Hewlett Packard
Enterprise

BEYOND SUPER

MAKING APPLICATIONS READY FOR THE EXASCALE ERA

Gabriele Paciucci Sr. Solution Architect EMEA

June, 2020

BEYOND SUPER

POWERING THE EXASCALE ERA

MODELING & SIMULATION + ARTIFICIAL INTELLIGENCE + BIG DATA ANALYTICS

RUNNING ON ONE MACHINE IN MISSION-CRITICAL WORKFLOWS

EXASCALE ERA

PERFORMS LIKE A SUPERCOMPUTER

RUNS LIKE A CLOUD

ADAPTIVE SUPERCOMPUTING

HPE's Cray Shasta supercomputer is focused on delivering innovative next-generation systems that integrate diverse processing technologies at the node level into a unified architecture, allowing customers to meet their users' continued demand for higher sustained performance.

- Flexibility in node design.
- Full software and user programming environment.
- Scalable HPC and storage network.
- Predictable HPC performance at scale.
- Cloud service delivery models.
- Support for Multi-Tenancy.

2023

AMD + AMD

Genoa Next-Gen

2022

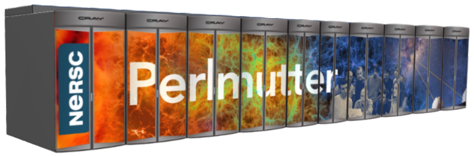
INTEL + INTEL

SPR PVC

2021

AMD + AMD

Trento MI200



2020

AMD + NVIDIA

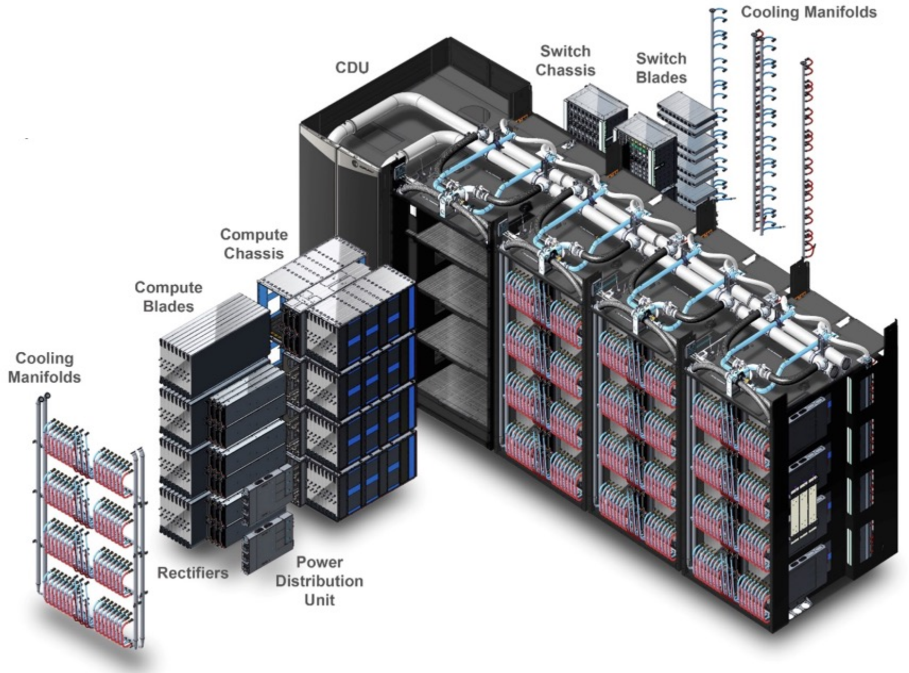
Milan Volta Next

WHAT MAKES CRAY SHASTA UNIQUE

SHASTA OLYMPUS INFRASTRUCTURE

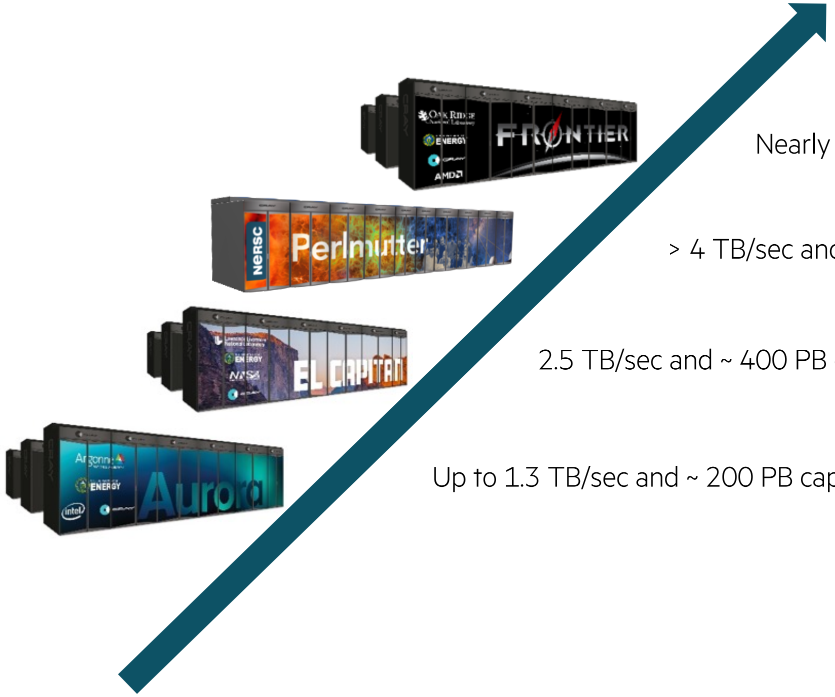
Architected for maximum performance, density, efficiency, and scale

- Up to 64 compute blades, and **512 processors per rack**
- Flexible bladed architecture supports **multiple generations** of CPUs, GPUs, and interconnect
- **Cableless interconnect** between switches and nodes inside chassis
- **100% direct liquid cooling** enables 300KW capability per rack
- Scales to 100’s of cabinets



CRAY CLUSTERSTOR E1000 : VERY POWERFUL, HYBRID AND SCALABLE

- Powerful design
- Intelligent tiering software
- Hybrid, HDD and SSD
- Scalable
- Directly connected to Slingshot fabric



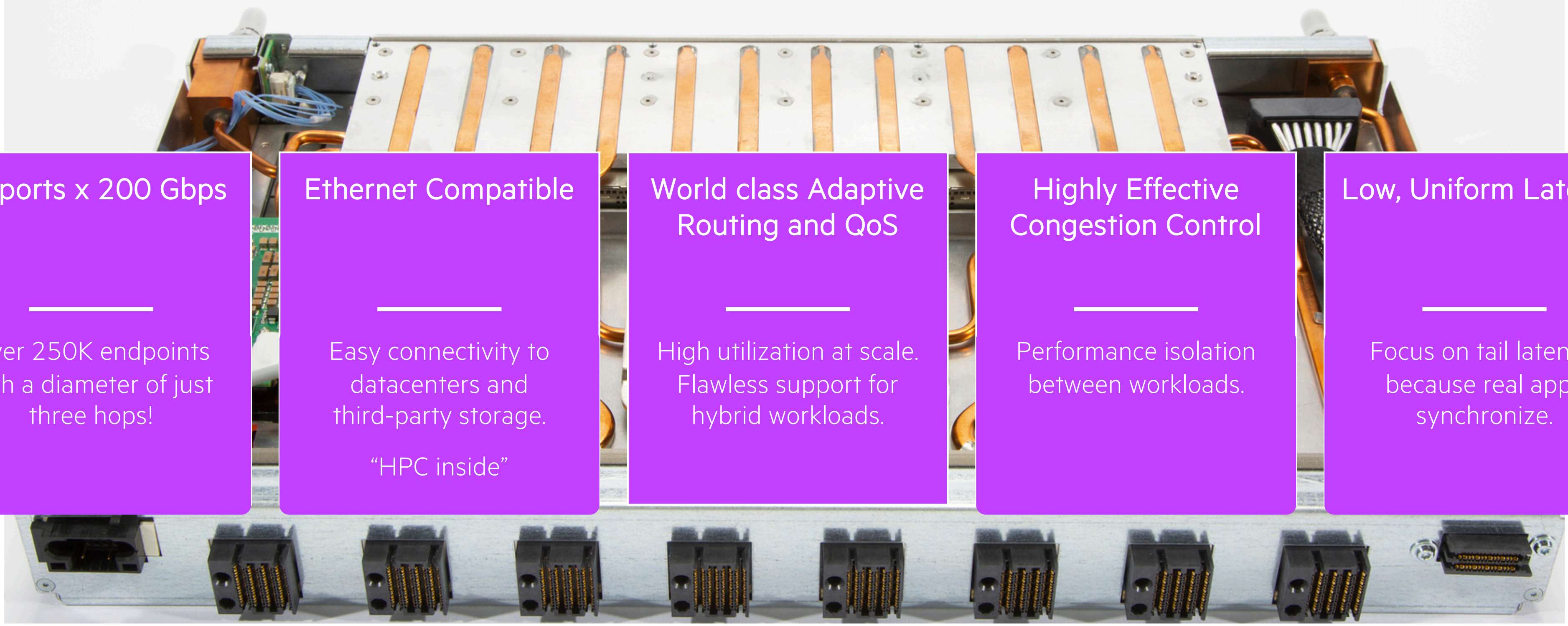
Nearly 10 TB/sec and > 700 PB capacity

> 4 TB/sec and 30PB (All Flash) capacity

2.5 TB/sec and ~ 400 PB capacity

Up to 1.3 TB/sec and ~ 200 PB capacity

SLINGSHOT: INTERCONNECT FOR A DATA-CENTRIC WORLD



64 ports x 200 Gbps

Over 250K endpoints with a diameter of just three hops!

Ethernet Compatible

Easy connectivity to datacenters and third-party storage.

“HPC inside”

World class Adaptive Routing and QoS

High utilization at scale. Flawless support for hybrid workloads.

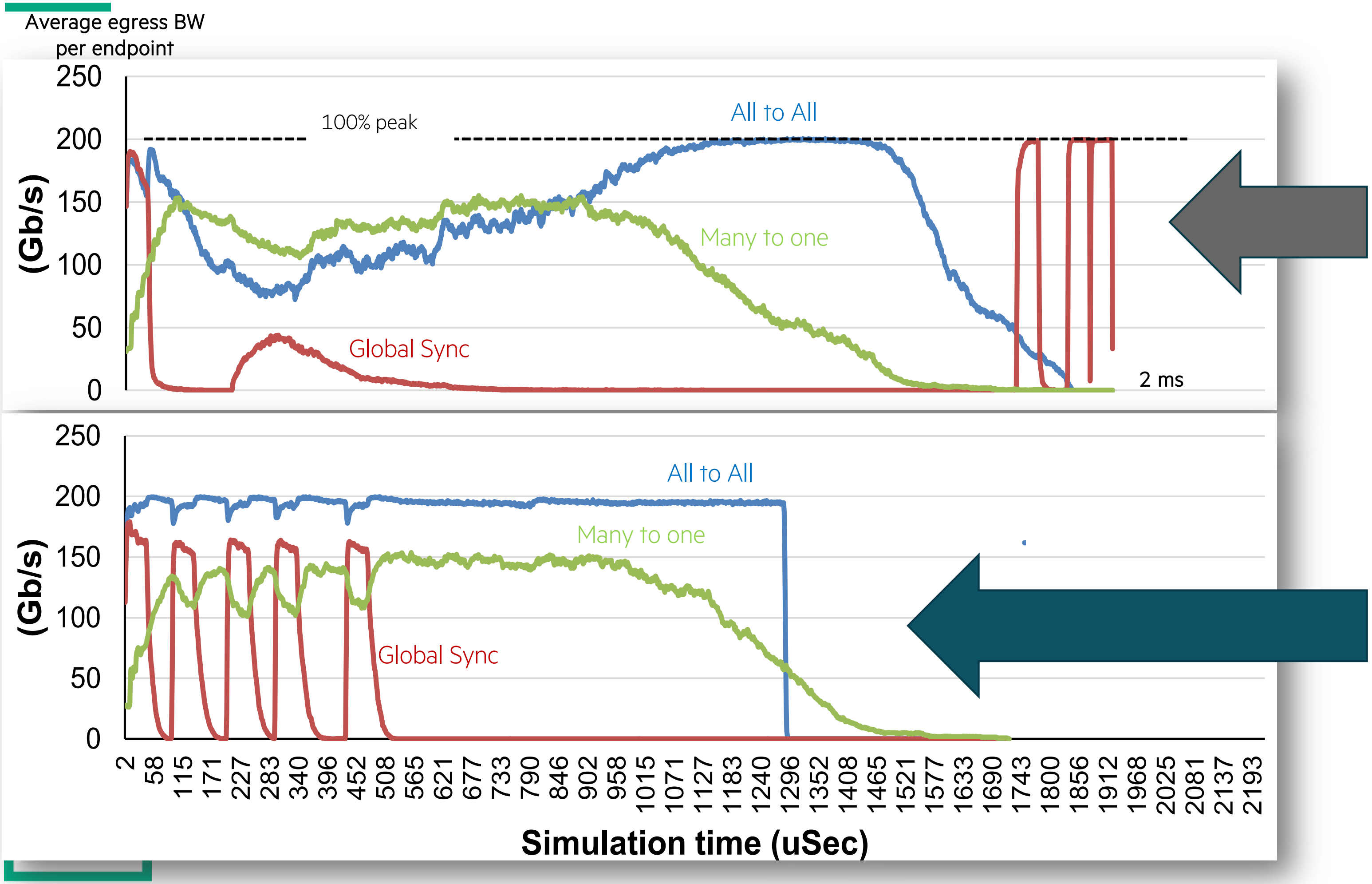
Highly Effective Congestion Control

Performance isolation between workloads.

Low, Uniform Latency

Focus on tail latency, because real apps synchronize.

CONGESTION MANAGEMENT PROVIDES PERFORMANCE ISOLATION

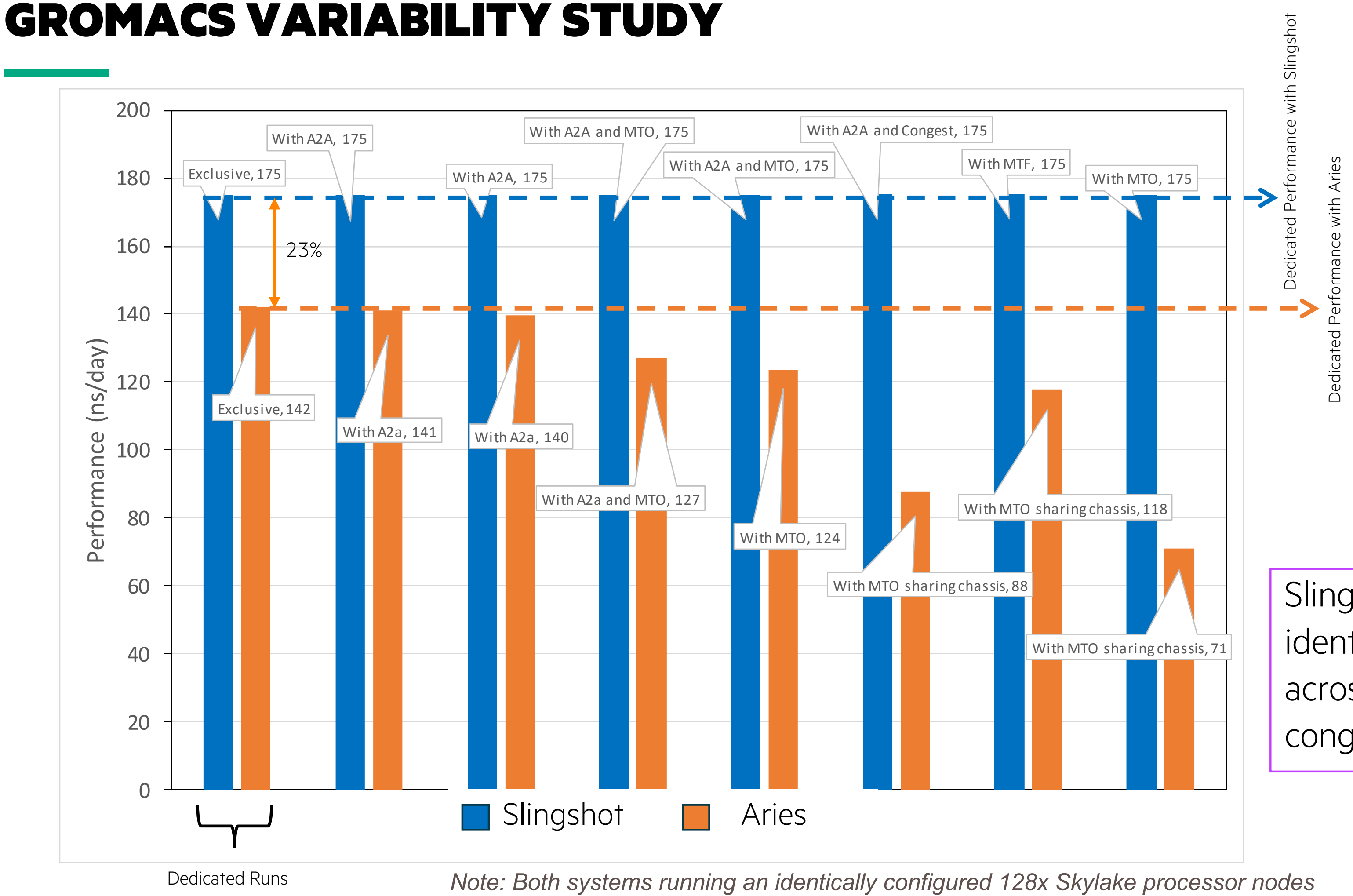


Job Interference in today's networks

Congesting (green) traffic hurts well-behaved (blue) traffic, and *really* hurts latency-sensitive, synchronized (red) traffic.

With Slingshot Advanced Congestion Management

GROMACS VARIABILITY STUDY



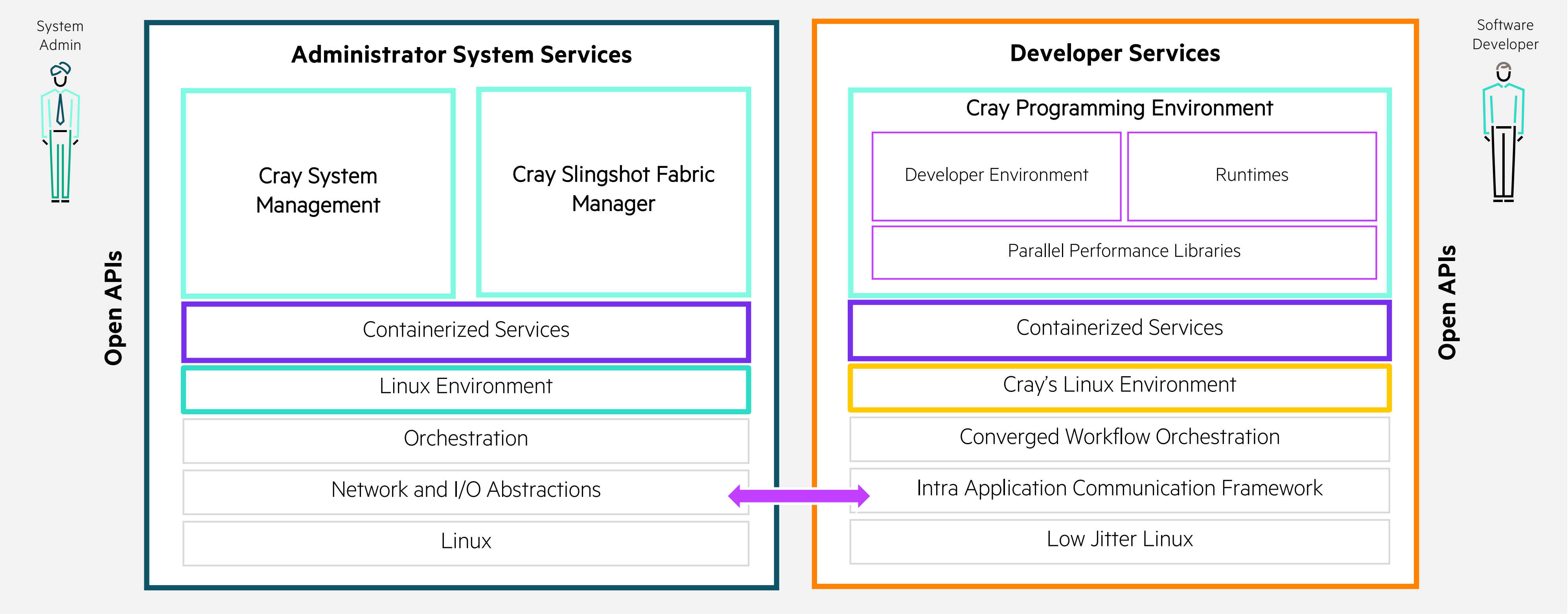
Note: Both systems running an identically configured 128x Skylake processor nodes

Highly Effective Congestion Control

Performance isolation between workloads.

Slingshot in Shasta delivers identical performance results across trials with every congestion type

CRAY SOFTWARE PLATFORM ARCHITECTURE

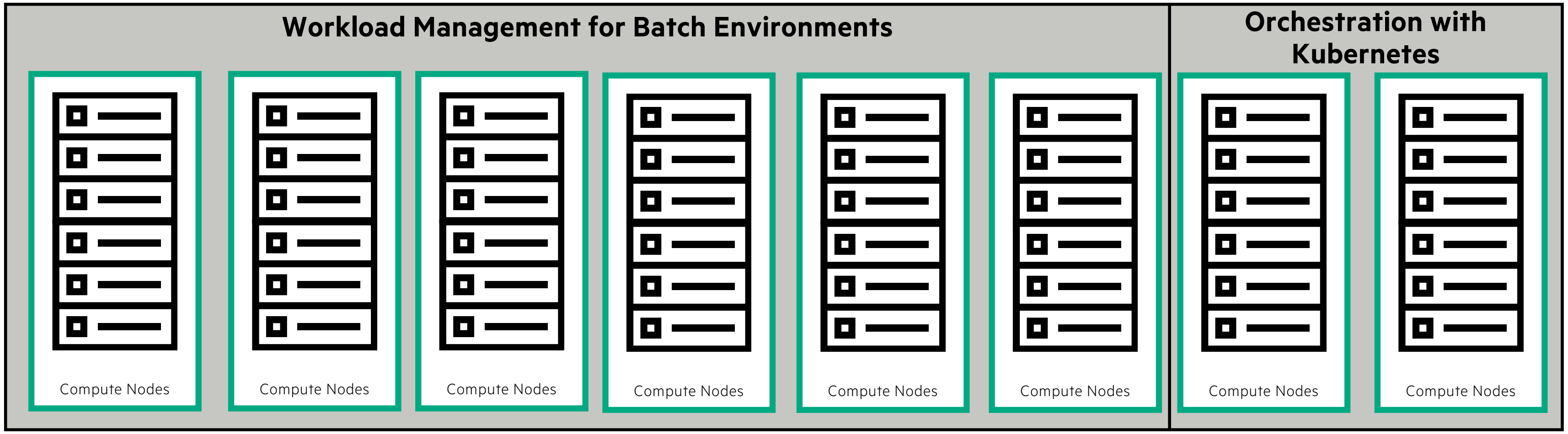


Expanding the power of supercomputing with the flexibility of cloud and full datacenter interoperability

PERFORMS LIKE A SUPERCOMPUTER RUNS LIKE A CLOUD

Flexible Workload Management

- **Batch environment**
  - Support for Slurm
- **Orchestrated workloads**
  - Leverage Kubernetes on compute nodes for orchestrated application (e.g. Urika for ML/DL)
  - Enables additional cloud-like provisioning of new workflows and enables cog-sim integration
- **Leverages built-in partitioning feature for Cray System Management**



ONE EXPERIENCE ACROSS ARCHITECTURES

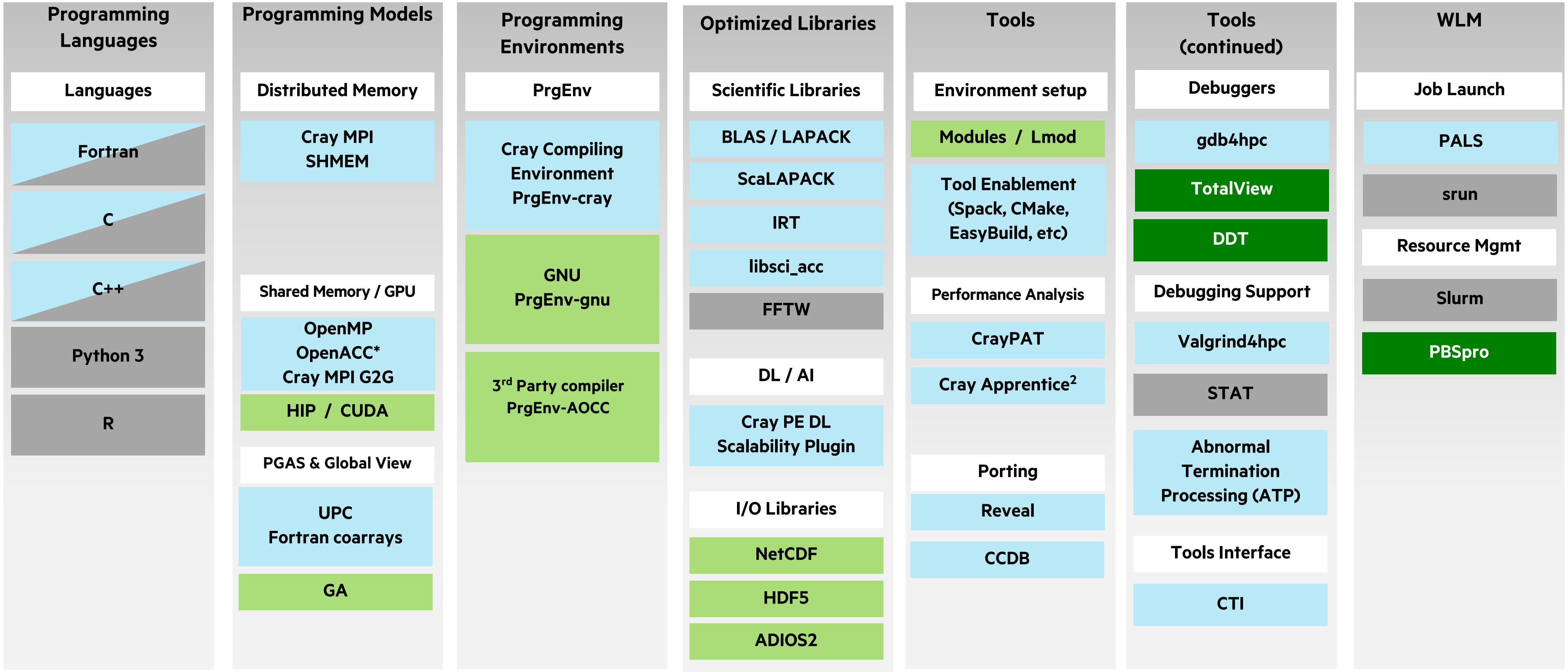
Capability	AMD Epyc (2019)	Intel Xeon (2020)	ARM CPU (2021)	NVIDIA GPU (2020)	AMD GPU (2021)	Intel GPU (2022)
Single management interface	●	●	●	●	●	●
Single monitoring framework and datastore	●	●	●	●	●	●
Cray OS	●	●	●	N/A	N/A	N/A
Portable development environment (containerized tools)	●	●	●	●	●	●
Slingshot Optimized Middleware: MPI, OpenMP, PGAS, SHMEM	●	●	●	●	●	●
Optimized Math Libraries	●	● ○	●	○	● ○	○
Code Optimization, Porting, and Debug	● ○	● ○	● ○	○	● ○	○
Fully supported compilation and execution environment (Fortran, C, C++, Python, and R)	● ○	● ○	● ○	○	● ○	○

Key

● Optimized or enhanced by Cray

○ Industry or 3rd party – integrated by Cray

CRAY DEVELOPMENT ENVIRONMENT



Programming Languages

- Languages
- Fortran
- C
- C++
- Python 3
- R

Programming Models

- Distributed Memory
- Cray MPI
SHMEM
- Shared Memory / GPU
- OpenMP
OpenACC*
Cray MPI G2G
- HIP / CUDA
- PGAS & Global View
- UPC
Fortran coarrays
- GA

Programming Environments

- PrgEnv
- Cray Compiling Environment
PrgEnv-cray
- GNU
PrgEnv-gnu
- 3rd Party compiler
PrgEnv-AOCC

Optimized Libraries

- Scientific Libraries
- BLAS / LAPACK
- ScaLAPACK
- IRT
- libsci_acc
- FFTW
- DL / AI
- Cray PE DL Scalability Plugin
- I/O Libraries
- NetCDF
- HDF5
- ADIOS2

Tools

- Environment setup
- Modules / Lmod
- Tool Enablement (Spack, CMake, EasyBuild, etc)
- Performance Analysis
- CrayPAT
- Cray Apprentice2
- Porting
- Reveal
- CCDB

Tools (continued)

- Debuggers
- gdb4hpc
- TotalView
- DDT
- Debugging Support
- Valgrind4hpc
- STAT
- Abnormal Termination Processing (ATP)
- Tools Interface
- CTI

WLM

- Job Launch
- PALS
- srun
- Resource Mgmt
- Slurm
- PBSpro

* Fortran only

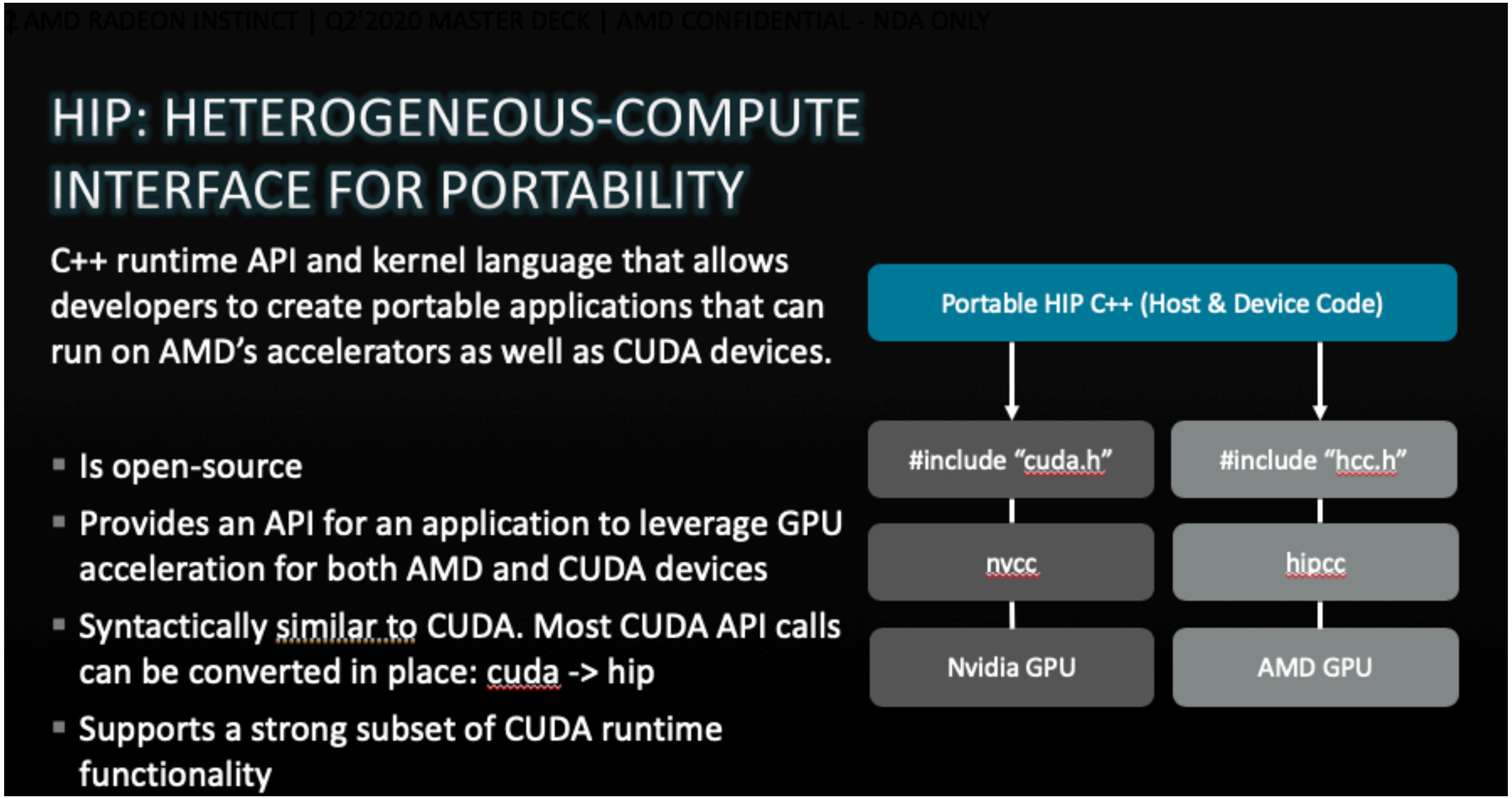
🟦 Cray/HPE Developed

▨ Cray/HPE added value to 3rd party

🟩 3rd party package

🟩 Licensed ISV SW

HIP: AMD OPEN PORTABLE GPU PROGRAMMING MODEL



2 AMD RADEON INSTINCT | Q2'2020 MASTER DECK | AMD CONFIDENTIAL - NDA ONLY

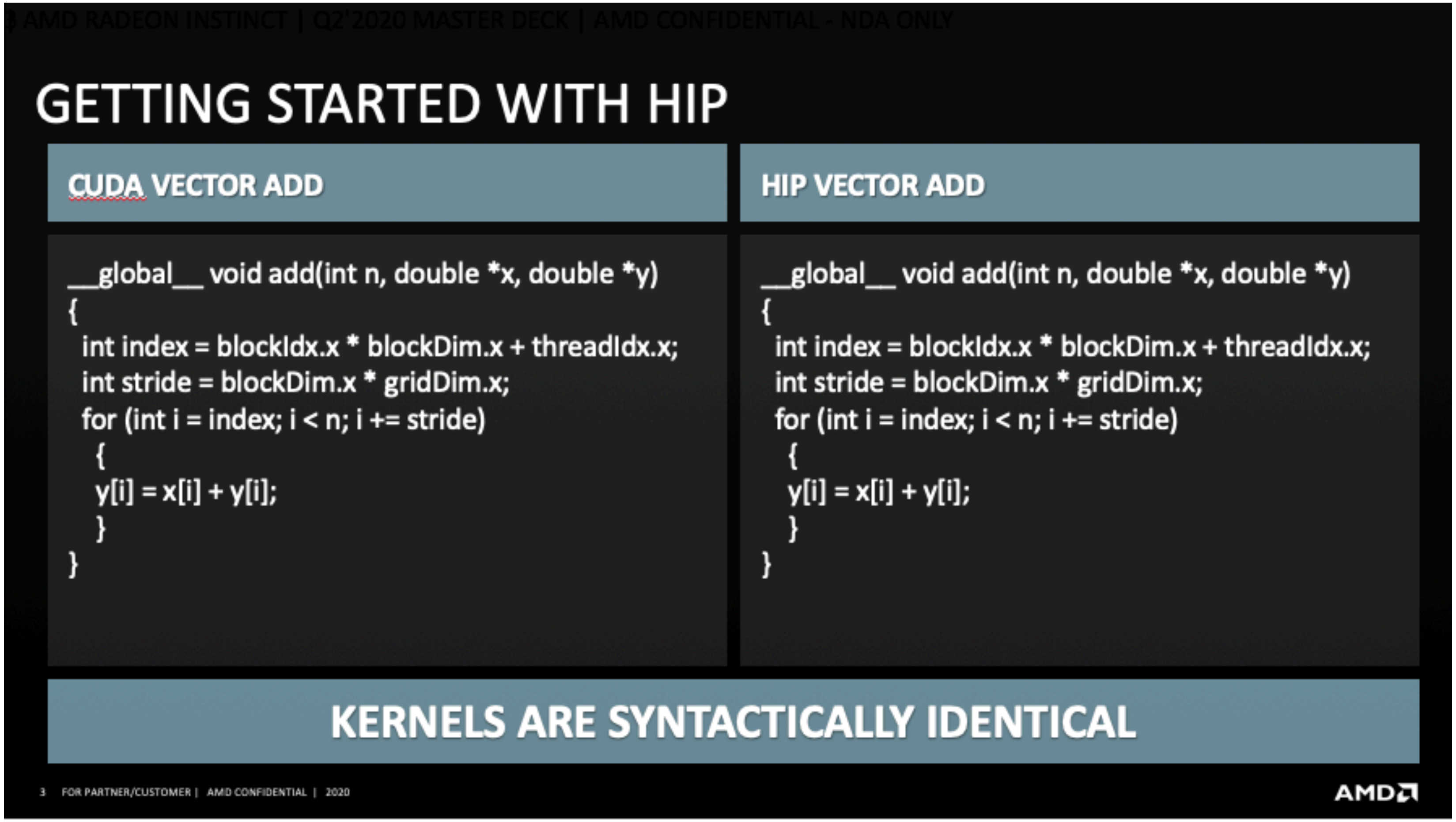
HIP: HETEROGENEOUS-COMPUTE INTERFACE FOR PORTABILITY

C++ runtime API and kernel language that allows developers to create portable applications that can run on AMD's accelerators as well as CUDA devices.

- Is open-source
- Provides an API for an application to leverage GPU acceleration for both AMD and CUDA devices
- Syntactically similar to CUDA. Most CUDA API calls can be converted in place: cuda -> hip
- Supports a strong subset of CUDA runtime functionality

https://www.olcf.ornl.gov/wp-content/uploads/2019/09/AMD_GPU_HIP_training_20190906.pdf

KERNEL CODE IS IDENTICAL BETWEEN CUDA AND HIP



3 AMD RADEON INSTINCT | Q2'2020 MASTER DECK | AMD CONFIDENTIAL - NDA ONLY

GETTING STARTED WITH HIP

CUDA VECTOR ADD

```
__global__ void add(int n, double *x, double *y)
{
  int index = blockIdx.x * blockDim.x + threadIdx.x;
  int stride = blockDim.x * gridDim.x;
  for (int i = index; i < n; i += stride)
    {
    y[i] = x[i] + y[i];
    }
}
```

HIP VECTOR ADD

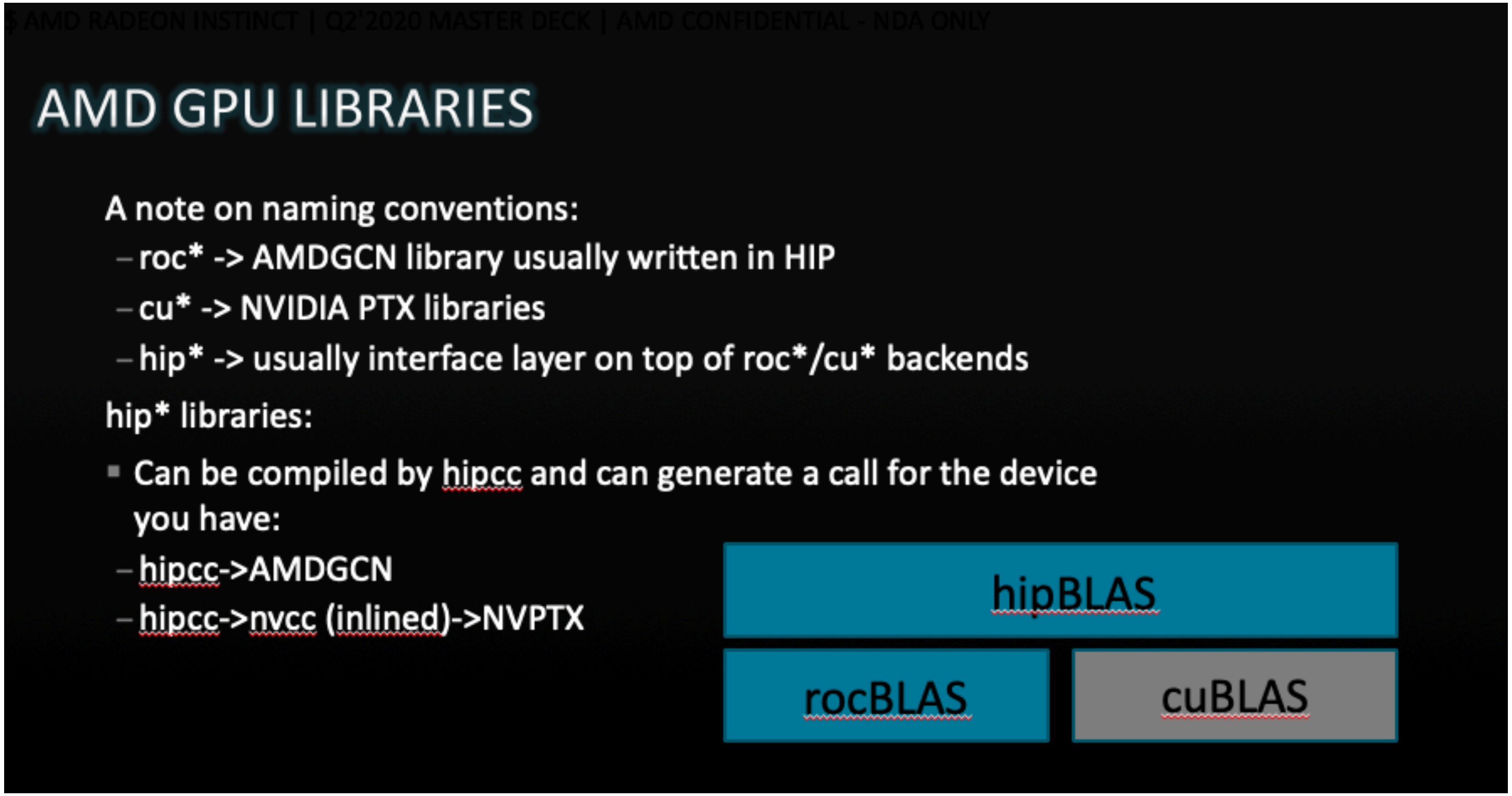
```
__global__ void add(int n, double *x, double *y)
{
  int index = blockIdx.x * blockDim.x + threadIdx.x;
  int stride = blockDim.x * gridDim.x;
  for (int i = index; i < n; i += stride)
    {
    y[i] = x[i] + y[i];
    }
}
```

KERNELS ARE SYNTACTICALLY IDENTICAL

3 FOR PARTNER/CUSTOMER | AMD CONFIDENTIAL | 2020

https://www.olcf.ornl.gov/wp-content/uploads/2019/09/AMD_GPU_HIP_training_20190906.pdf

HIP PORTABLE SCIENTIFIC LIBRARIES



AMD RADEON INSTINCT | Q2'2020 MASTER DECK | AMD CONFIDENTIAL - NDA ONLY

AMD GPU LIBRARIES

A note on naming conventions:

- roc* -> AMDGCN library usually written in HIP
- cu* -> NVIDIA PTX libraries
- hip* -> usually interface layer on top of roc*/cu* backends

hip* libraries:

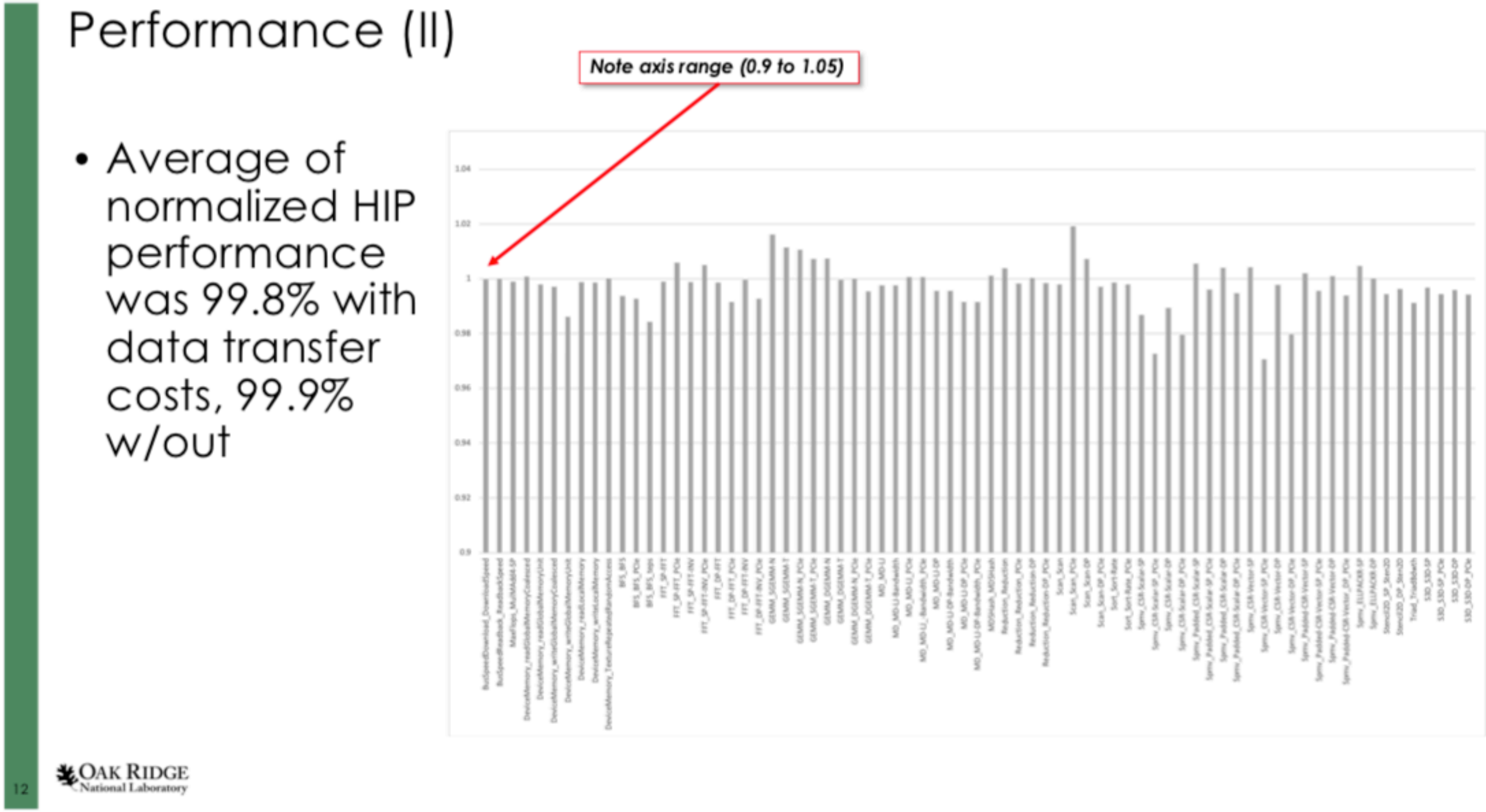
- Can be compiled by hipcc and can generate a call for the device you have:
  - hipcc->AMDGCN
  - hipcc->nvcc (inlined)->NVPTX

https://www.olcf.ornl.gov/wp-content/uploads/2019/09/AMD_GPU_HIP_training_20190906.pdf

PORTING THE QUDA LIBRARY TO HIP

- MILC collaboration code for lattice QCD calculations
- The MILC Code is a body of high performance research software written in C for doing SU(3) lattice gauge theory on high performance computers as well as single-processor workstations. A wide variety of applications are included.
- MILC implementation for GPU is largely based on the QUDA library
- QUDA depends on many additional CUDA libraries: Eigen, CuFFT, CuBLAS, CuRAND, Thrust, CUB
- AMD asked a local consulting company to port QUDA to HIP:
  - 10K lines of hand-tuned CUDA kernels -> hipify converted without problems
  - 35K lines of header code -> hipify mostly converted but needed manual switch to new library dependencies
  - 74K lines of library code -> mostly successful hipify conversion, reuired some manual changes
  - 34K lines of test suite code -> hipify converted without problems
  - Template use for tests -> no solution, manual porting
- 15 developer days
- Much much bigger effort without HIP tools

HIP MEASURED PERFORMANCE ON NVIDIA GPU



Performance (II)

- Average of normalized HIP performance was 99.8% with data transfer costs, 99.9% w/out

Note axis range (0.9 to 1.05)

12

Oak Ridge National Laboratory

https://www.olcf.ornl.gov/wp-content/uploads/2019/10/Roth-HIP-on-Summit-20191009.pdf

THANK YOU

gabriele.paciucci@hpe.com