Computational Performance Workshop, May 5-7, 2020

Aurora

Argonne's First Exascale System

Kris Rowe
Argonne Leadership Computing Facility

Tim Williams
Deputy Director, Computational Science Division

Outline

Hardware

- System Overview
- Compute & Memory
- Network
- Storage
- Aurora Testbed

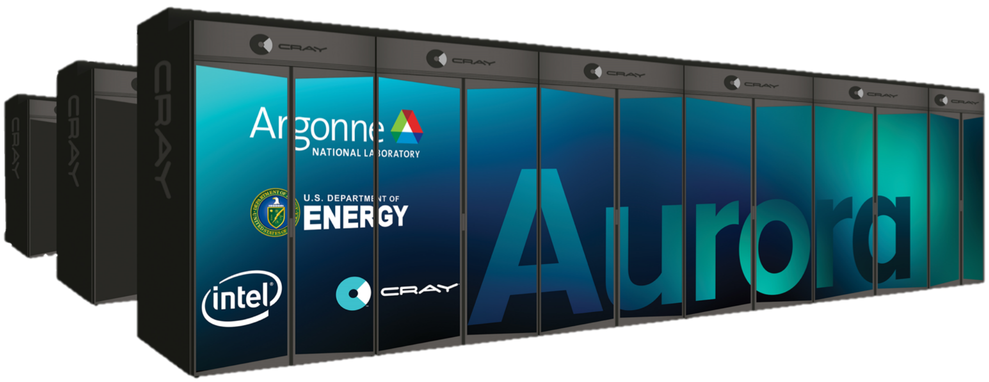
Software

- Three pillars
- Programming models
- Tools and libraries
- Development platforms
- Support for ECP teams

Further details on Aurora can be discussed under NDA

If you are not covered by an Aurora NDA please talk to your project PI

HARDWARE



Arriving at Argonne in 2021

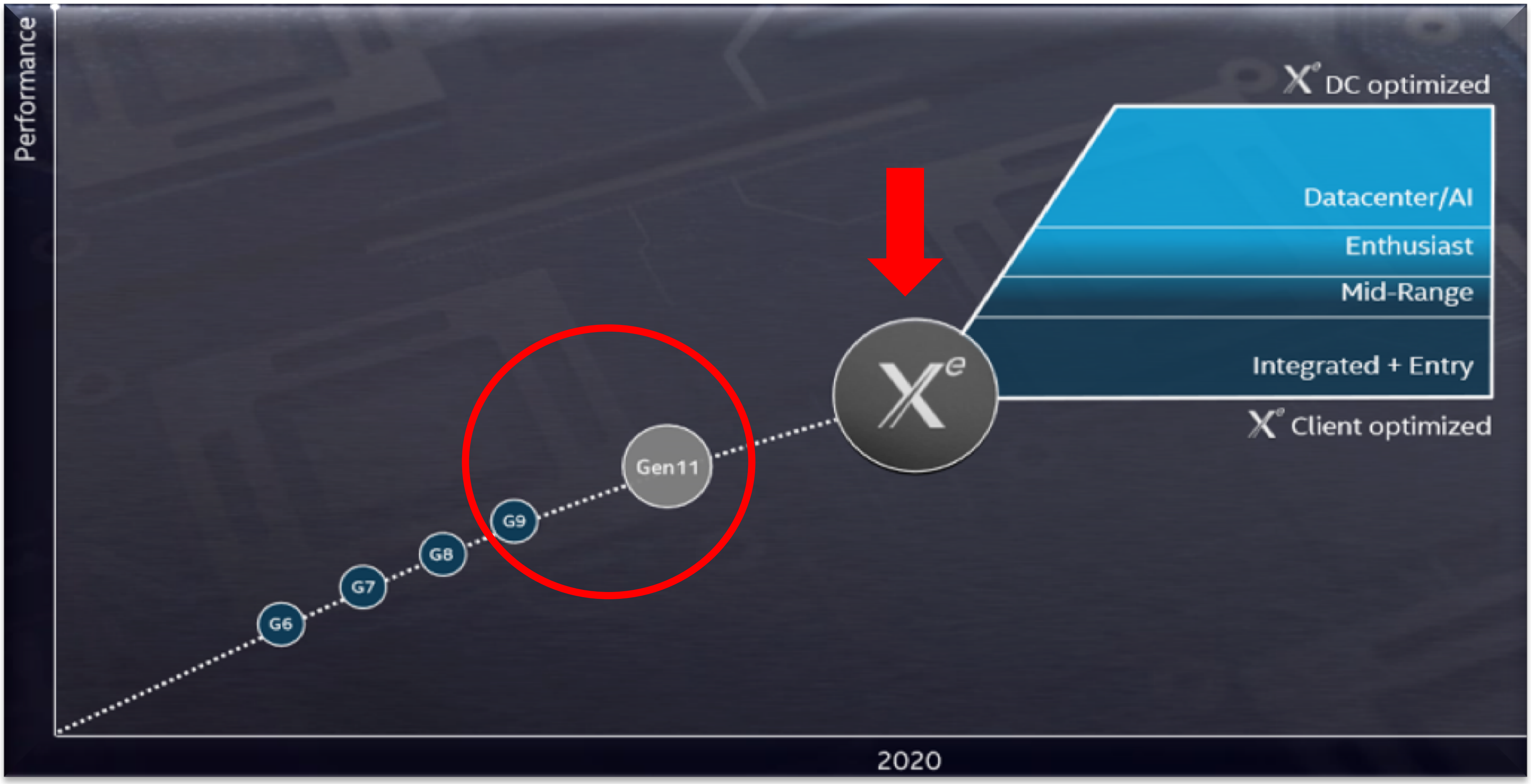
Performance
- > 1 Exaflops Sustained

Compute & Memory
- Intel Xeon Sapphire Rapids CPUs
- Intel Xe Ponte Vecchio GPUs
- >10 PB Total Memory

Network
- Cray Shasta Cabinets
- Cray Slingshot Fabric

Storage
- *Data Asynchronous Object Storage (DAOS)*
  - > 230 PB Capacity
  - > 25 TB/s Bandwidth
- *Lustre*
  - > 150 PB Capacity
  - ~ 1TB/s Bandwidth

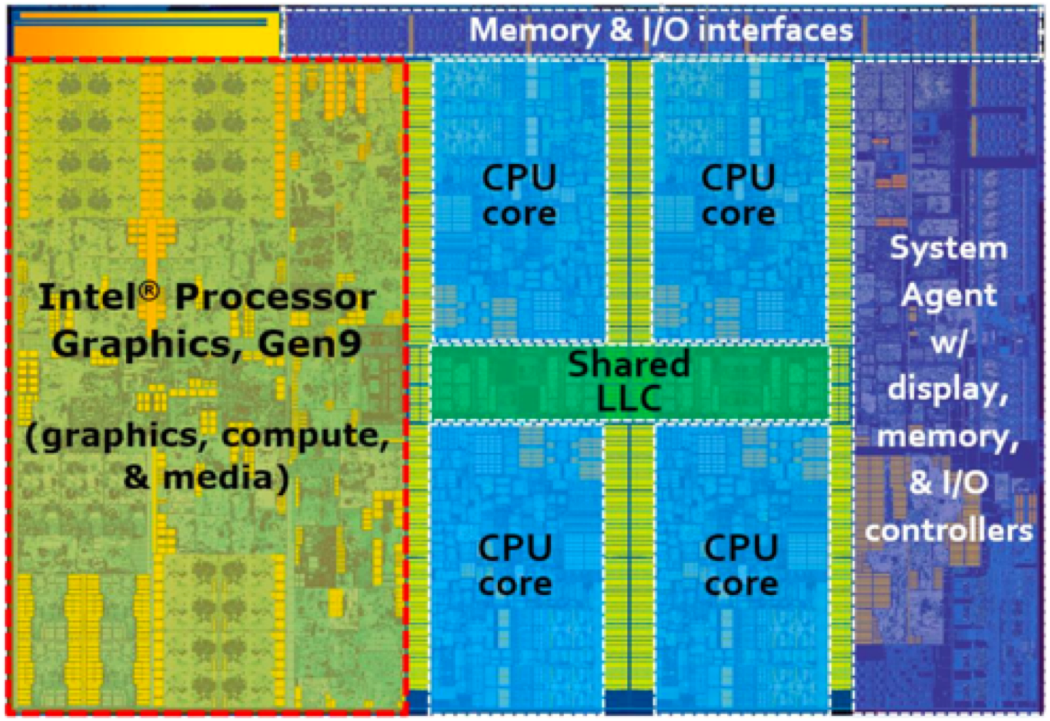
The Evolution of Intel GPUs



Source: Intel

Intel GPUs

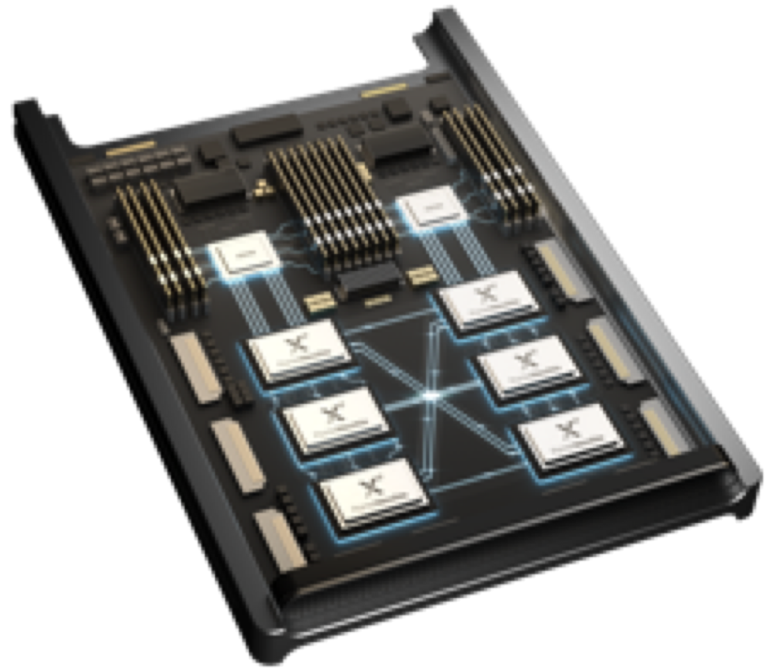
- Recent integrated generations:
  - Gen9 – used in Skylake
  - Gen11 – used in Ice Lake
- Gen9
  - 100-300 Gflops (DP) Peak Performance
  - Low power by design
- Upcoming X^e GPU
  - Discrete



Layout of Architecture components for an Intel Core i7 processor 6700K for desktop systems (91 W TDP, 122 mm)

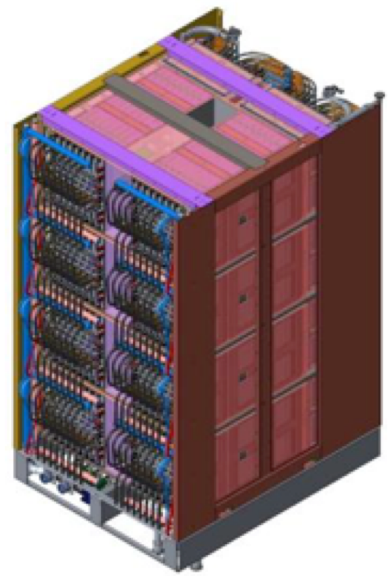
Aurora Compute Node

- 2 Intel Xeon Sapphire Rapids CPUs
- 6 Intel X^e Ponte Vecchio GPUs
  - All to all connection
  - Low latency and high bandwidth
- 8 Slingshot Fabric endpoints
- Unified Memory Architecture across CPUs and GPUs



Cray Shasta System

- Supports diversity of processors
- Scale-optimized cabinets for
  - Density
  - Cooling
  - High network bandwidth
- Standard cabinets for flexibility
- Cray SW stack with improved modularity
- Unified by a single, high-performance interconnect



Optimized Rack

Standard Rack

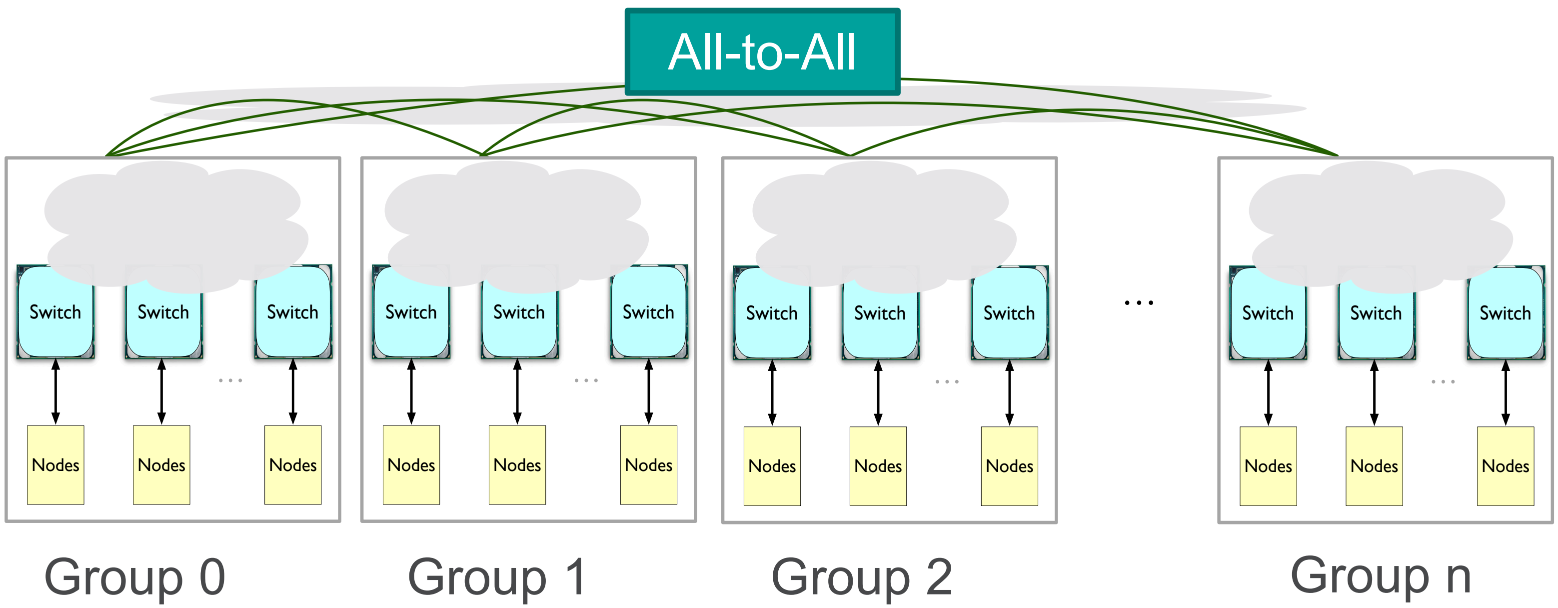
Cray Slingshot Network

- Slingshot is the next generation scalable interconnect by Cray
  - 8th major generation
- Builds on Cray's expertise in high performance network following
  - Gemini (Titan, Blue Waters)
  - Aries (Theta, Cori) – 5 hop dragonfly topology
- Slingshot introduces:
  - Congestion management
  - Traffic classes
  - 3 hop dragonfly

https://www.cray.com/products/computing/slingshot
https://www.cray.com/resources/slingshot-interconnect-for-exascale-era

Dragonfly Topology

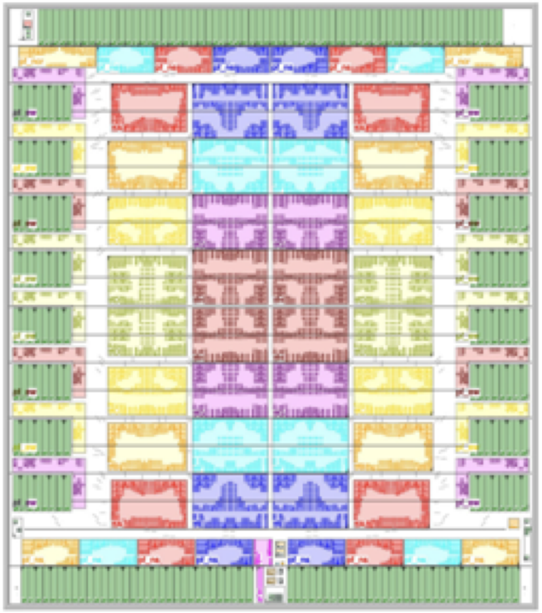
- Several groups are connected together in an all-to-all topology.
- Optical cables are used for the long links between groups.
- Low-cost electrical links are used to connect the NICs in each node to their local router and the routers in a group



https://www.cray.com/sites/default/files/resources/CrayXCNetwork.pdf

High Bandwidth Switch: Rosetta

- Multi-level congestion management
  - To minimize the impact of congested applications on others
  - Very low average and tail latency
- Quality of Service (QoS) – Traffic Classes
  - Class: *Collection of buffers, queues and bandwidth*
  - Intended to provide isolation between applications via traffic shaping
- Aggressive adaptive routing
  - Expected to be more effective for 3-hop dragonfly due to closer congestion information
- High performance multicast and reductions



Rosetta Switch

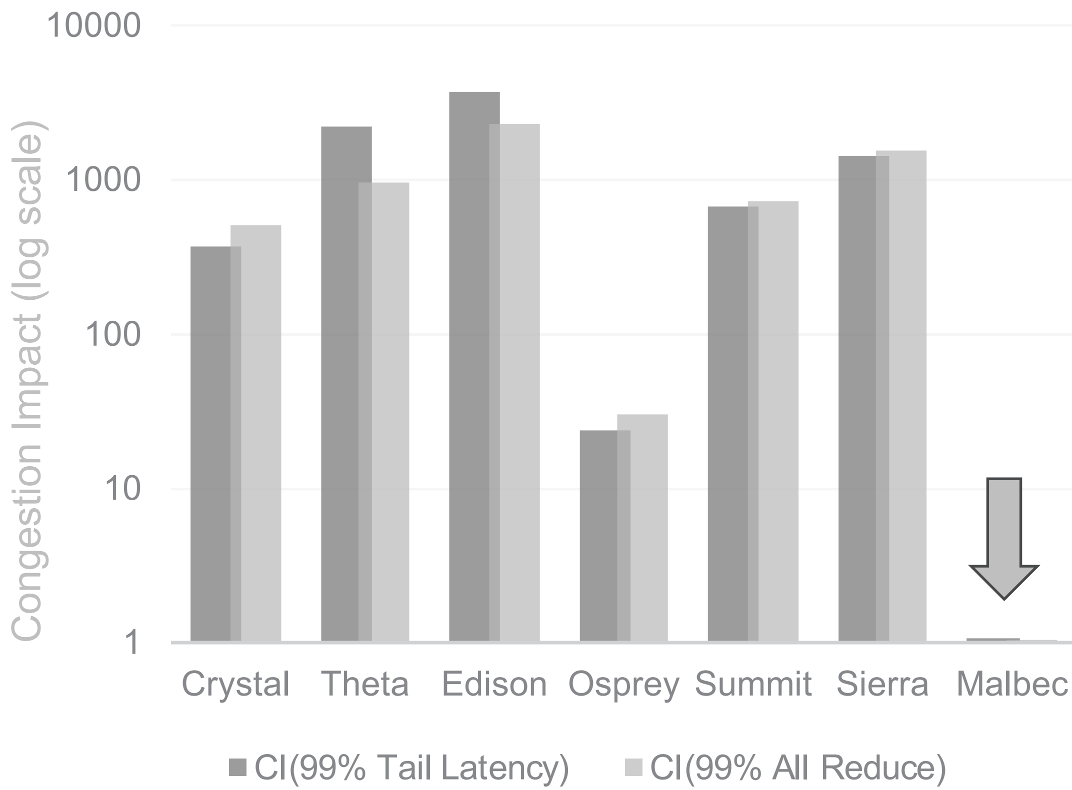
25.6 Tb/s per switch, from 64 - 200 Gbs ports (25GB/s per direction)

GPCNeT (Global Performance and Congestion Network Tests)

- A new benchmark developed by Cray in collaboration with Argonne and NERSC
- Publicly available at https://github.com/netbench/GPCNET
- Goals:
  - Proxy for real-world application communication patterns
  - Measure network performance **under load**
  - Look at both **mean** and **tail latency**
  - Look at interference between workloads
  - What is the **Congestion management**?
    - How well the system performs under congestion

Congestion Management with Slingshot

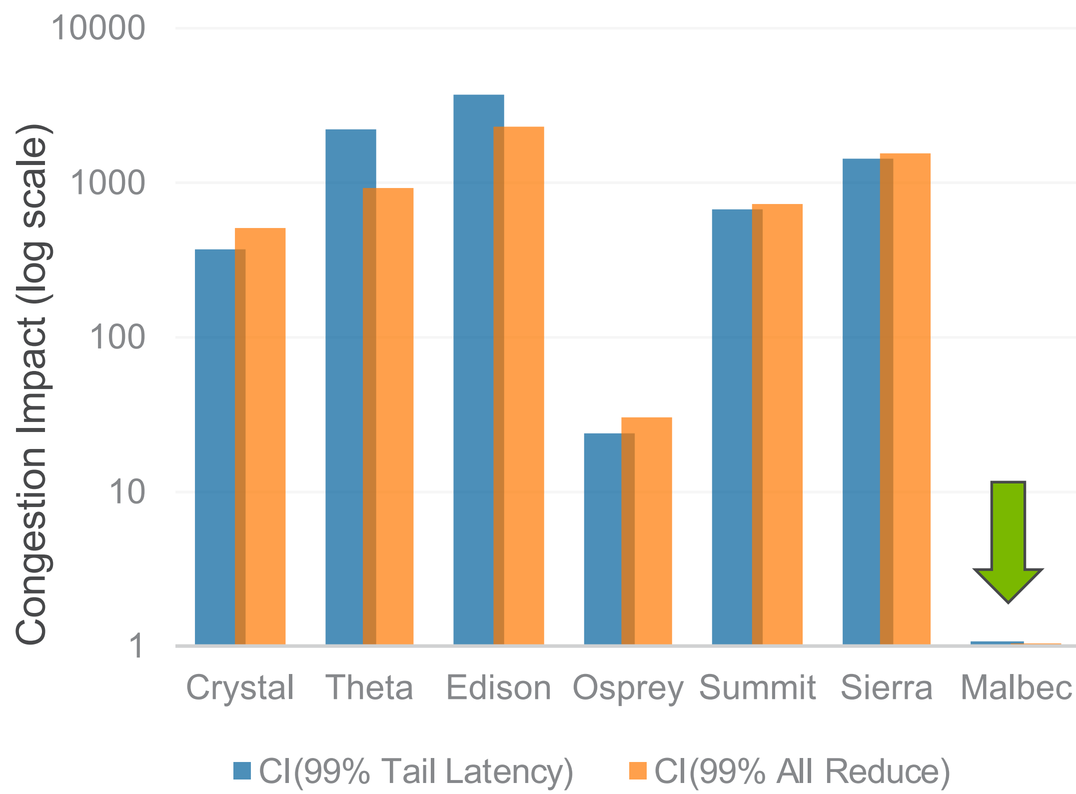
- GPCNeT benchmark suit was utilized
- Tail latency was measured at:
  - congested
  - isolated
- Benchmark was run over the full machine for multiple runs
- Congestion impact:
  - Latency $_{congested}$/Latency $_{isolated}$
- Aries congestion impact is in order of 1000x
- InfiniBand has reduced congestion impact (20-1000x)
- Slingshot testbed (Malbec) has minimal congestion impact



https://dl.acm.org/doi/pdf/10.1145/3295500.3356215

Congestion Management with Slingshot

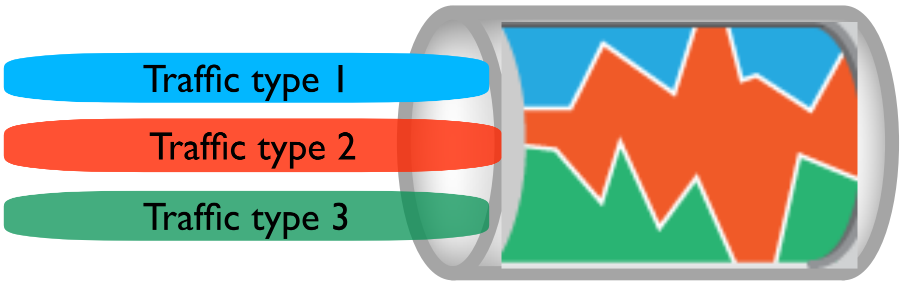
- GPCNeT benchmark suit was utilized
- Tail latency was measured at:
  - congested
  - isolated
- Benchmark was run over the full machine for multiple runs
- Congestion impact:
  - $\text{Latency}_{\text{congested}}/\text{Latency}_{\text{isolated}}$
- Aries congestion impact is in order of 1000x
- InfiniBand has reduced congestion impact (20-1000x)
- Slingshot testbed (Malbec) has minimal congestion impact



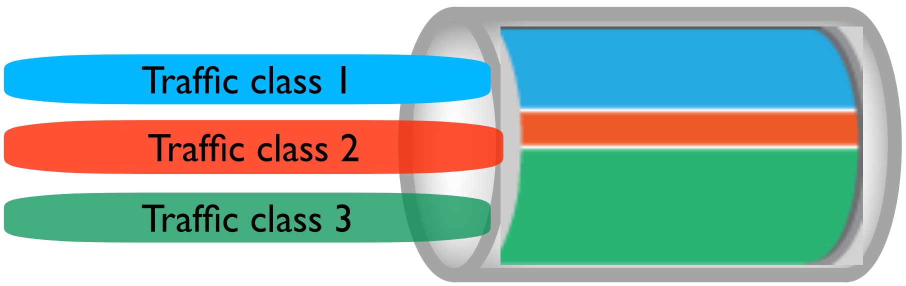
https://dl.acm.org/doi/pdf/10.1145/3295500.3356215

Slingshot Quality of Service Classes

- Highly tunable QoS classes
- Supports multiple, overlaid virtual networks …
- Jobs can use multiple traffic classes
- Provides performance isolation for different types of traffic



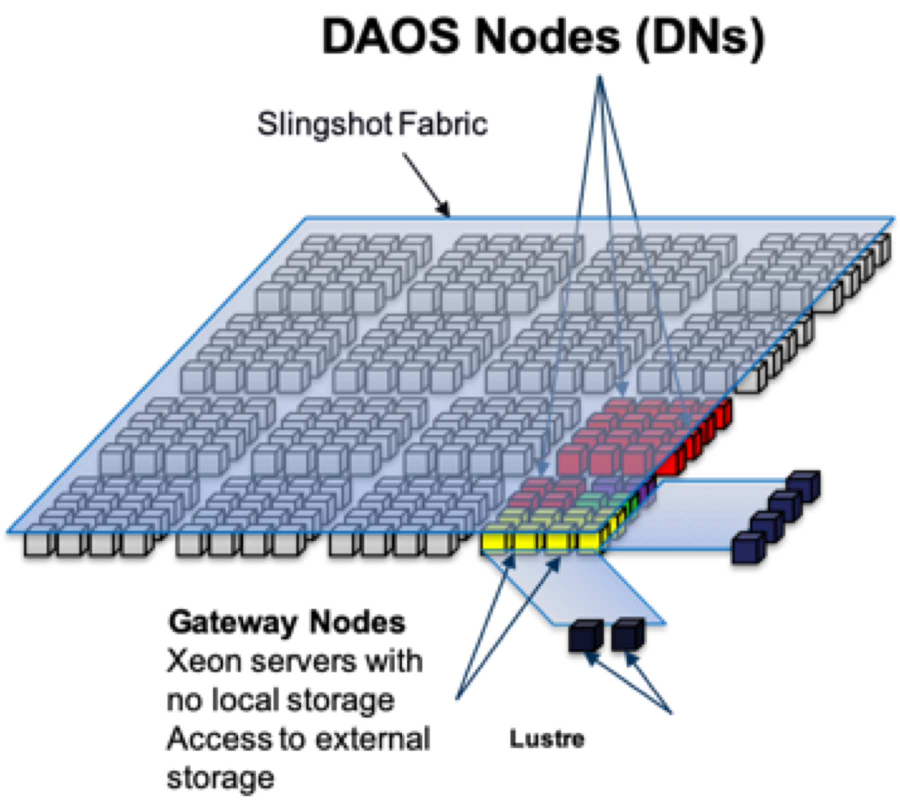
Bandwidth sharing without QoS



Bandwidth sharing with QoS

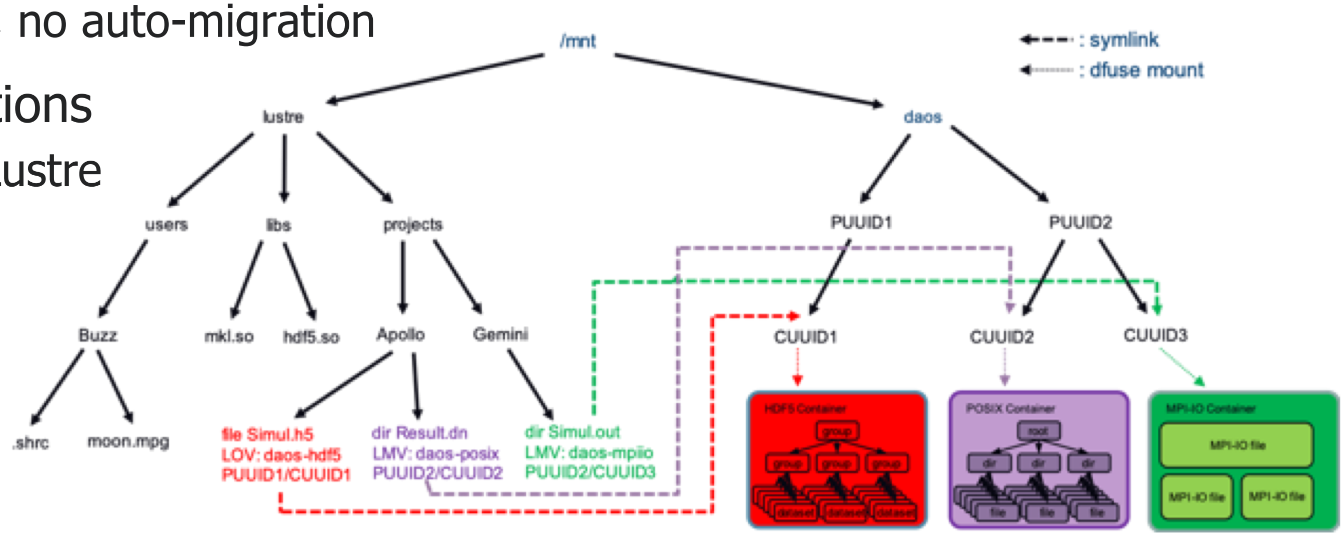
Distributed Asynchronous Object Store (DAOS)

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

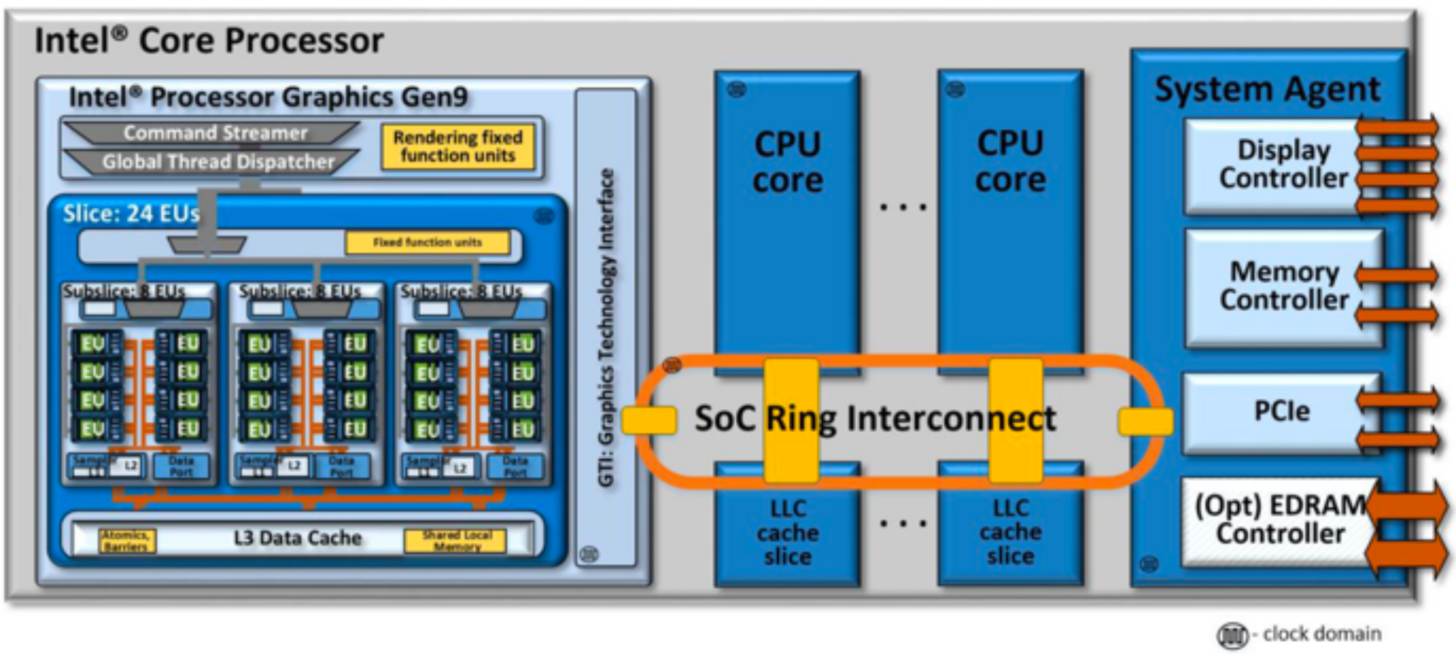


User Environment

- User sees a single storage namespace
  - This is in Lustre Compatibility with legacy systems
  - Links point to DAOS containers within the /project directory
  - DAOS aware software interpret these links and access the DAOS containers
- Data resides in a single place (Lustre or DAOS)
  - Explicit data movement, no auto-migration
- Suggested storage locations
  - Source and binaries in Lustre
  - Bulk data in DAOS



Intel Gen9 Integrated Graphics



- CPUs and GPUs are integrated on same chip:
  - GPU and CPU same memory
  - GPU, cores, and memory are connected by an internal ring interconnect
  - Shared Last Level Cache

Gen9 (GT4) GPU Characteristics

Characteristics	Value	Notes
Clock Freq.	1.15 GHz	
Slices	3	
EUs	72	3 slice * 3 sub-slices * 8 EUs
Hardware Threads	504	72 EUs * 7 threads
Concurrent Kernel Instances	16,128	504 threads * SIMD-32
L3 Data Cache Size	1.5 MB	3 slices * 0.5 MB/slice
Max Shared Local Memory	576 KB	3 slice * 3 sub-slices * 64 KB/sub-slice
Last Level Cache Size	8 MB	
eDRAM size	128 MB	
32b float FLOPS	1152 FLOPS/cycle	72 EUs * 2 FPUs * SIMD-4 * (MUL + ADD)
64b float FLOPS	288 FLOPS/cycle	72 EUs * 1 FPU * SIMD-2 * (MUL + ADD)
32b integer IOPS	576 IOPS/cycle	72 EUs * 2 FPUs * SIMD-4

331.2 DP GFlops

Summary of Aurora Hardware Architecture

- Intel/Cray machine arriving at Argonne in 2021
  - Sustained performance > 1 Exaflops
- Aurora node architecture
  - 2 Intel Xeon Sapphire Rapids CPUs
  - 6 Intel X^e Pointe Vecchio GPUs
  - 8 Slingshot Fabric endpoints
  - Unified Memory Architecture across CPUs and GPUs
- Cray Sling Shot Network and Shasta platform
  - Highly tunable congestion management, traffic classes
- I/O
  - DAOS (≥ 230 PB capacity and ≥ 25 TB/s)
  - Lustre (150 PB of storage and ~ 1TB/s)

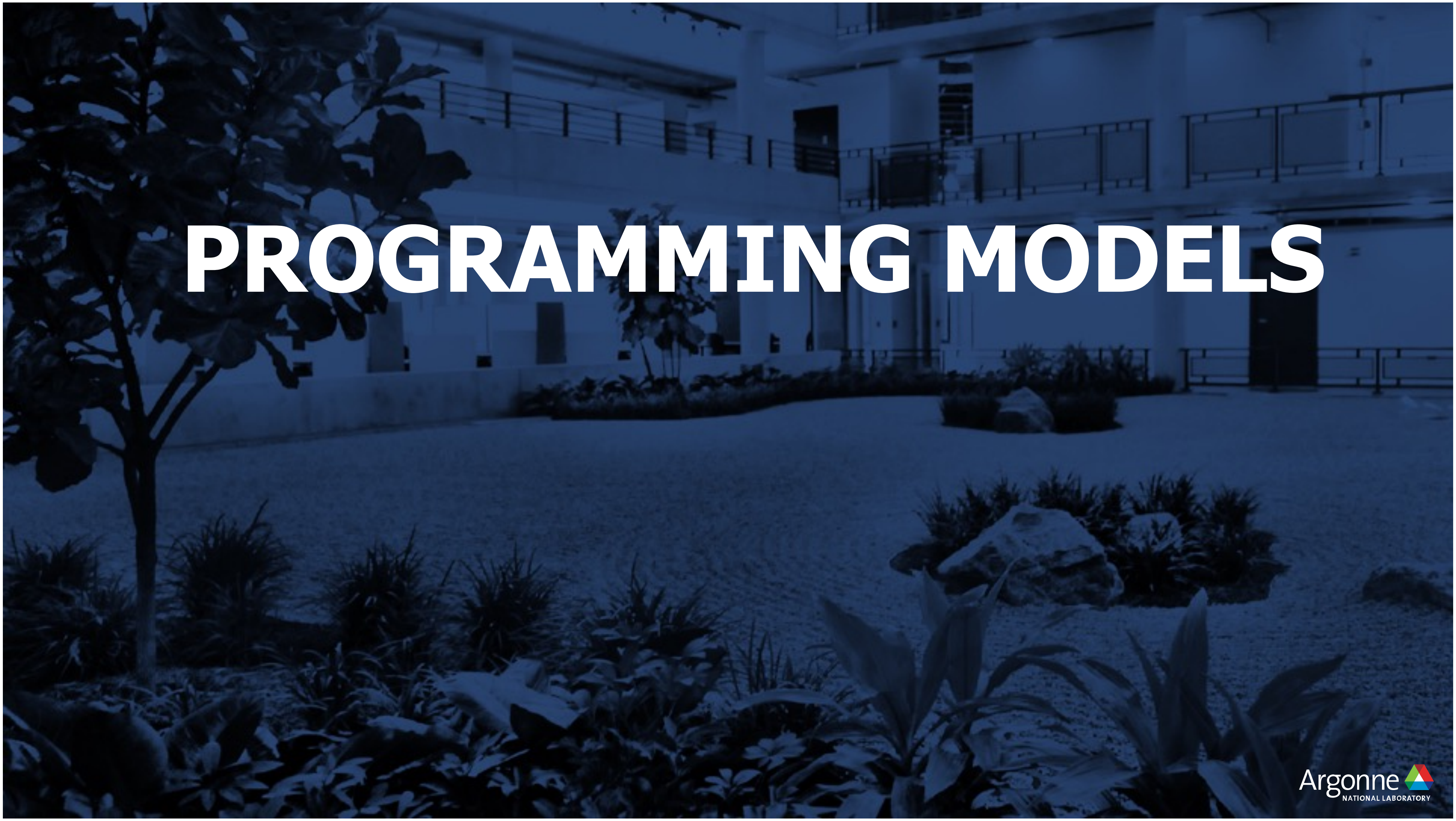
SOFTWARE

Three Pillars

Simulation	Data	Learning
HPC Languages	Productivity Languages	Productivity Languages
Directives	Big Data Stack	DL Frameworks
Parallel Runtimes	Statistical Libraries	Statistical Libraries
Solver Libraries	Databases	Linear Algebra Libraries

- Compilers, Performance Tools, Debuggers
- Math Libraries, C++ Standard Library, libc
- I/O, Messaging
- Containers, Visualization
- Scheduler
- Linux Kernel, POSIX

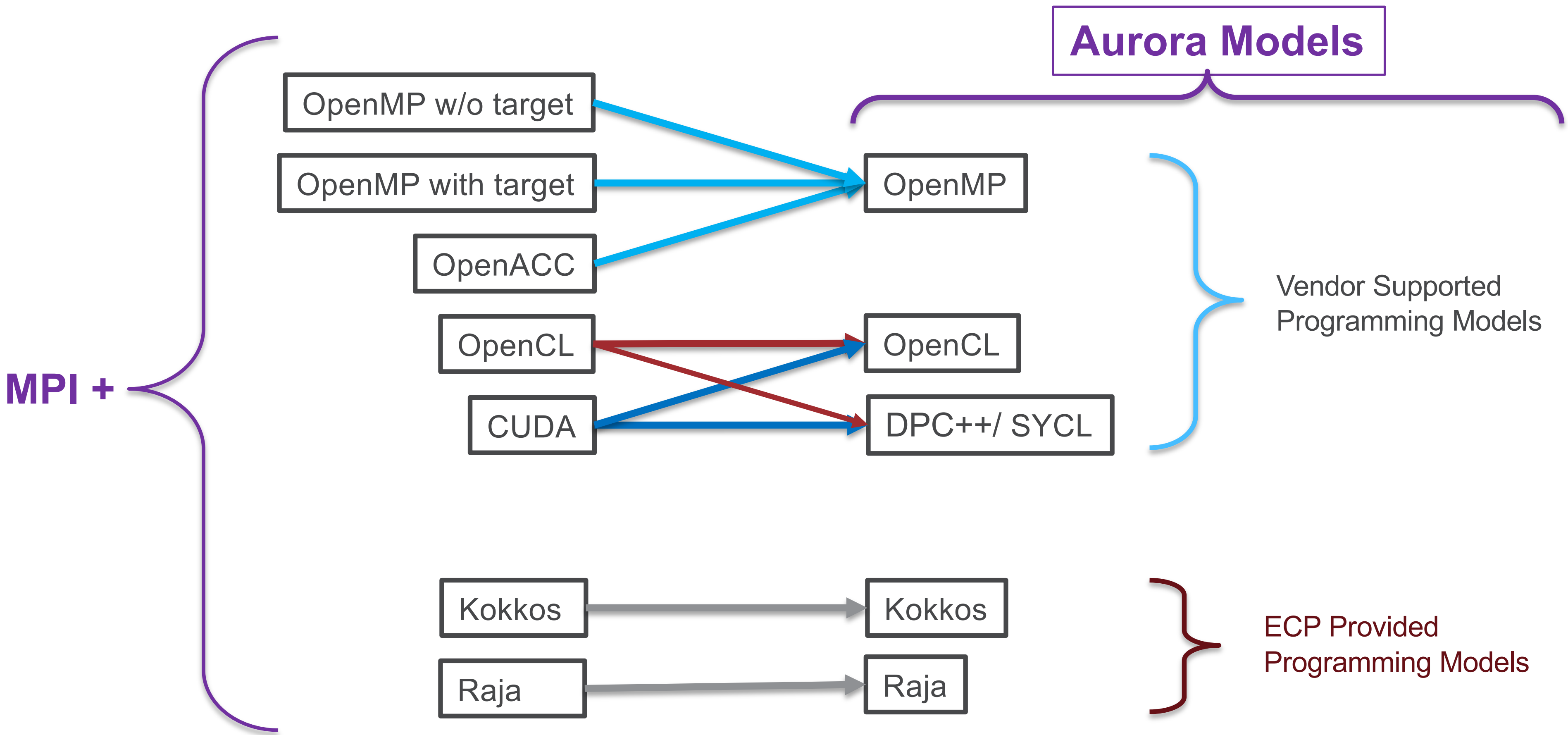
PROGRAMMING MODELS



Available Aurora Programming Models

- Aurora applications may use:
  - OpenMP 5
  - DPC++/SYCL
  - OpenCL
  - Kokkos
  - Raja

Mapping of Existing Programming Models to Aurora



OpenMP 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
- For Aurora, OpenACC codes could be converted into OpenMP
  - ALCF staff will assist with conversion, training, and best practices
  - Automated translation possible through the clacc conversion tool (for C/C++)

OpenMP®

https://www.openmp.org/

OpenMP 4.5/5: for Aurora

- OpenMP 4.5/5 specification has significant updates to allow for improved support of accelerator devices

Offloading code to run on accelerator	Distributing iterations of the loop to threads	Controlling data transfer between devices

#pragma omp target *[clause[[,] clause],…]*
    structured-block
#pragma omp declare target
    declarations-definition-seq
#pragma omp declare variant*(*variant-func-id) clause new-line*
    function definition or declaration
	#pragma omp teams *[clause[[,] clause],…]*
	    structured-block
	#pragma omp distribute *[clause[[,] clause],…]*
	    for-loops
	#pragma omp loop* *[clause[[,] clause],…]*
	    for-loops
		map *([map-type:] list)*
		    map-type:=alloc | tofrom | from | to | …
		#pragma omp target data *[clause[[,] clause],…]*
		    structured-block
		#pragma omp target update *[clause[[,] clause],…]*

Runtime support routines:

- void **omp_set_default_device**(int dev_num)
- int **omp_get_default_device**(void)
- int **omp_get_num_devices**(void)
- int **omp_get_num_teams**(void)

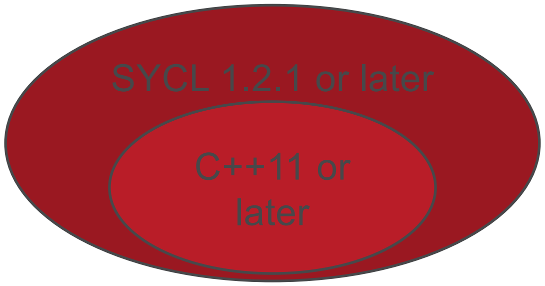
Environment variables

- Control default device through OMP_DEFAULT_DEVICE
- Control offload with OMP_TARGET_OFFLOAD

* denotes OMP 5

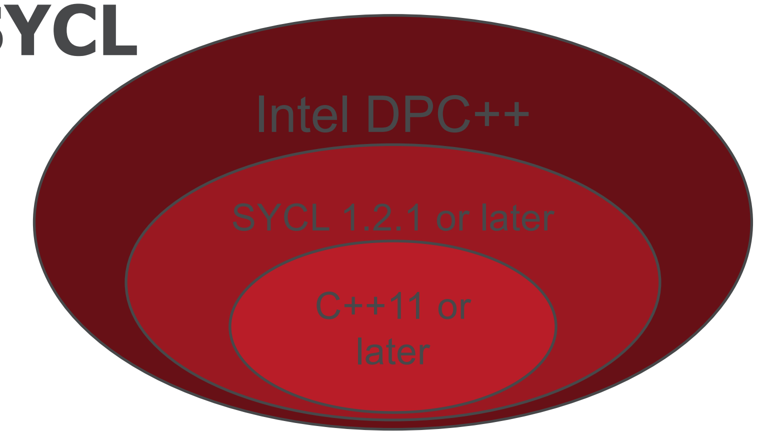
DPC++ (Data Parallel C++) and SYCL

- SYCL
  - Khronos standard specification
  - SYCL is a C++ based abstraction layer (standard C++11)
  - Builds 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**



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

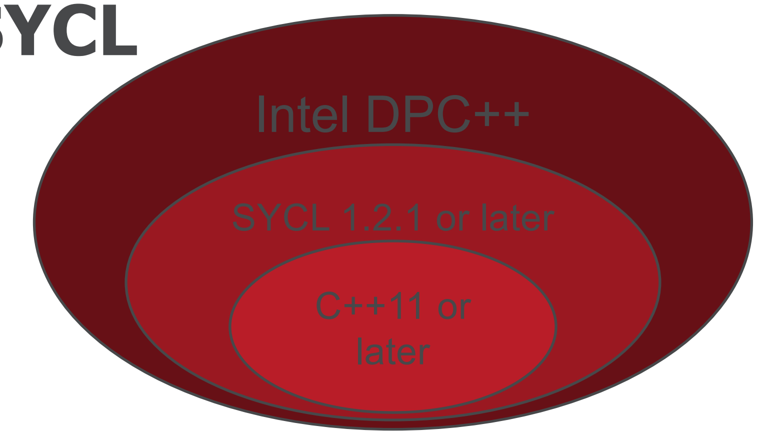
- SYCL
    - Khronos standard specification
    - SYCL is a C++ based abstraction layer (standard C++11)
    - Builds on OpenCL **concepts** (but single-source)
    - *SYCL is designed to be as close to standard C++ as possible*
- Current Implementations of SYCL:
    - ComputeCPP™ (www.codeplay.com)
    - 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**
- DPC++
  - Part of Intel oneAPI specification
  - Intel extension of SYCL to support new innovative features
  - Incorporates SYCL 1.2.1 specification and Unified Shared Memory
  - Add language or runtime extensions as needed to meet user needs



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

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



- SYCL
  - Khronos standard specification
  - SYCL is a C++ based abstraction layer (standard C++11)
  - Builds 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**
- DPC++
  - Part of Intel oneAPI specification
  - Intel ex
  - Incorp
    Memo
  - Add la
    needs

New open-source DPC++ for NVIDIA from CodePlay:
https://github.com/codeplaysoftware/sycl-for-cuda

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.
a	removes requirement to manually name lambdas that define kernels.
	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

OpenCL

- Open standard for heterogeneous device programming (CPU, GPU, FPGA)
  - Utilized for GPU programming
- 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
- Programming aspects:
  - Requires host and device code be in different files
  - Typically uses JIT compilation
- Example: https://github.com/alcf-perfengr/alcl

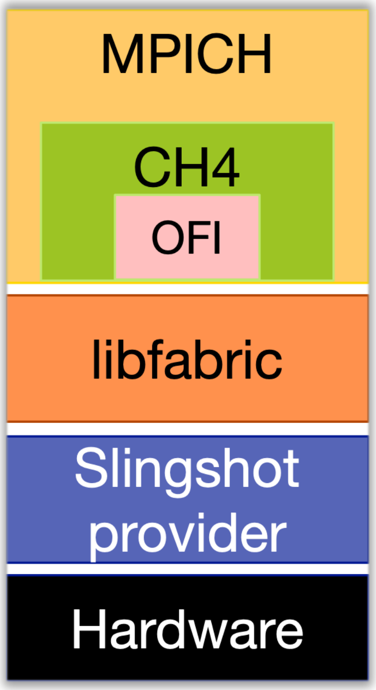
MPI on Aurora

- 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_MUTIPLE (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
  - Control variables



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)
  - Implementations of the oneAPI specification and analysis and debug tools to help programming

Intel Fortran for Aurora

- Fortran 2008
- OpenMP 5
- New compiler—LLVM backend
  - Strong Intel history of optimizing Fortran compilers

Other Aurora Languages

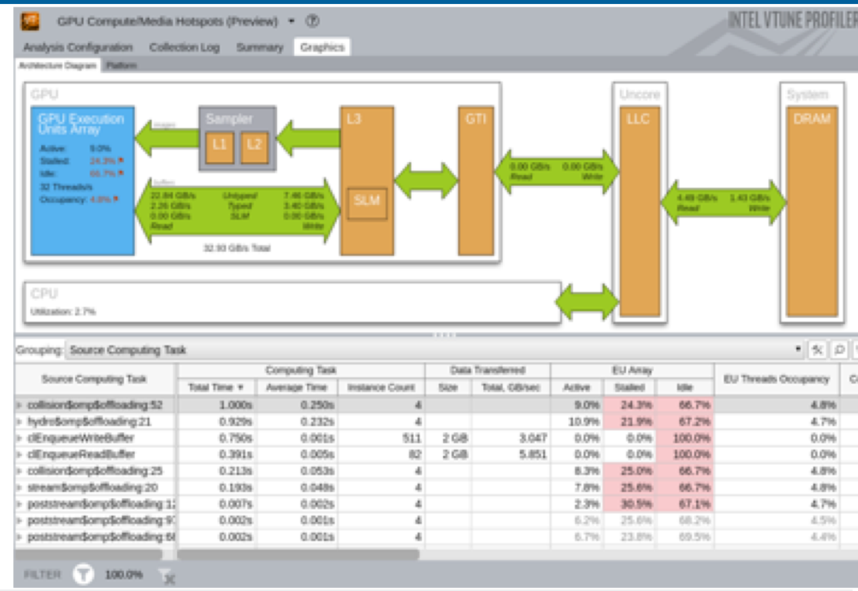
- C++ } OpenMP 5
- C
- DPC++
- Python

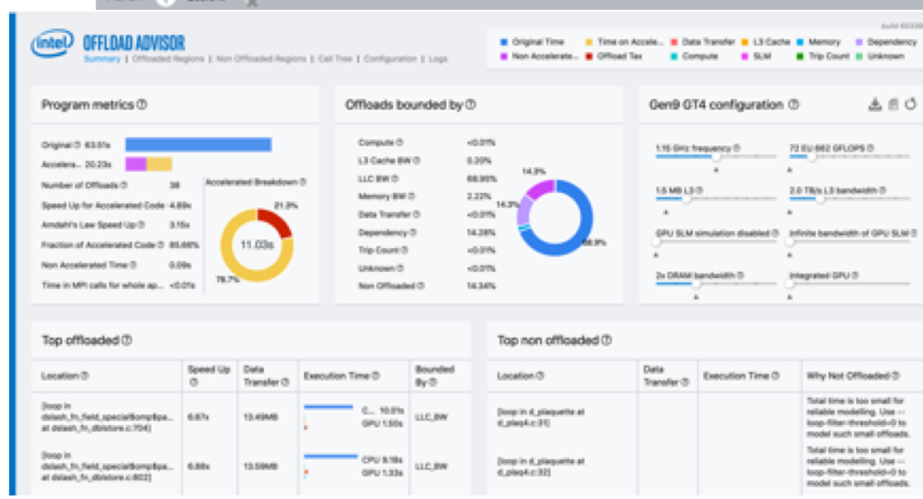
TOOLS AND LIBRARIES

Intel VTune and Advisor

- ❑ 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





Intel MKL – Math Kernel Library

- Highly tuned algorithms
  - FFT
  - Linear algebra (BLAS, LAPACK)
    - Sparse solvers
  - Statistical functions
  - Vector math
  - Random number generators
- Optimized for every Intel platform
- oneAPI MKL (oneMKL)
  - https://software.intel.com/en-us/oneapi/mkl

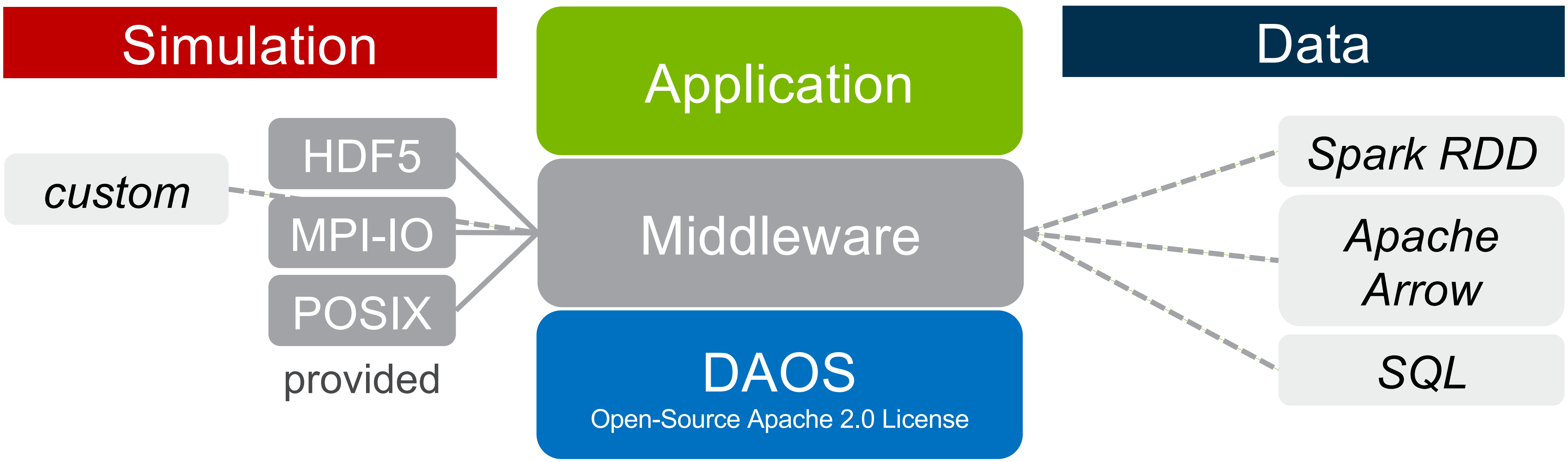
oneAPI beta includes DPC++ support

AI and Analytics

- Libraries to support AI and Analytics
  - OneAPI Deep Neural Network Library (oneDNN)
    - High Performance Primitives to accelerate deep learning frameworks
    - Powers Tensorflow, PyTorch, MXNet, Intel Caffe, and more
    - Running on Gen9 today (via OpenCL)
  - oneAPI Data Analytics Library (oneDAL)
    - Classical Machine Learning Algorithms
    - Easy to use one line daal4py Python interfaces
    - Powers Scikit-Learn
  - Apache Spark MLlib

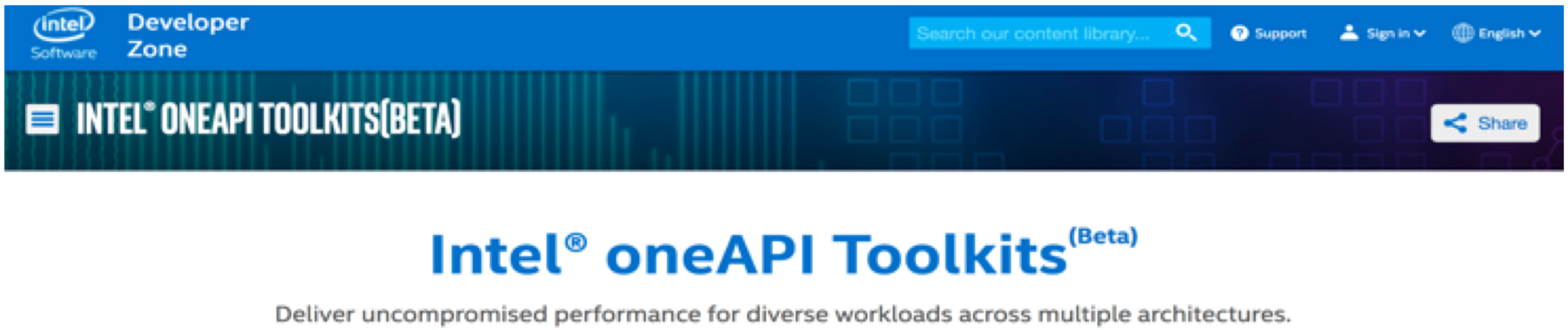
DAOS

- Fast for broad spectrum of read/write patterns
- Wide concurrency
- Redundancy
- Common layer to build I/O middleware on



Try it now
Try it now!
Argonne
NATIONAL LABORATORY

Intel oneAPI Toolkits

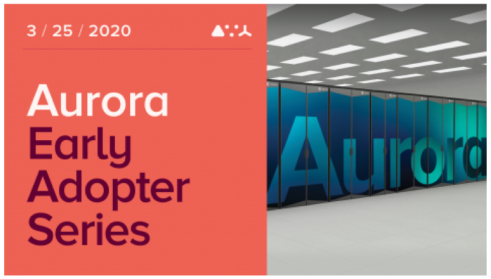


- Use Intel's DevCloud systems
  - oneAPI Toolkits: https://software.intel.com/ONEAPI
- Or, roll your own
  - oneAPI public beta
  - Intel Gen 9 system such as NUC

Learn More

- Aurora Early Adopters webinar series
  - https://www.alcf.anl.gov/aurora-early-adopter-series

- Aurora Documentation
  - https://www.alcf.anl.gov/support-center/aurora



Questions?

Acknowledgements

- Argonne Leadership Computing Facility and Computational Science Division Staff

- This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of two U.S. Department of Energy organizations (Office of Science and the National Nuclear Security Administration) responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering and early testbed platforms, in support of the nation's exascale computing imperative.

- This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.


Thank You
Argonne
NATIONAL LABORATORY

References

- Intel Gen9 GPU Programmer's Reference Manual: https://en.wikichip.org/w/images/3/31/intel-gfx-prm-osrc-skl-vol03-gpu_overview.pdf
- Raja Koduri, Intel Chief Architect, Keynote at Intel HPC Developer Conference, Nov 17, 2019. https://s21.q4cdn.com/600692695/files/doc_presentations/2019/11/DEVCON-2019_16x9_v13_FINAL.pdf
- DAOS:
  - DAOS Community Home: https://wiki.hpdd.intel.com/display/DC/DAOS+Community+Home
  - DAOS User Group meeting: https://wiki.hpdd.intel.com/display/DC/DUG19
- Slingshot
  - Hot Interconnects Keynote by Steve Scott, CTO, Cray: http://www.hoti.org/hoti26/slides/sscott.pdf
  - HPC User Forum talk by Steve Scott, Sep 2019
  - https://www.nextplatform.com/2019/08/16/how-cray-makes-ethernet-suited-for-hpc-and-ai-with-slingshot/