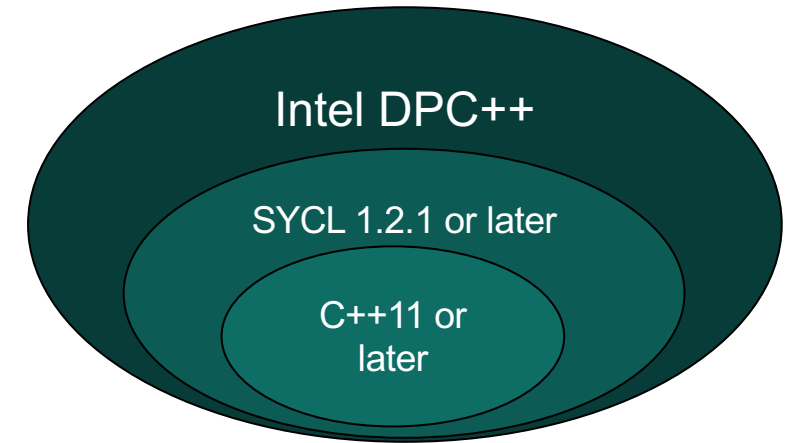
# DPC++ (SYCL)

## DPC++

- Intel implementation of SYCL standard
- Add language or runtime extensions as needed to meet user needs
- Incorporates SYCL 1.2.1 specification and Unified Shared Memory
- Part of Intel oneAPI specification

## SYCL

- Khronos standard specification
- SYCL is a C++ based abstraction layer (standard C++11)
- Based on OpenCL concepts (but single-source)
- *SYCL is designed to be as close to standard C++ as possible*
- Current Implementations of SYCL:
  - ComputeCPP™ (www.codeplay.com)
  - Intel SYCL (github.com/intel/llvm)
  - triSYCL (github.com/triSYCL/triSYCL)
  - hipSYCL (github.com/illuhad/hipSYCL)
  - **Runs on today's CPUs and nVidia, AMD, Intel GPUs**

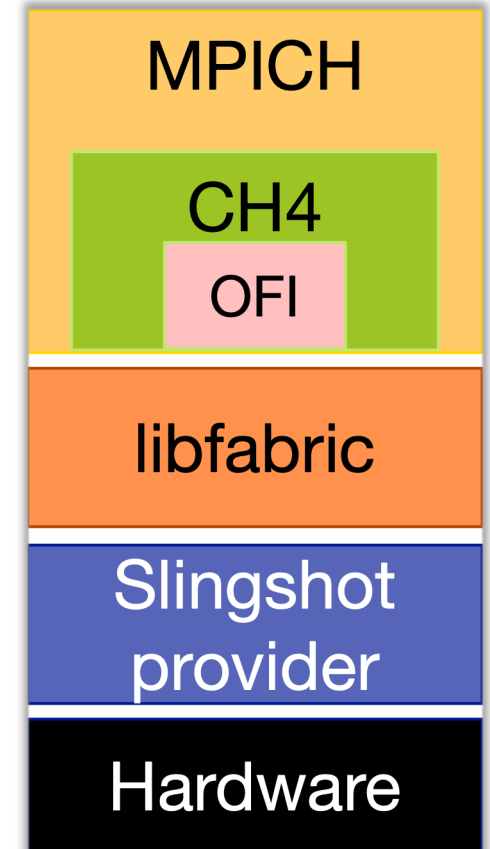


Extensions	Description
Unified Shared Memory (USM)	defines pointer-based memory accesses and management interfaces.
In-order queues	defines simple in-order semantics for queues, to simplify common coding patterns.
Reduction	provides reduction abstraction to the ND-range form of parallel for.
Optional lambda name	removes requirement to manually name lambdas that define kernels.
Subgroups	defines a grouping of work-items within a work-group.
Data flow pipes	enables efficient First-In, First-Out (FIFO) communication (FPGA-only)

[https://spec.oneapi.com/oneAPI/Elements/dpcpp/dpcpp\\_root.html#extensions-table](https://spec.oneapi.com/oneAPI/Elements/dpcpp/dpcpp_root.html#extensions-table)

# MPI

- MPICH for Aurora
- Based on open source MPICH with new features to support Aurora
- Uses OFI (Open Fabrics Interface) to communicate with the Slingshot Interconnect
- Redesigned to reduce instruction counts and remove non-scalable data structures
- Innovative collective algorithms
- Optimized Threading Support
- Shared Memory Optimizations through XPMEM
- MPICH is GPU aware for Intel GPUs
  - It is built on top of oneAPI Level Zero
  - It supports point to point, one-sided, and collectives
  - Support for different data types through the Yaksa library
  - OSU benchmarks with Level Zero used to validate correctness and performance



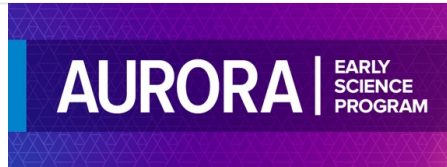
# MPI (Contd)

- Multiple optimizations added to support the unique hardware features of Aurora
  - Intel GPUs and all-to-all connectivity across the GPUs inside the node
  - Multiple NICs on the same node. MPICH supports:
    - Distribution of processes to NICs
    - Striping (a single rank distributes a single message across multiple NICs)
    - Hashing (a single rank sends different messages through different NICs, e.g., depending on the communicator or the target rank)
    - Efficient Multithreading support to use multiple NICs
  - Collectives optimized for Dragonfly network topology
- Most of the optimizations are already upstreamed in the MPICH repository. The rest is coming soon

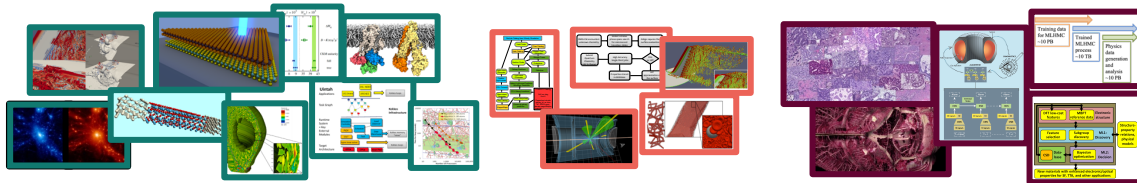


# AURORA APPLICATIONS

# Exascale Applications/Software Readiness



- ALCF Aurora Early Science Program (ESP)
- 9 **Simulation**, 10 **Data** and **Learning** projects
- **Every project will run a proposed science campaign on Aurora**
- Training: Workshops, Hackathons, Dungeon Sessions, webinars
- Argonne postdoc and staff support (Catalysts)

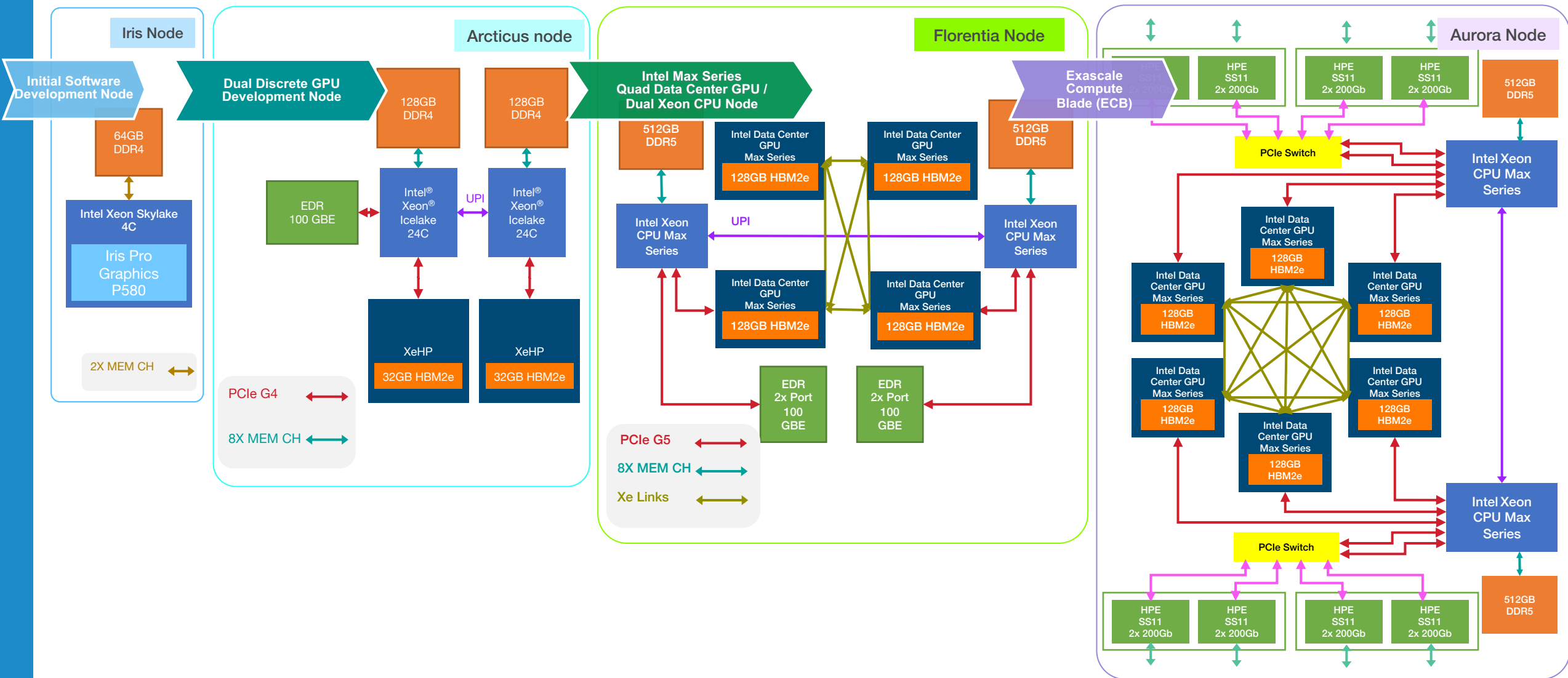


- DOE Exascale Computing Project (ECP)
- 3 technical areas: **Application Development**, **Software Technology**, **Hardware and Integration**
  - **AD**: 21 applications projects preparing codes for exascale
  - **ST**: 66 unique software products
  - **HI**: Applications Integration: deploy apps on specific exascale systems (Aurora, Frontier)
- AppInt funding for Argonne staff for Aurora:
  - ALCF working with 15 ECP AD so far

Argonne-Intel Center of Excellence – dedicated Intel staff

# Evolution of Intel GPU Testbeds at Argonne

2019 - Iris      2020 - Arcticus XeHP GPU      2021 - Arcticus XeHP GPUs v2      2022 - Florentia Intel Max Series GPU      2022 - Sunspot Aurora TDS      2023 - Aurora

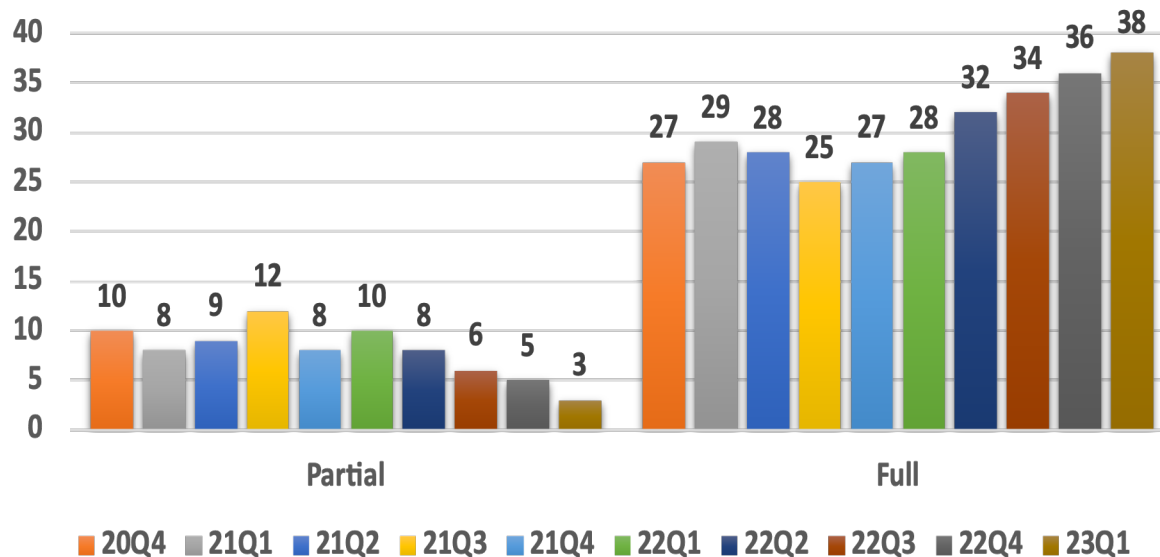


# Tracking Aurora Applications Development

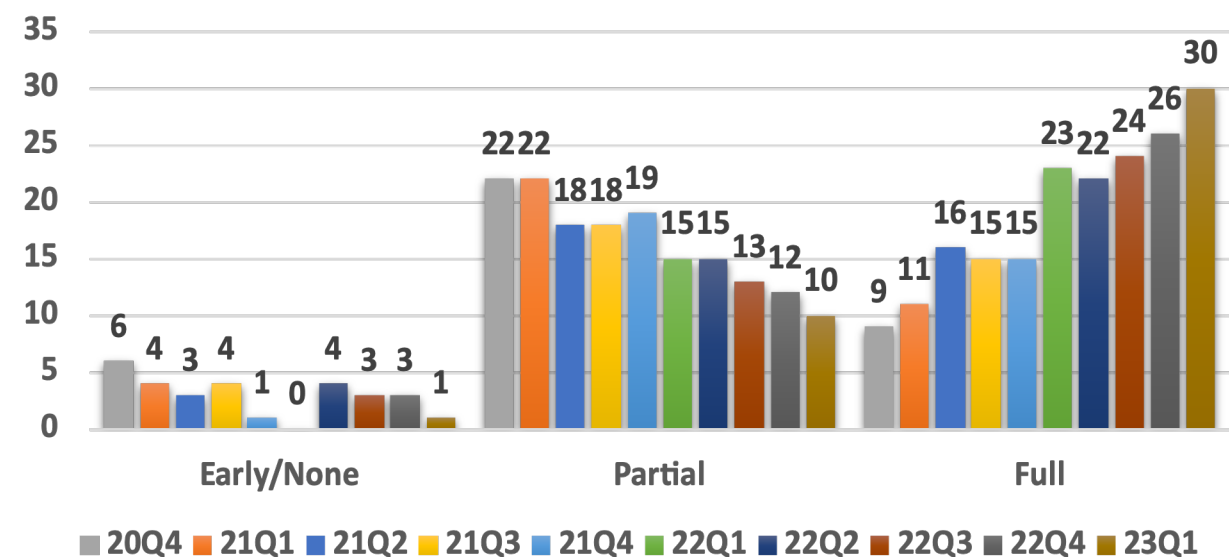
- Steps in application preparation
  - Implementation of science and algorithms
  - Porting to Aurora programming models
  - Testing with Aurora SDK on Aurora testbeds
  - Tuning for performance on Aurora testbeds
  - Scaling across the Aurora system

- ALCF and Intel working with over 40 projects to ready codes for Aurora
- Effort from over 60 Argonne & Intel people and numerous outside teams

## Application Science Implementation



## Port to Aurora Programming Models



# Aurora Applications Status

Running
Running
Running
Partially Running
Porting in Progress

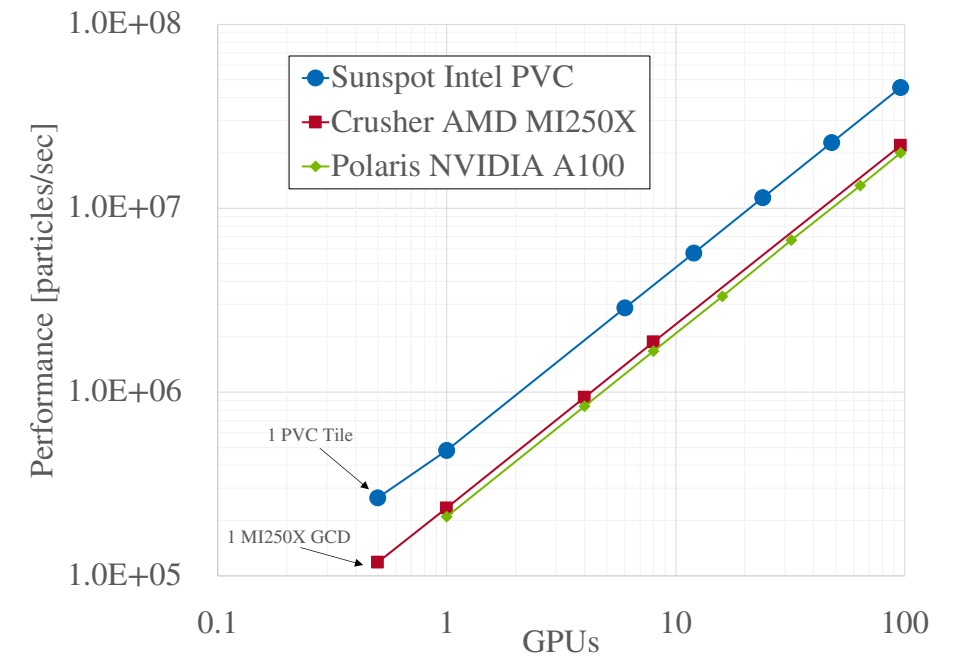
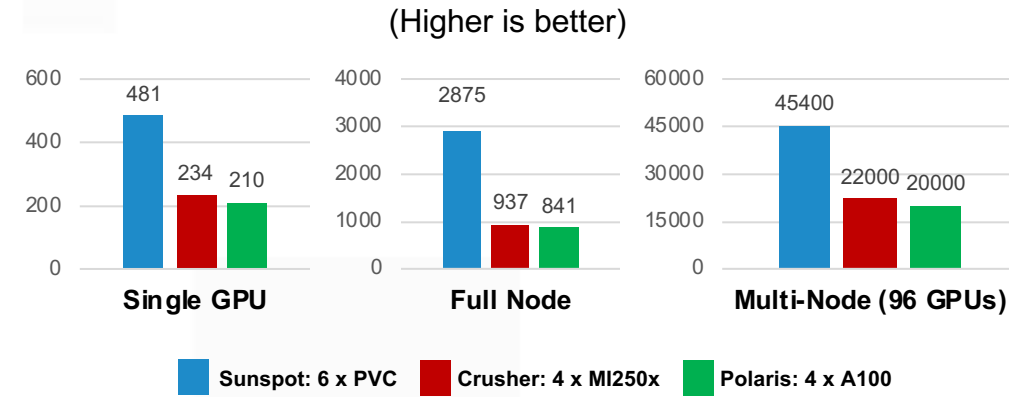
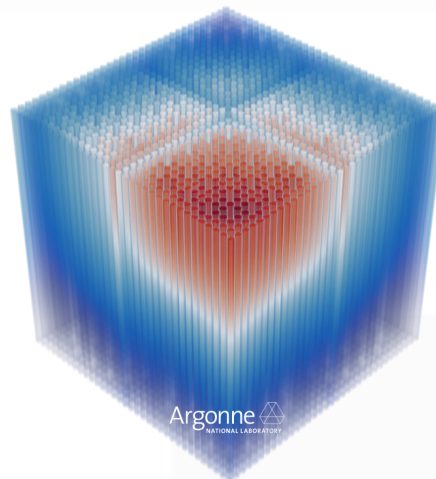
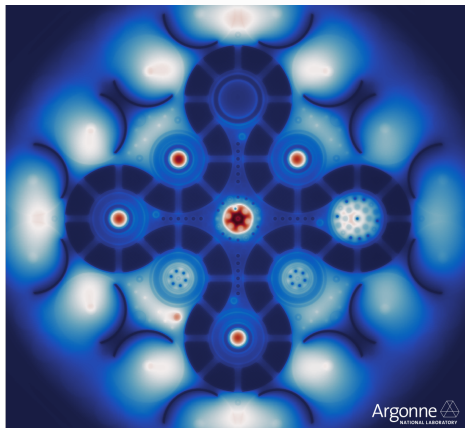
Application	Status
XGC	Running
NAMD	Running
FloodFillNetwork	Running
HACC	Running
QUDA	Running
OpenMC	Running
Flash-X/Thornado	Running
NWChemEX	Running
AMR-Wind	Partially Running
CANDLE/UNO	Partially Running
HARVEY	Partially Running
NekRS	Partially Running
LAMMPS	Partially Running
GENE	Partially Running
FusionDL	Partially Running
MadGraph	Partially Running
BerkelyGW	Porting in Progress
PHASTA	Porting in Progress
MFIX-Exa	Porting in Progress
Chroma	Porting in Progress
cctbx	Porting in Progress

Application	Status
MILC	Porting in Progress
QMCPACK	Partially Running
E3SM-MMF	Partially Running
SW4	Partially Running
DCMesh	Partially Running
LATTE	Partially Running
Grid	Partially Running
GAMESS	Partially Running
NYX	Partially Running
Uintah	Partially Running
Data Driven CFD	Partially Running
DarkSkyMining	Porting in Progress
Flow Based Generative Model	Porting in Progress
Nalu-Wind	Porting in Progress
GEM	Porting in Progress
RXMD-NN	Porting in Progress
mb_aligner	Porting in Progress
spiniFEL	Porting in Progress
Multi-Grid Parameter Opt.	Porting in Progress
FastCaloSim	Porting in Progress

# OpenMC (courtesy of John Tramm)

<https://docs.openmc.org>

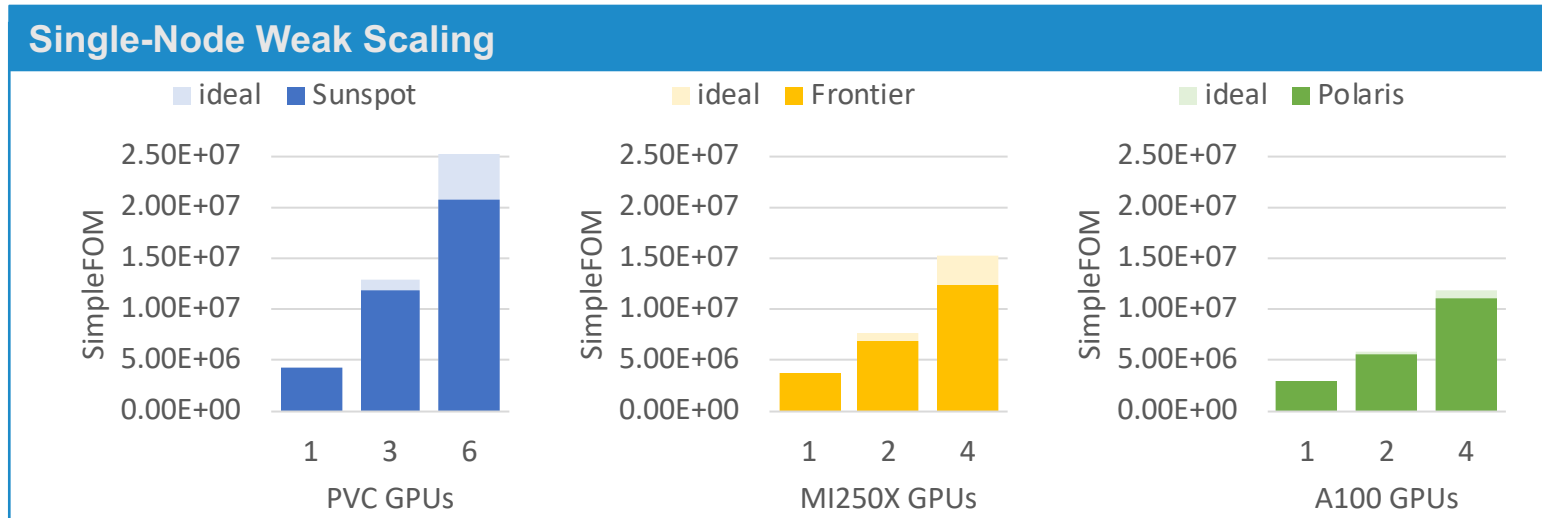
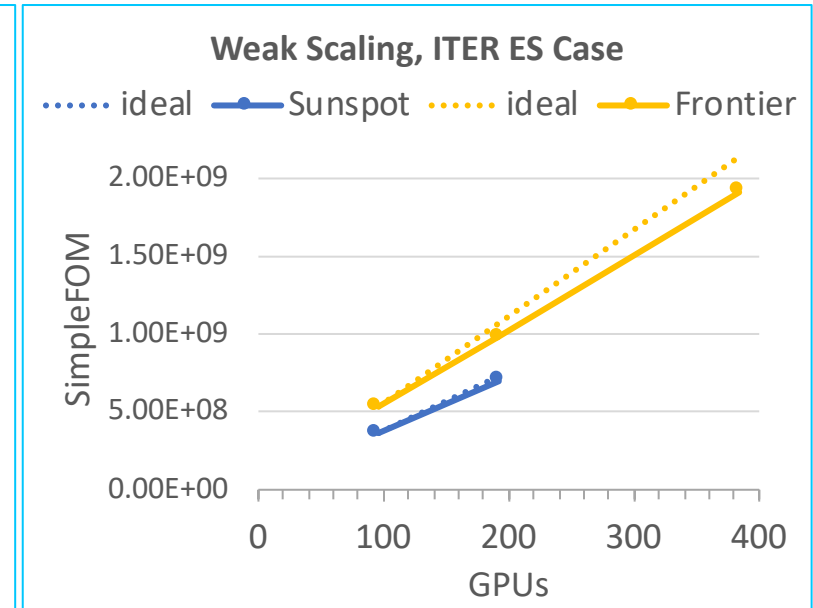
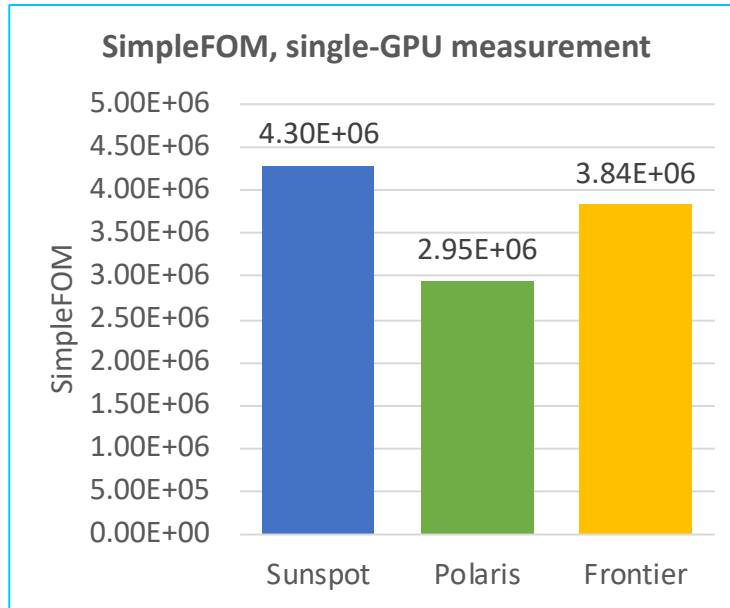
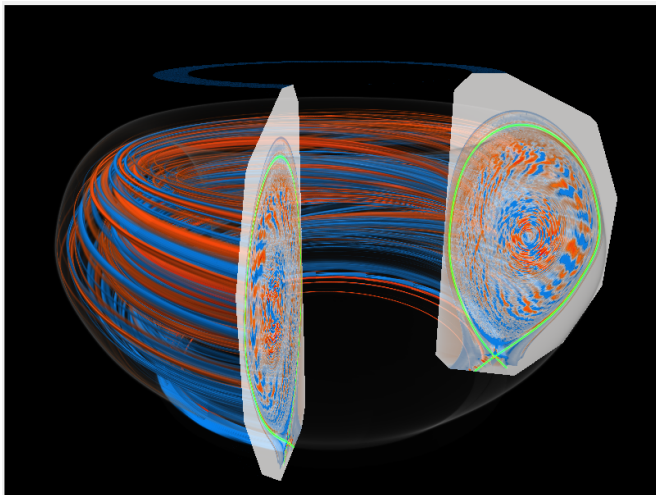
- OpenMC is being developed as part of the ECP ExaSMR project (PIs: Steven Hamilton, Paul Romano)
- OpenMC is a Monte Carlo particle transport code written in C++ and the OpenMP target offloading programming model
- The project seeks to accelerate the design of small modular nuclear reactors by generating virtual reactor simulation datasets with high-fidelity, coupled physics models for reactor phenomena that are truly predictive
- The Monte Carlo method employed by OpenMC is considered the "gold standard" for high-fidelity but these methods suffer from a very high computational cost.
- The extreme performance gains OpenMC has achieved on GPUs is finally bringing within reach a much larger class of problems that historically were deemed too expensive to simulate using Monte Carlo methods.





**ESP Project PI: CS Chang**  
**ECP Project PI: Amitava Bhattacharjee**

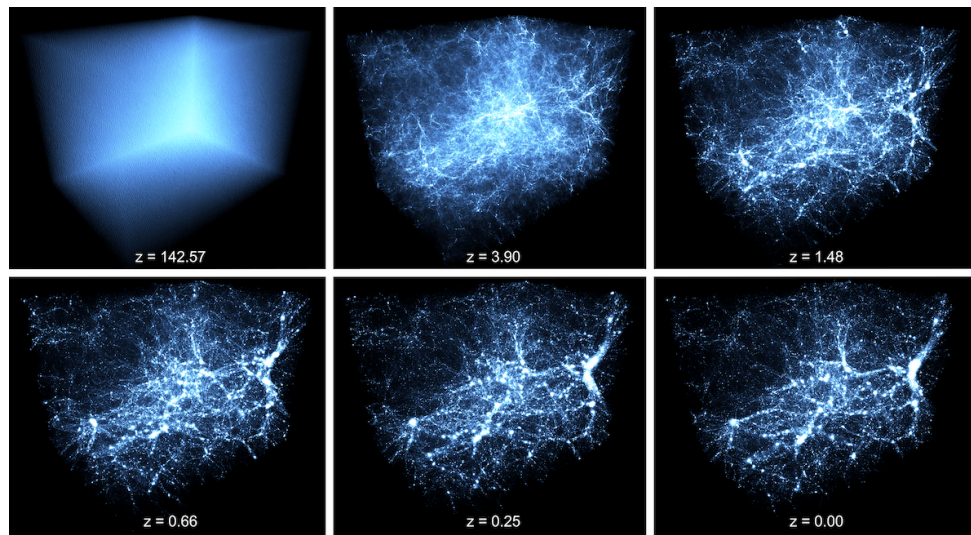
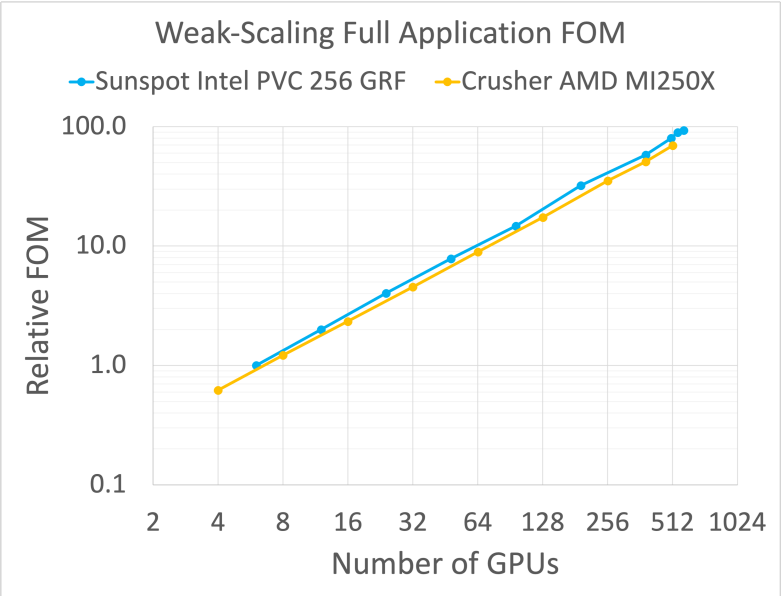
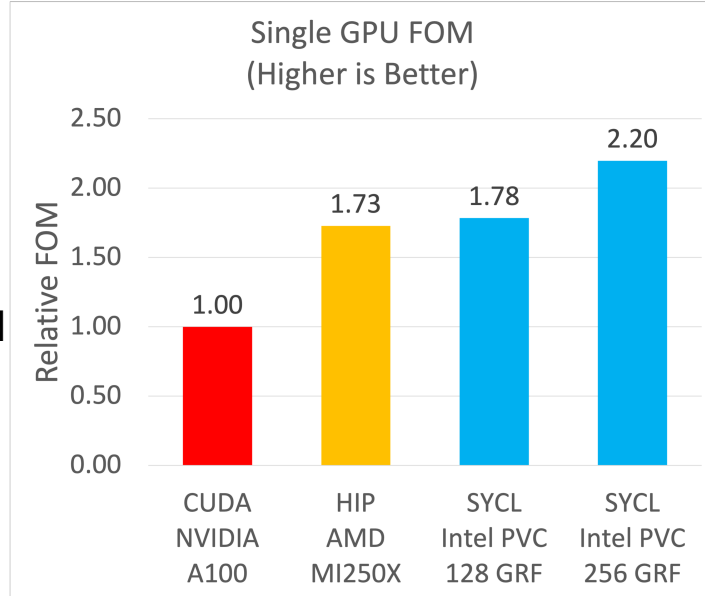
- Science case: Predict ITER fusion reactor plasma behavior with Tungsten impurity ions sputtered from the divertor
- Gyrokinetic particle-in-cell simulation of tokamak plasma using C++ and:
  - Kokkos/SYCL on Intel GPUs
  - Kokkos/HIP on AMD GPUs
  - Kokkos/CUDA on NVIDIA GPUs



# CRK-HACC (courtesy Adrian Pope, Steve Rangel, Nick Frontiere)

**ESP/HACC PI: Katrin Heitmann**  
**ECP/ExaSky PI: Salman Habib**

- CRK-HACC simulates the formation of large-scale structures in the Universe over cosmological time.
- CRK-HACC employs n-body methods for gravity and a novel formulation of Smoothed Particle Hydrodynamics.
- CRK-HACC is a mixed-precision C++ code, with FLOPS-intense sections implemented using architecture-specific programming models in FP32 precision.



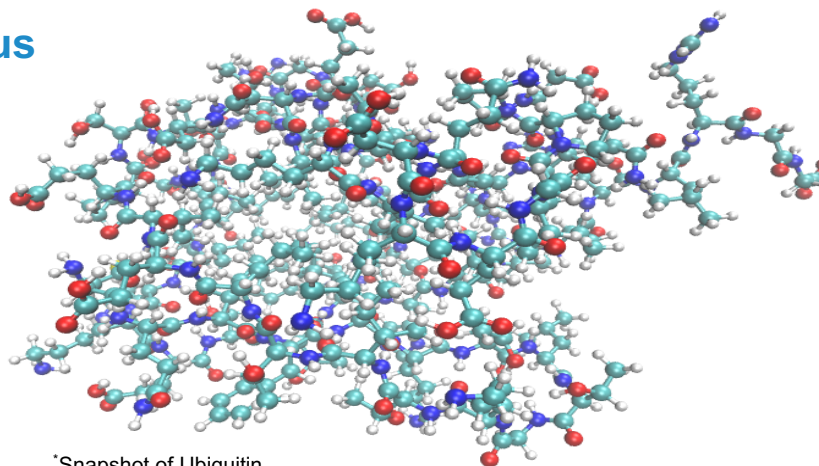
- CUDA and HIP are maintained as a single source with macros.
- SYCL kernels were translated from CUDA using SYCLomatic and custom LLVM-based tools, including optimizations for Intel GPUs.
- Figure-of-Merit (FOM) has units of particle-steps per second.
- Single GPU FOM problem used 33 million particles per GPU, and Intel PVC results are shown for both small (128) and large (256) General-purpose Register File (GRF) modes.
- Weak-scaling results are shown with the full application FOM, where the GPU represents roughly 80% of the total wall clock.

# NWChemEx (Courtesy of Ajay Panyala)

<https://github.com/NWChemEx-Project>

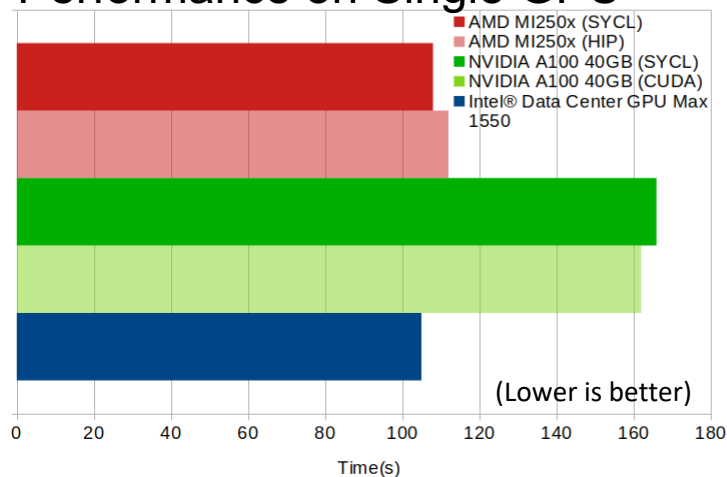
ESP & Project Project PI: Theresa Windus

- NWChemEx is a general purpose electronic structure code, which includes
  - Array of high-fidelity coupled cluster methods
  - Hartree-Fock, DFT, MP2 methods
  - Reduced-scaling DLPNO formulation
  - Molecular dynamics
- Programming models: C++, CUDA, HIP, SYCL
  - Communication frameworks: Global Arrays, UPC++, MADNESS
  - Tensor Contraction Engines: TAMM, TiledArray
- Key physics modules
  - DLPNO-CCSD(T)
    - Reduced-scaling implementation for GPU platforms



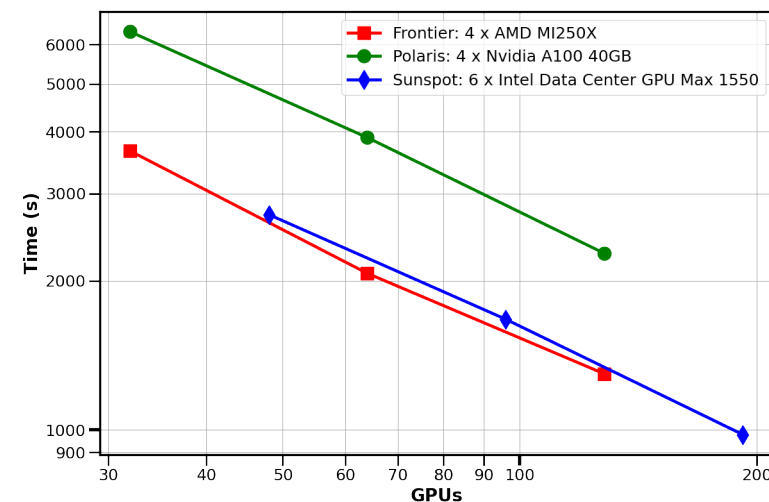
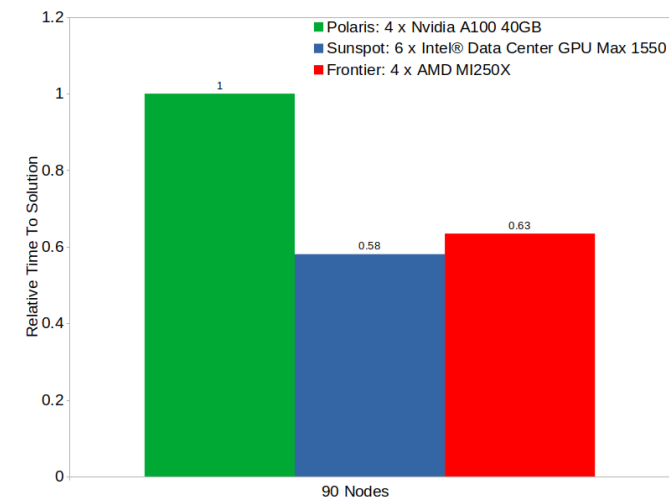
\*Snapshot of Ubiquitin Protein

## Performance on Single GPU



- Single GPU, Time in seconds for DLPNO-CCSD per iteration
- Performance of SYCL on NVIDIA & AMD were comparable with native CUDA & HIP respectively

## Strong Scaling Performance on 90-nodes



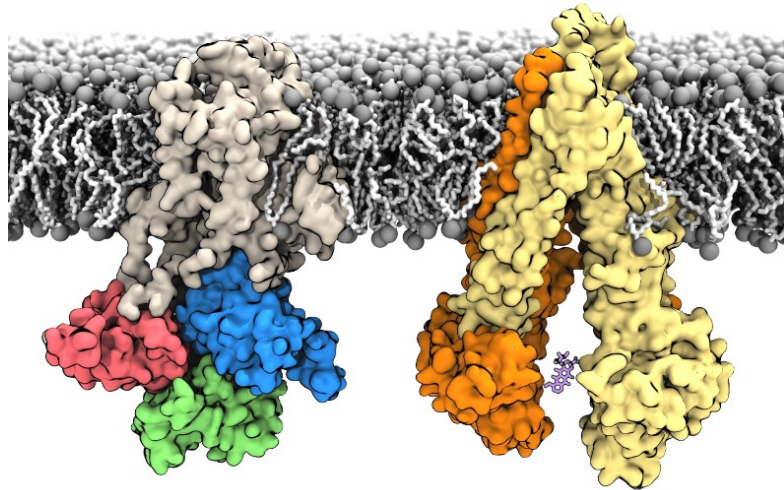
- Acknowledgment: Work performed by the NWChemEx team members without any architecture specific optimizations

# NAMD 2 (Courtesy of Wei Jiang)

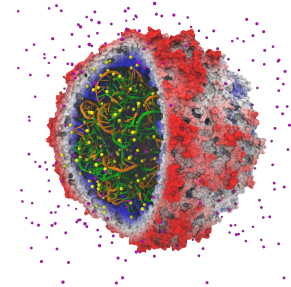
## Scalable molecular dynamics for exascale computations

### ESP PI: Benoit Roux

- Simulate large biomolecular systems or complex macromolecular machines
- Science problem: molecular structure-function relationship
- Algorithm: particle motion integration with short- and long-range force calculation
- Fine-grained force-domain decomposition
- Written using C/C++, Charm++, CUDA, HIP, SYCL



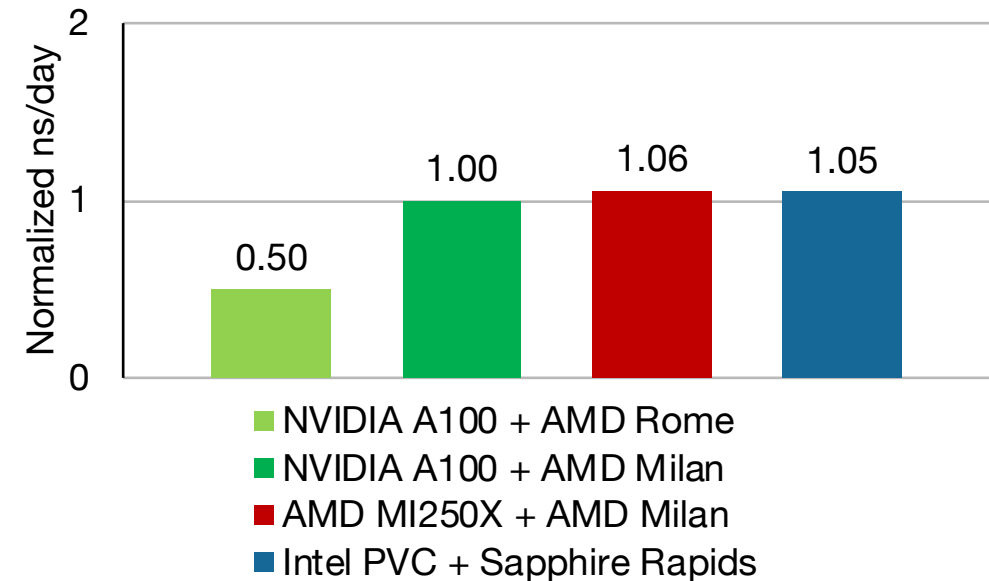
NAMD website: <https://www.ks.uiuc.edu/Research/namd>



Benchmarking NAMD 2.15alpha2 on STMV (1.06M atoms) NVE simulation: CHARMM force field (12A cutoff), rigid bonds with 2 fs timestep, multiple time stepping with 4 fs PME

(Higher is better)

### Single-GPU Results



*NAMD GPU-offload performance depends on both GPU and CPU performance together with host-device latency and bandwidth*

**THANK YOU**

# EXTRA SLIDES



# Aurora Exascale Architecture

ExaComm ISC 2023

**Kalyan Kumaran**

Director of Technology at the Argonne Leadership Computing Facility (ALCF)  
Argonne National Laboratory

**25 May 2023**

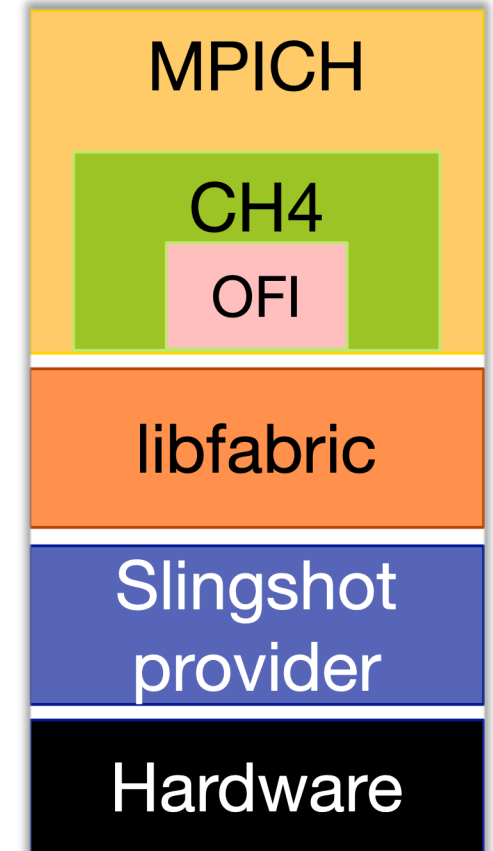
# DOE Leadership Computing Facility

- Established in 2004 as a collaborative, multi-lab initiative funded by DOE's *Advanced Scientific Computing Research* program
- Operates as **one facility** with two centers, at Argonne and at Oak Ridge National Laboratory
- Deploys and operates at least two advanced architectures that are **10-100 times more powerful** than systems typically available for open scientific research
- **Fully dedicated** to open science to address the ever-growing needs of the scientific community



# MPI

- Intel MPI & Cray MPI
  - MPI 3.0 standard compliant
- The MPI library will be thread safe
  - Allow applications to use MPI from individual threads
  - Efficient MPI\_THREAD\_MULTIPLE (locking optimizations)
- Asynchronous progress in all types of nonblocking communication
  - Nonblocking send-receive and collectives
  - One-sided operations
- Hardware and topology optimized collective implementations
- Supports MPI tools interface



# Aurora System Overview

## COMPUTE

Total Cabinets (#)	166
Total Nodes (#)	10,624
Total Intel Data Center GPU Max Series (#)	63,744
Total Intel Xeon Max Series w HBM (#)	21,248
Total DP Peak Flops (EF)	$\geq 2.0$

## MEMORY

Total GPU HBM Memory (PB)	8.16
Peak GPU HBM BW (PB/s)	208.9
Total Xeon HBM Memory (PB)	1.36
Peak Xeon HBM BW (PB/s)	30.5
Total DDR5 Memory (PB)	10.9
Peak Xeon DDR5 BW (PB/s)	5.95

## FABRIC

Fabric (name)	Slingshot11
NIC	HPE SS – 200Gbps
Switch	HPE SS 64 ports @ 200Gbps
Fabric Topology (name)	Dragonfly
Peak injection BW (PB/s)	2.12
Peak Bi-section BW (PB/s)	0.69

## STORAGE

Distributed Asynchronous Object Store (DAOS) (PB) (usable)	230
DAOS Performance (TB/s)	31
DAOS Nodes (#)	1024
Lustre GFS (PB)	150
Lustre Performance (TB/s)	1

# Aurora Compilers

Intel compilers will be the primary compilers on Aurora

- Provide C, C++, Fortran
- Optimized for Intel Data Center GPU Max Series
- Implemented using LLVM infrastructure
- Intel will upstream LLVM GPU backend and other related improvements
  - Enables 3<sup>rd</sup> party compiler development

## LLVM

- Clang – C/C++
- Flang – Fortran
- Will generate code for Intel Data Center GPU Max Series
- Will incorporate ECP optimization enhancements

## Cray

- Optimized C, C++, Fortran compilers for x86
- Utilizes LLVM components

# Libraries

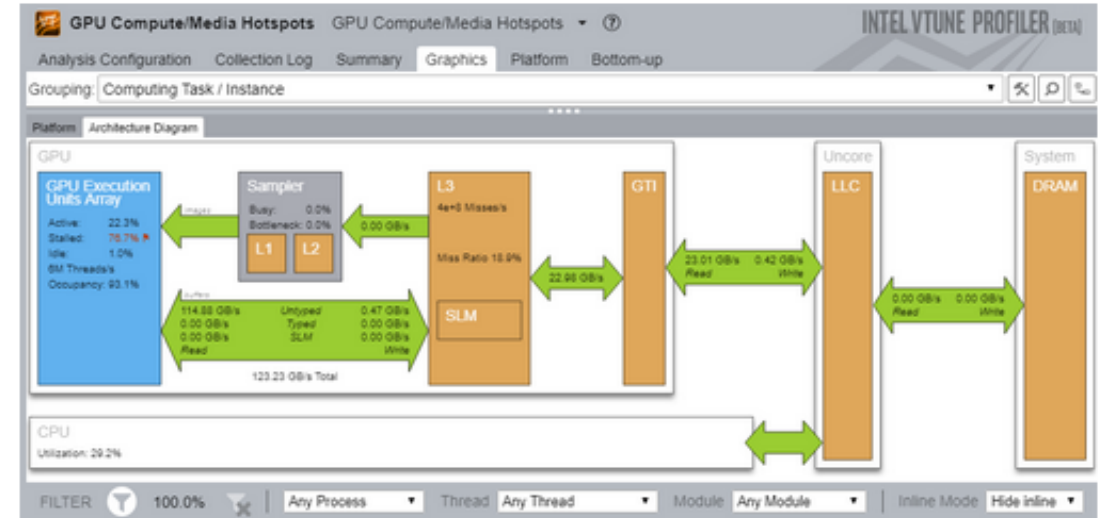
- oneMKL
  - Optimized math libraries
- oneDAL
  - Optimized Data analytics
- oneCCL
  - Optimized communications for deep learning models
- oneDNN
  - Optimized deep learning routines
- Python
  - Tensorflow
  - PyTorch
  - Numba
  - Scikit-learn
- Apache Spark MLlib



# Tools

## VTune Profiler

- Widely used performance analysis tool
- Currently supports analysis on Intel integrated GPUs
- Will support future 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

APIs and Tools for hardware counters and binary instrumentation for performance measurement

# Data Science and Learning on Aurora

Aurora will provide for a familiar, productive and performant HPC and AI software stack

Intel AI Analytics Toolkit

- Python and Productivity Languages
  - Numba, NumPy, etc.
  - JAX and Julia
- Deep Learning Frameworks:
  - PyTorch, TensorFlow, Horovod, DDP, Deepspeed
- Machine Learning
  - OneDAL, scikit-learn, XGBoost, etc.
- Optimized and scalable communication using OneCCL
- Spark BigData Analytics stack
- Profiling and debugging tools

Intel oneAPI

USA (English) Sign In

## Intel® AI Analytics Toolkit

Achieve End-to-End Performance for AI Workloads

oneAPI POWERED

Features What's Included Documentation & Code Samples Training Specifications Help

### Accelerate Data Science & AI Pipelines

The Intel® AI Analytics Toolkit gives data scientists, AI developers, and researchers familiar Python\* tools and frameworks to accelerate end-to-end data science and analytics pipelines on Intel® architectures. The components are built using oneAPI libraries for low-level compute optimizations. This toolkit maximizes performance from preprocessing through machine learning, and provides interoperability for efficient model development.

Using this toolkit, you can:

- Deliver high-performance deep learning (DL) training on Intel® XPUs and integrate fast inference into your AI development workflow with Intel-optimized DL frameworks: TensorFlow\* and PyTorch\*, pretrained models, and low-precision tools.
- Achieve drop-in acceleration for data preprocessing and machine learning workflows with compute-intensive Python\* packages: Modin\*, scikit-learn\*, and XGBoost\* optimized for Intel.
- Gain direct access to Intel analytics and AI optimizations to ensure that your software works together seamlessly.

Get Access

### Develop in the Cloud

Get what you need to build, test, and optimize your oneAPI projects for free. With an Intel® DevCloud account, you get 120 days of access to the latest Intel® hardware—CPUs, GPUs, FPGAs—and Intel oneAPI tools and frameworks. No software downloads. No configuration steps. No installations.

Get It Now

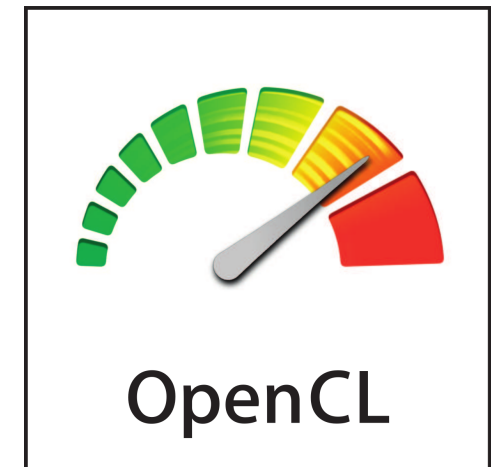
### Download the Toolkit

Get It Now

<https://software.intel.com/content/www/us/en/develop/tools/oneapi/ai-analytics-toolkit.html>

# OpenCL

- Open standard for heterogeneous device programming (CPU, GPU, FPGA)
- Standardized by multi-vendor Khronos Group, V 1.0 released in 2009
  - AMD, Intel, nVidia, ...
  - Many implementations from different vendors
- Intel implementation for GPU is Open Source (<https://github.com/intel/compute-runtime>)
- SIMT programming model
  - Distributes work by abstracting loops and assigning work to threads
  - Not using pragmas / directives for specifying parallelism
- Similar model to CUDA
- Consists of a C compliant library with kernel language
  - Kernel language extends C
  - Has extensions that may be vendor specific



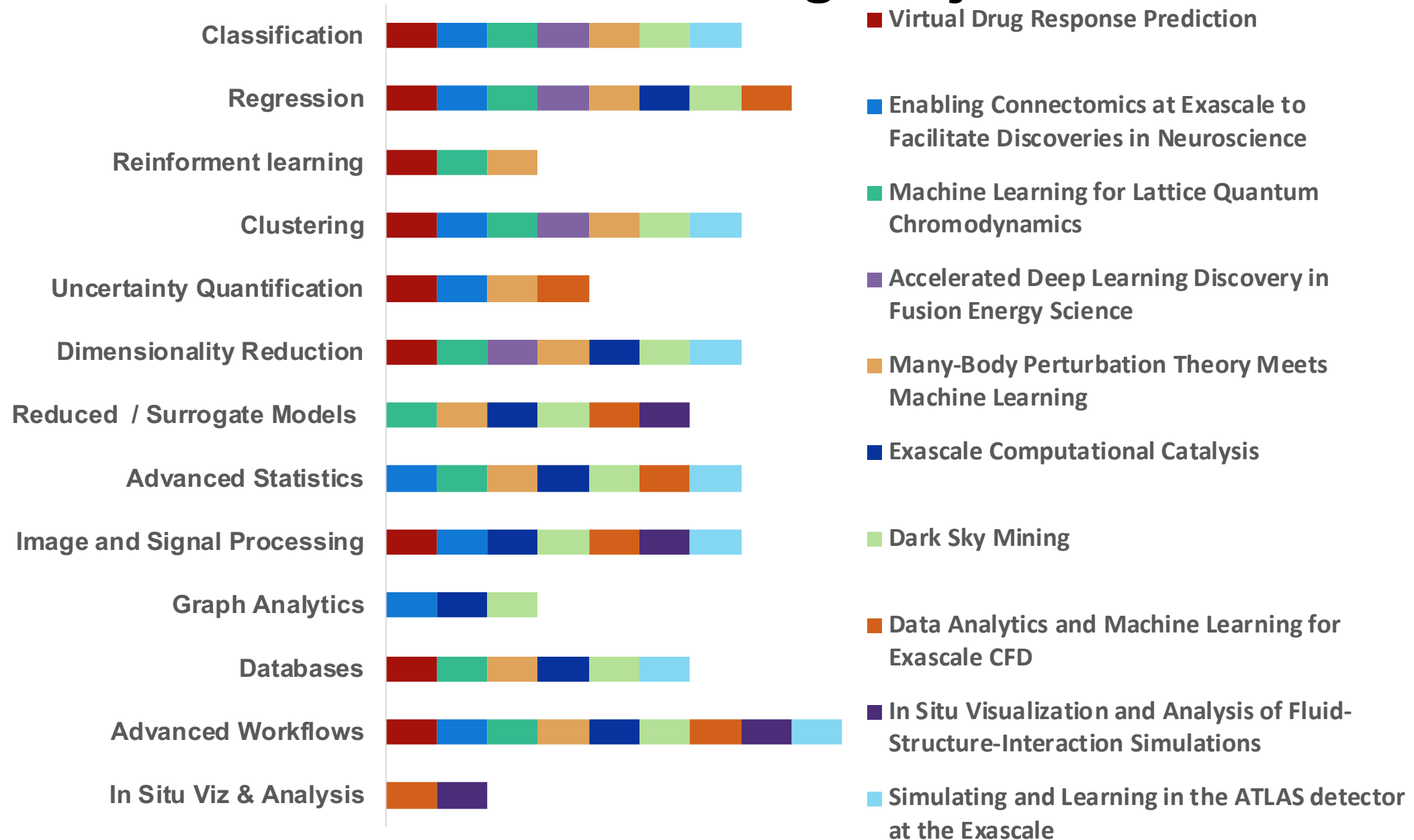
# OpenMP 4.5/5

- OpenMP 5 constructs will provide directives based programming model for Intel GPUs
- Available for C, C++, and Fortran
- A portable model expected to be supported on a variety of platforms (Aurora, Frontier, Perlmutter, ...)
- Optimized for Aurora
- Integration with MKL for GPU offload
- For Aurora OpenACC codes could be converted into OpenMP
  - Automated translation possible through the clacc conversion tool (for C/C++)



<https://www.openmp.org/>

# Aurora ESP Data and Learning Projects and Methods



Learning

Data



# Bridging ESP Projects to Aurora

- To be ready for Early Science runs, projects must
  - Demonstrate INCITE level computational readiness (scaling, use GPUs, ready proposed problem in short order)
  - Complete model validations, preliminary studies, parameter-setting exercises
  - Finish integrating complex workflows for Data and Learning projects with realistic data
- Portability of applications, components, and workflows to Polaris

## Simulation components

- OpenMP 4.5+
- Kokkos
- SYCL
- PETSc, math libraries
- *Many apps have explicit NVIDIA implementations*

## Data components

- Spark
- HDF5
- ADIOS
- MPI-IO
- Databases
- Numba
- Python

## Learning components

- TensorFlow
- PyTorch
- Distributed DL (eg., Horovod)
- Scikit Learn
- JAX
- Julia

## Workflows

- Containers
- Balsam
- funcX/Parsl
- Python-based workflows