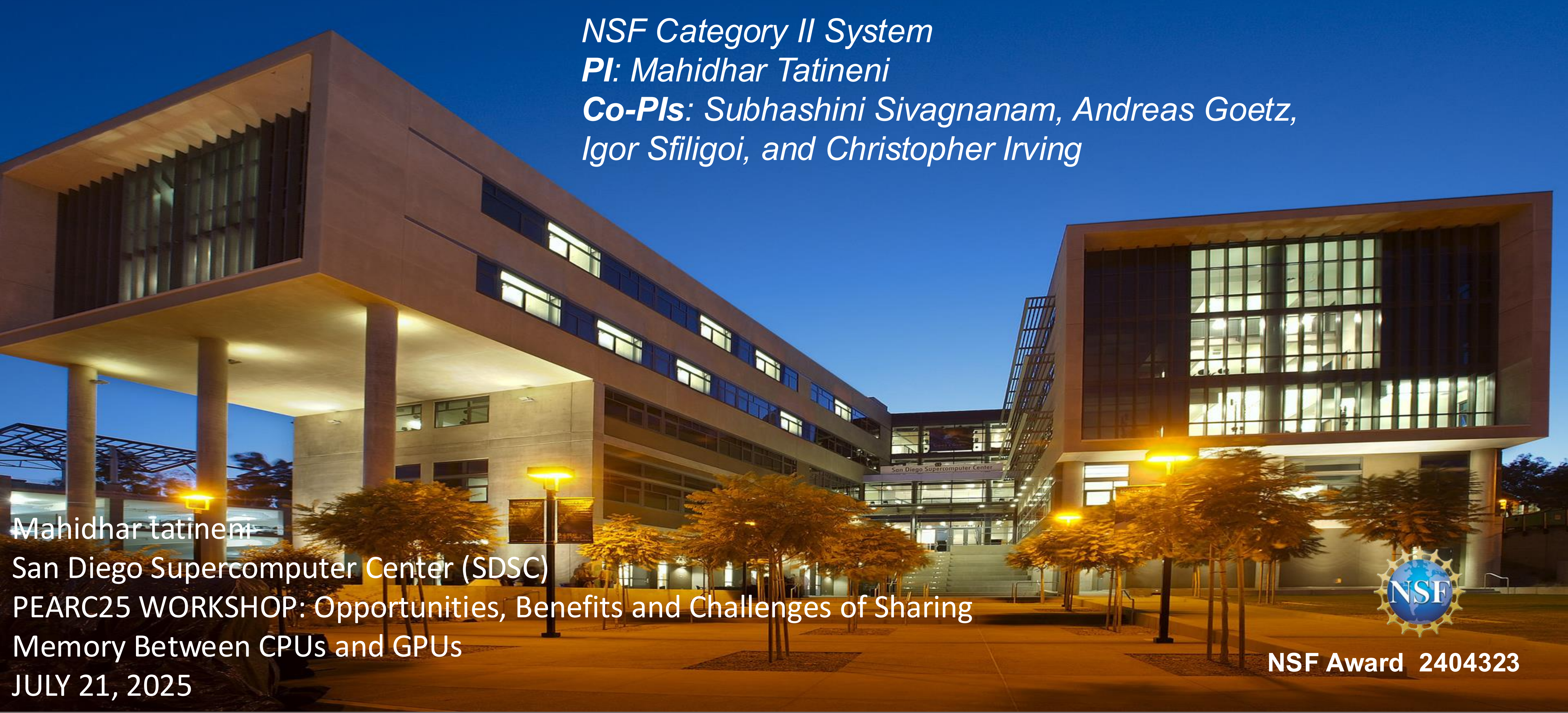
COSMOS: INNOVATIVE SYSTEM FEATURING AMD MI300A APUs

NSF Category II System
***PI**: Mahidhar Tatineni*
***Co-PIs**: Subhashini Sivagnanam, Andreas Goetz, Igor Sfiligoi, and Christopher Irving*



Mahidhar tatineni
San Diego Supercomputer Center (SDSC)
PEARC25 WORKSHOP: Opportunities, Benefits and Challenges of Sharing Memory Between CPUs and GPUs
JULY 21, 2025

NSF Award 2404323

A category II testbed system supported by NSF Award # 2404323, Office of Advanced Cyberinfrastructure (OAC).

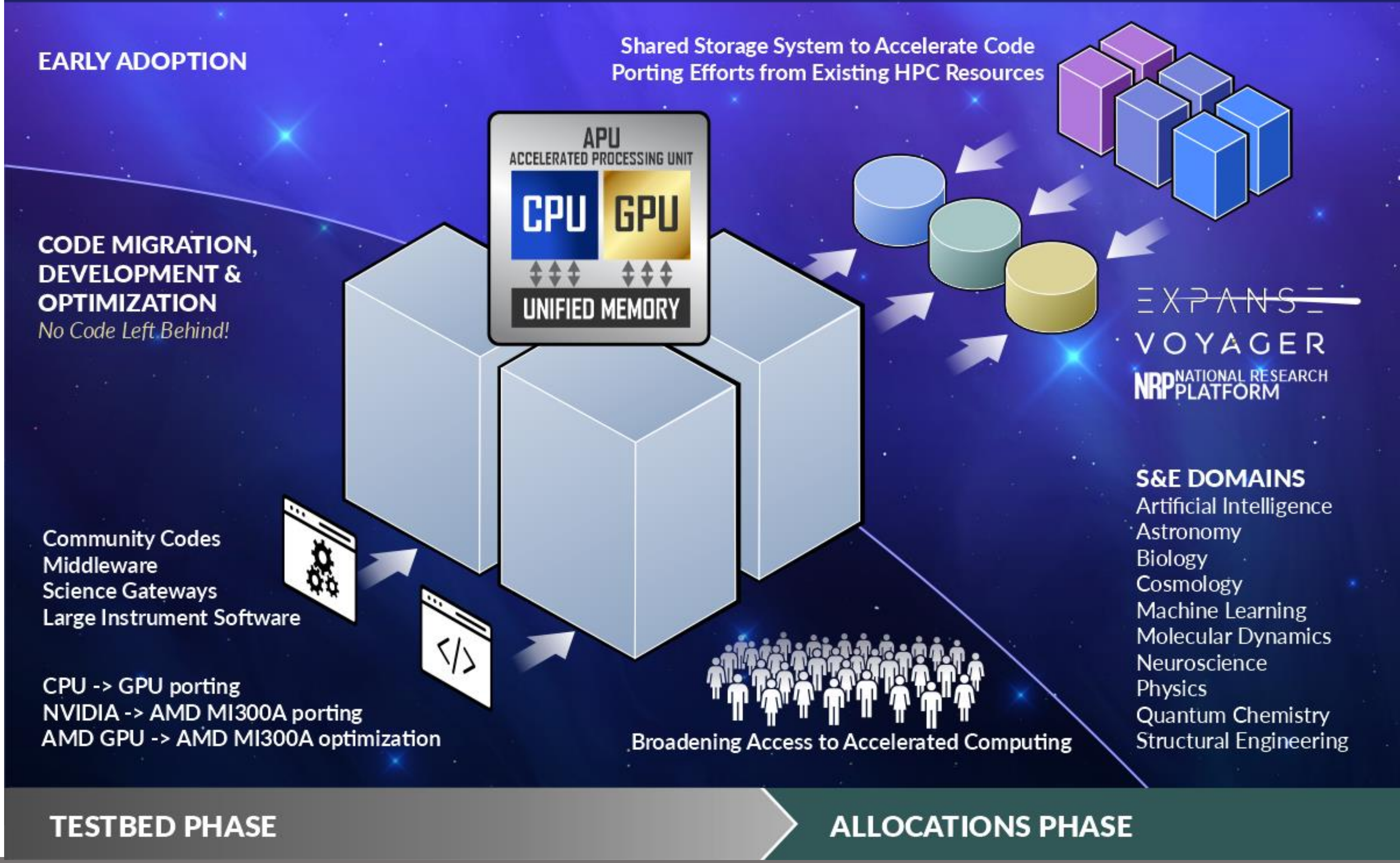
***PI**: Mahidhar Tatineni*

***Co-PIs**:*
Subhashini Sivagnanam
Andreas Goetz
Igor Sfiligoi
Christopher Irving

Senior Personnel:
Robert Sinkovits
Mary Thomas

COSMOS

DEMOCRATIZING the ACCELERATOR ECOSYSTEM for SCIENCE and DISCOVERY



EARLY ADOPTION

CODE MIGRATION, DEVELOPMENT & OPTIMIZATION
No Code Left Behind!

Shared Storage System to Accelerate Code Porting Efforts from Existing HPC Resources

Community Codes
Middleware
Science Gateways
Large Instrument Software

CPU -> GPU porting
NVIDIA -> AMD MI300A porting
AMD GPU -> AMD MI300A optimization

Broadening Access to Accelerated Computing

S&E DOMAINS
Artificial Intelligence
Astronomy
Biology
Cosmology
Machine Learning
Molecular Dynamics
Neuroscience
Physics
Quantum Chemistry
Structural Engineering

TESTBED PHASE

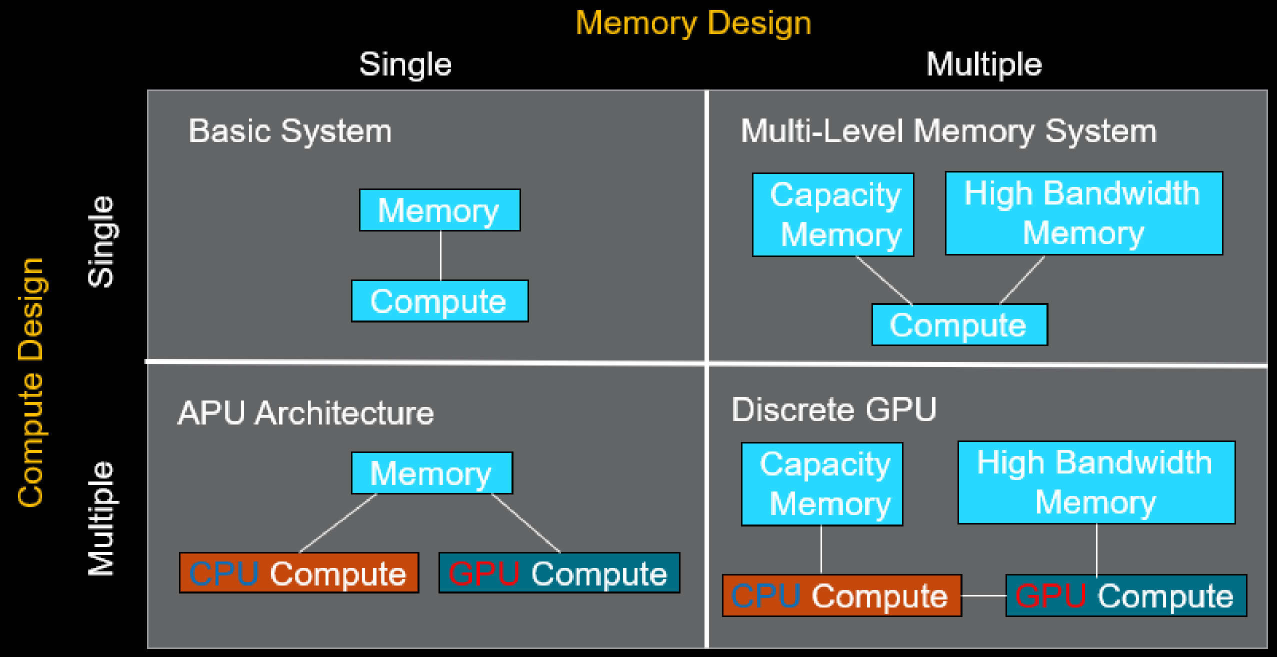
ALLOCATIONS PHASE

Cosmos: Democratizing the Accelerator Ecosystem for Science and Discovery

- Ease the pathway for accelerating a wide range of AI, science and engineering applications, including many that have not yet been ported to GPUs.
- Enabling technology is the AMD Instinct MI300A Accelerated Processing Unit (APU) that integrates CPU and GPU capabilities with unified memory. Total of 42 nodes (4APUs/node), with a total of 168 APUs.
- APU memory architecture facilitates an incremental programming approach, easing porting of applications, and enabling many communities to adopt GPU acceleration on this first NSF datacenter APU-based HPC system.
- HPE Cray Supercomputing EX2500 with HPE Slingshot interconnect, 4-socket nodes with all-to-all connectivity using AMD high-speed interconnect. Fully liquid cooled.
- High-performance storage from VAST (600TB) that provides the high IOPS, and bandwidth needed for the anticipated mixed-application workload
- Research collaborations with applications from artificial intelligence, astronomy, neuroscience, molecular biology, structural engineering and more. Target community codes, science gateways and enabling middleware.

The APU architecture of MI300A

Foundation of MI300A Design



Memory Design

Single

Multiple

Compute Design

Single

Multiple

Basic System

Memory

Compute

Multi-Level Memory System

Capacity Memory

High Bandwidth Memory

Compute

APU Architecture

Memory

CPU Compute

GPU Compute

Discrete GPU

Capacity Memory

High Bandwidth Memory

CPU Compute

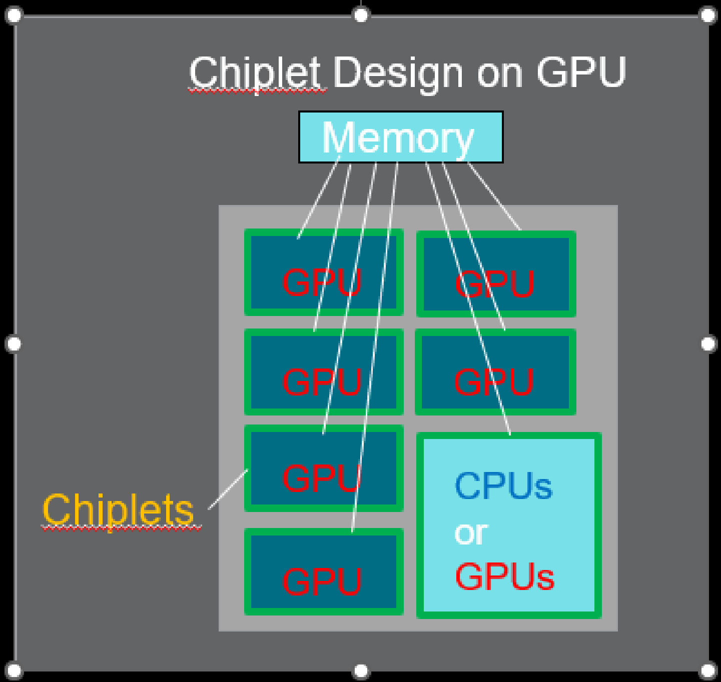
GPU Compute

Taxonomy categorizes architecture by dominance of hardware components

- ❖Memory Dominant – architecture revolves around a single memory space
- ❖Compute Dominant – architecture centered around a single compute resource
- ❖APU is primarily characterized by compute units being able to address all memory

1. Single Memory Space for both CPU and GPU

2. Chiplet Design on a Larger Silicon footprint



Chiplet Design on GPU

Memory

GPU

GPU

GPU

GPU

GPU

GPU

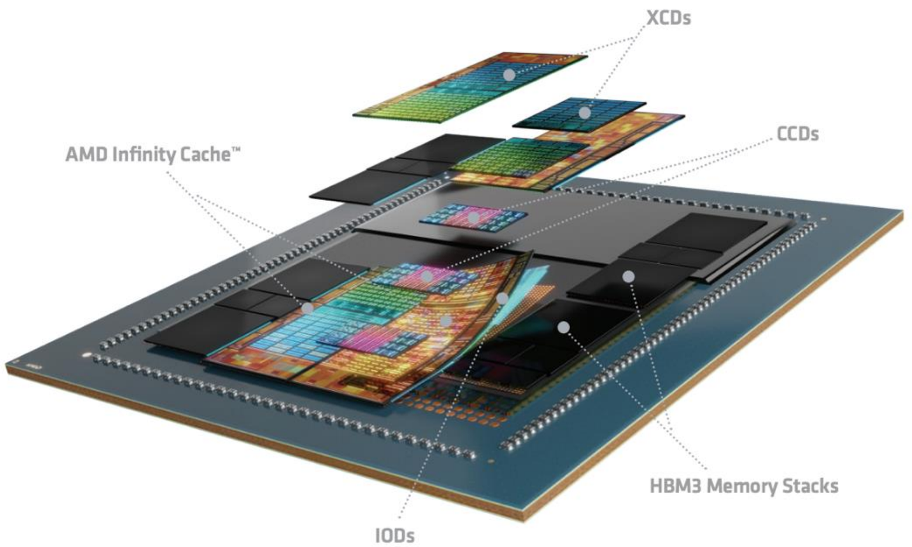
CPUs or GPUs

Chiplets

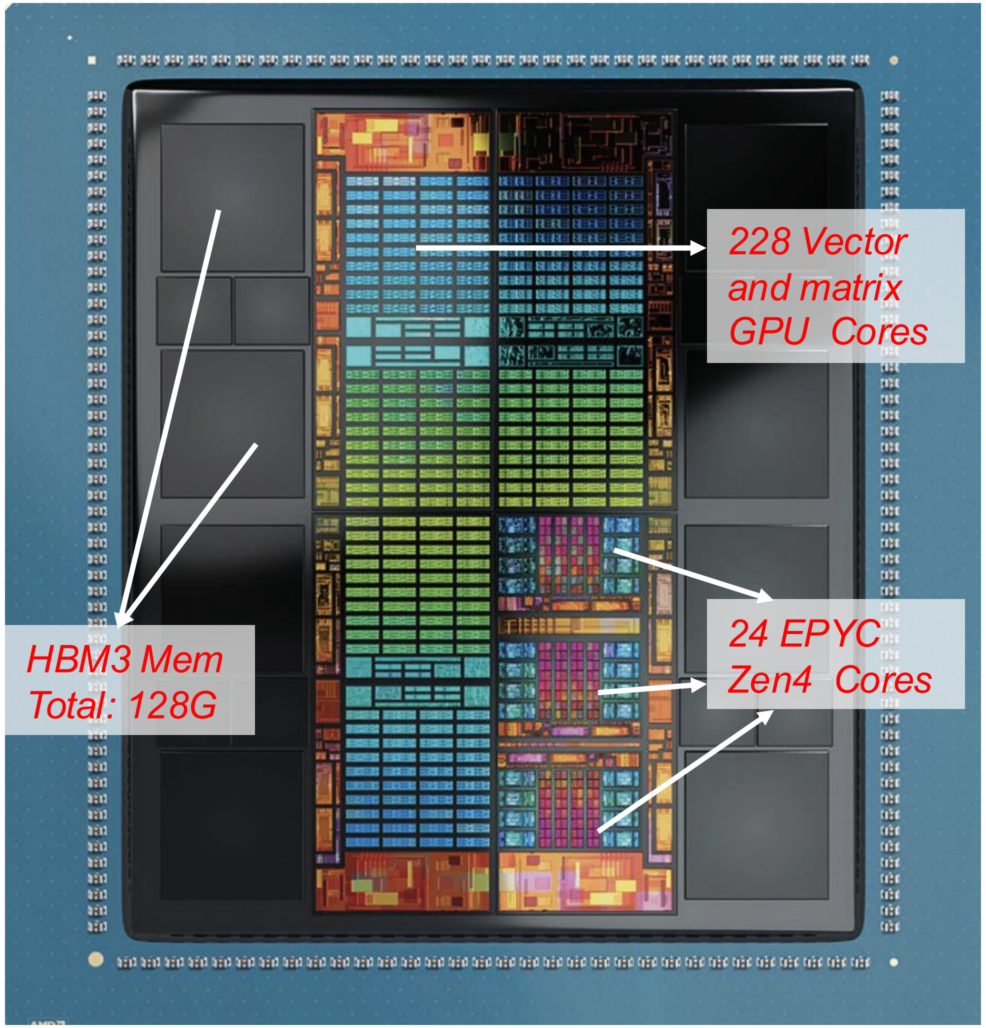
Lots more silicon space and chiplets gives more compute capability

Slide Credit: Bob Robey, AMD

AMD MI300A Architecture



- *MI300A APU combines CPU, GPU, and memory on one package. Six accelerated compute dies (XCDs) combined with 3 chiplets with 8 zen4 cores each.*
- *128 GB of HBM3 memory shared coherently between CPUs and GPUs. 5.3 TB/s peak throughput.*
- *256 MB AMD Infinity Cache (last level) shared between XCDs and CPUs*
- *Design helps eliminate need to do host/device data copies and eases code development (going towards our goal of democratizing access to accelerators!)*



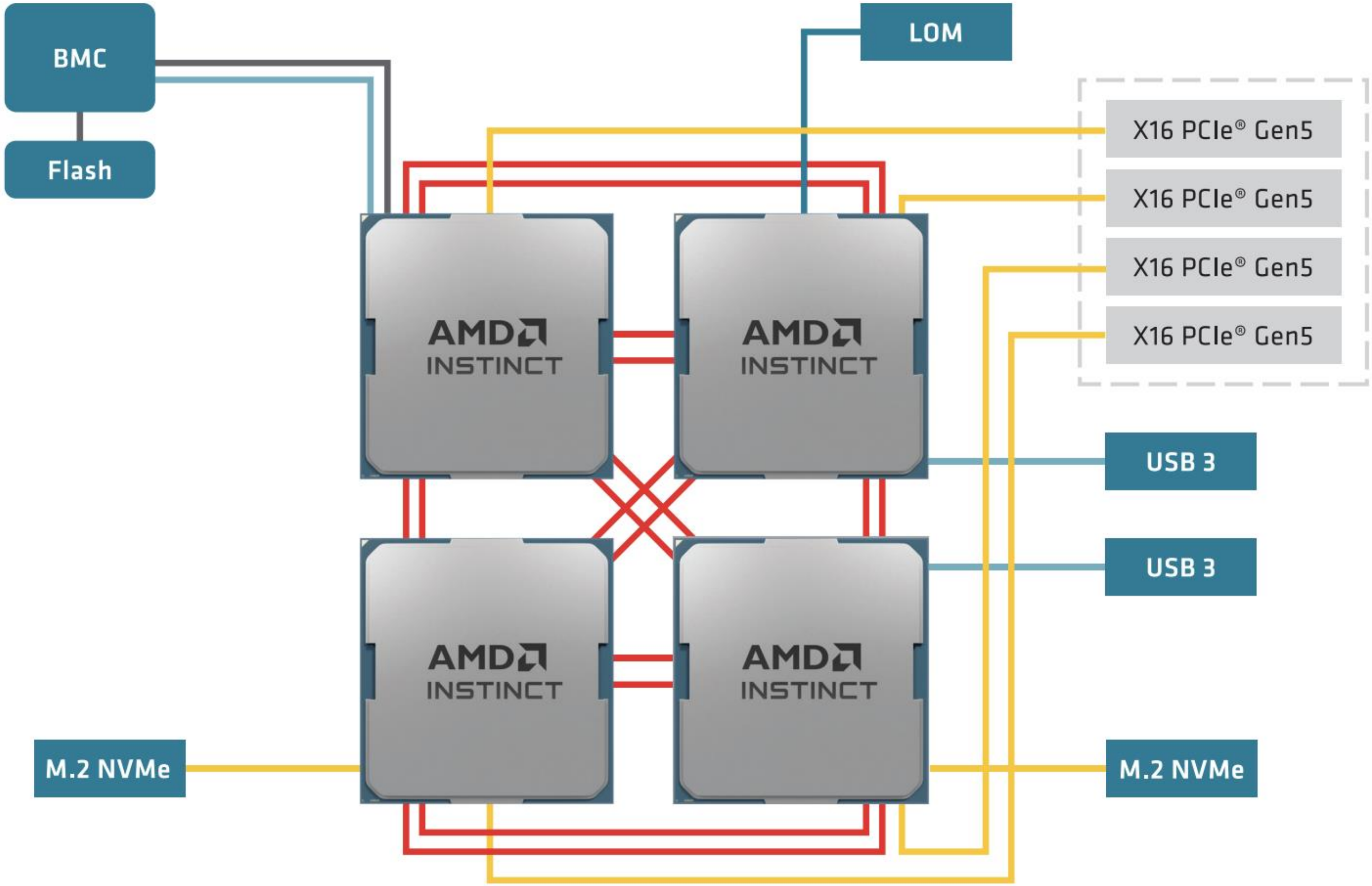
228 Vector and matrix GPU Cores

HBM3 Mem Total: 128G

24 EPYC Zen4 Cores

Reference: https://www.amd.com/content/dam/amd/en/documents/instinct-tech-docs/white-papers/amd-cdna-3-white-paper.pdf

Typical 4-APU Node Architecture



- 4 MI300A APUs fully connected via AMD's inter-chip global memory interface (xGMI)
- xGMI provides 768 GB/s aggregate bi-directional bandwidth among all the APUs, 256 GB/s bi-directional peer-to-peer bandwidth between each pair of APUs.
- Each APU is served by a 200-Gbps Slingshot NIC, which provides connectivity to the Slingshot interconnect

Reference: https://www.amd.com/content/dam/amd/en/documents/instinct-tech-docs/white-papers/amd-cdna-3-white-paper.pdf

HPE CRAY EX2500 Supercomputer

Single rack of EX2500

Compute Chassis: Provides power, cooling, system control, and network fabric for up to 8 compute blade slots

3 Chassis per rack, 8 blades per chassis, and each EX255a blade can support:

•Two 4-socket AMD MI300a Accelerator APU nodes

•128GB HBM3 per APU

•Up to 8 HPE Slingshot 200Gbit/sec ports per blade

•1 local NVMe - M.2 SSD per node(upto 2 per blade)

•2 Board Management Controllers (BMC) per blade

•Cooled with cold plate

Reference: https://www.hpe.com/psnow/doc/a00094635enw

Cosmos: Full System Details

- 42 nodes, each with 4 APUs (168 total) integrated into full system using HPE's Slingshot technology
- Slingshot technology is an extension of Ethernet with HPC features that include low latency, high packet rates, and load-based adaptive routing to minimize network congestion
- High-performance storage from VAST (~300TB) that provides the high IOPS, and bandwidth needed for the anticipated mixed-application workload
- 5 PB of Ceph capacity storage
- Qumulo storage for home filesystem
- HPE EX2500 rack with liquid cooling
- HPE Cray System Management tools
- Full suite of HPE/Cray software development tools – programming environment; debugging, performance measurement and analysis tools (CrayPat).

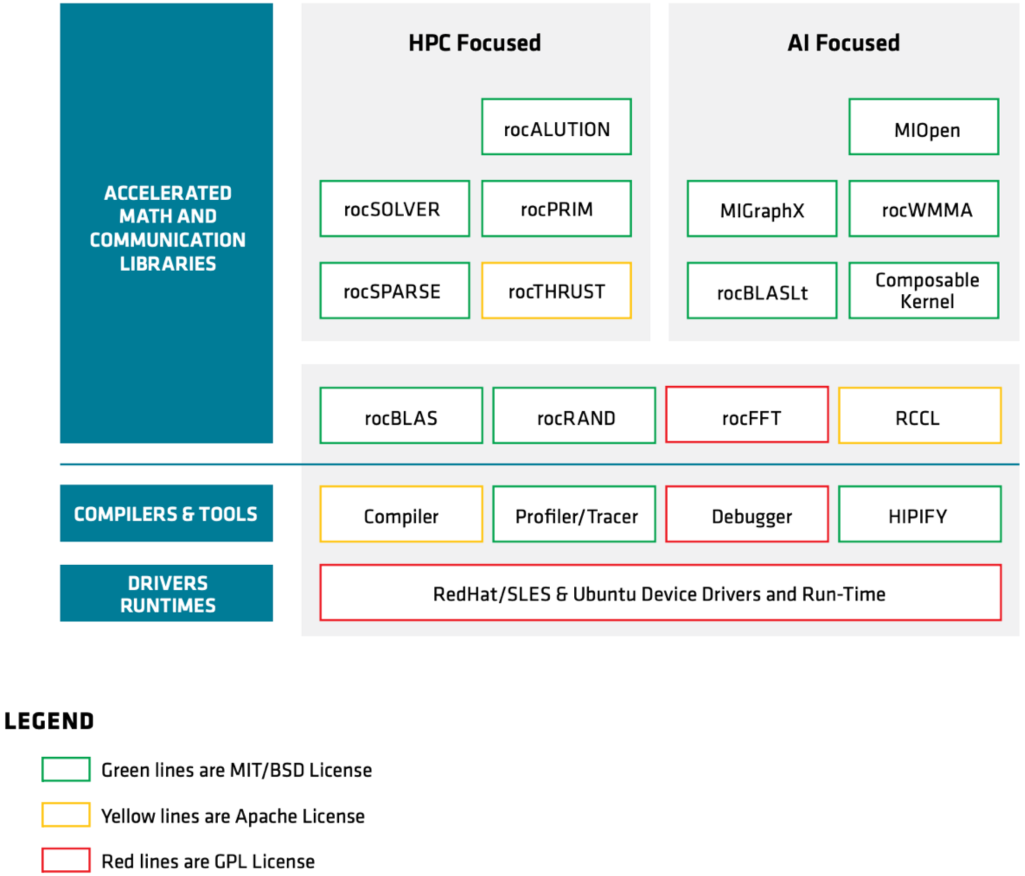
Programming the MI300A

- APU Programming Model
    - Memory movement calls or directives not needed
        - Saves most of the work in porting from CPU
    - Can port one loop at a time and interleave CPU and GPU work
    - Lots of programming languages
        - Native – HIP/ROCm
        - Pragma-based – OpenMP® ; Example of application ported with this approach: OpenFOAM
        - Portability Frameworks such as Kokkos/Raja
- Standard compilers – AMD clang/flang
    - Additional: GNU, Cray compilers + ROCm/HIP
- Can run applications written for previous AMD Instinct™ GPUs
    - Any applications that are already ported using HIP/ROCm should work. Some API calls may not be supported and need workaround

Programming the MI300A: ROCm Platform

- AMD ROCm™ open software platform for accelerated computing is a comprehensive set of open-source APIs, compilers, libraries, and development tools

- Targeted at both HPC and AI focused workloads.



ACCELERATED MATH AND COMMUNICATION LIBRARIES

HPC Focused

rocALUTION

rocSOLVER

rocPRIM

rocSPARSE

rocTHRUST

AI Focused

MIOpen

MIGraphX

rocWMMA

rocBLASLt

Composable Kernel

rocBLAS

rocRAND

rocFFT

RCCL

COMPILERS & TOOLS

Compiler

Profiler/Tracer

Debugger

HIPIFY

DRIVERS RUNTIMES

RedHat/SLES & Ubuntu Device Drivers and Run-Time

LEGEND

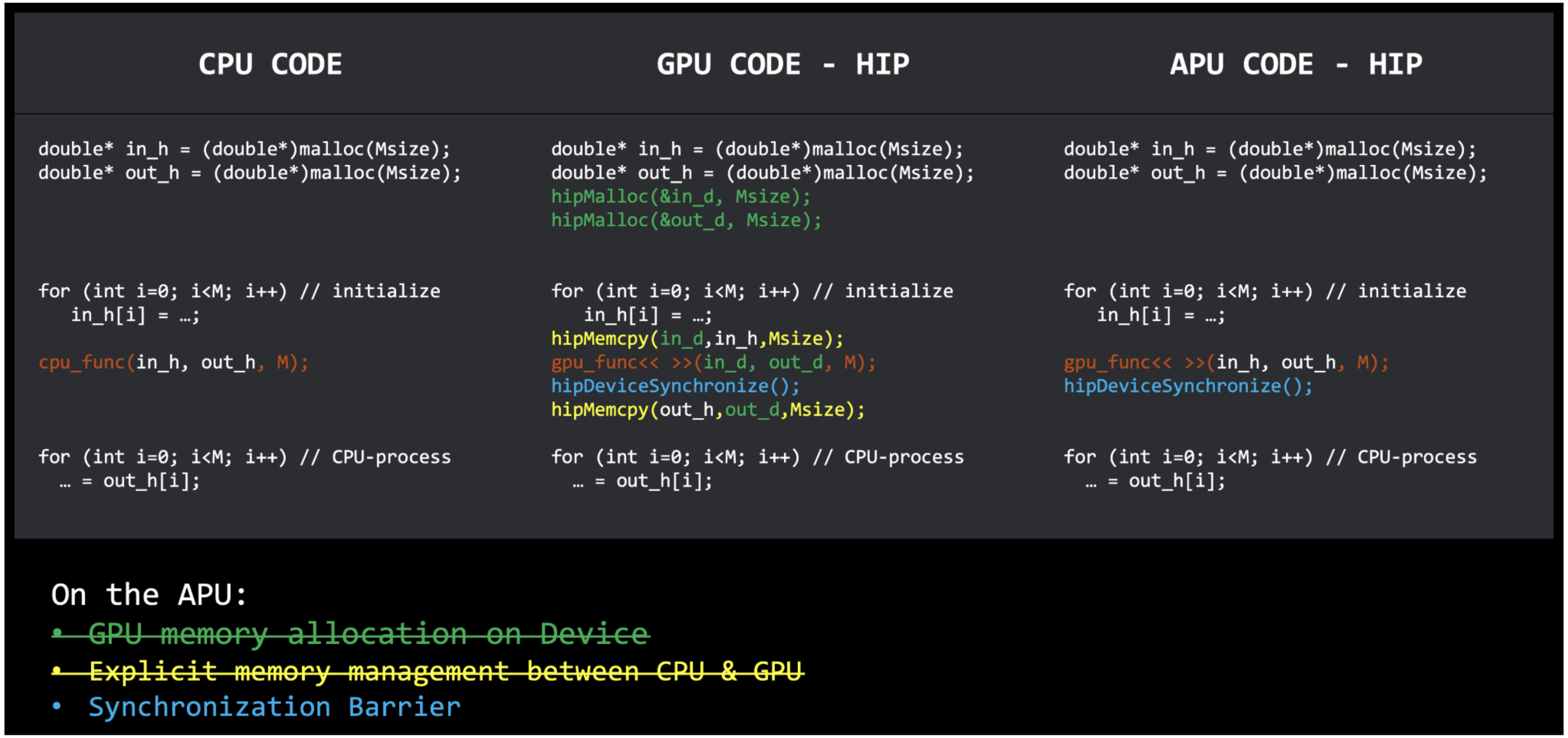
🟩 Green lines are MIT/BSD License

🟨 Yellow lines are Apache License

🟥 Red lines are GPL License

Reference: https://www.amd.com/content/dam/amd/en/documents/instinct-tech-docs/white-papers/amd-cdna-3-white-paper.pdf

MI300A: APU Programming Model



CPU CODE

```
double* in_h = (double*)malloc(Msize);
double* out_h = (double*)malloc(Msize);


for (int i=0; i<M; i++) // initialize
   in_h[i] = …;

cpu_func(in_h, out_h, M);


for (int i=0; i<M; i++) // CPU-process
  … = out_h[i];
```

GPU CODE - HIP

```
double* in_h = (double*)malloc(Msize);
double* out_h = (double*)malloc(Msize);
hipMalloc(&in_d, Msize);
hipMalloc(&out_d, Msize);

for (int i=0; i<M; i++) // initialize
   in_h[i] = …;
hipMemcpy(in_d,in_h,Msize);
gpu_func<< >>(in_d, out_d, M);
hipDeviceSynchronize();
hipMemcpy(out_h,out_d,Msize);

for (int i=0; i<M; i++) // CPU-process
  … = out_h[i];
```

APU CODE - HIP

```
double* in_h = (double*)malloc(Msize);
double* out_h = (double*)malloc(Msize);


for (int i=0; i<M; i++) // initialize
   in_h[i] = …;

gpu_func<< >>(in_h, out_h, M);
hipDeviceSynchronize();


for (int i=0; i<M; i++) // CPU-process
  … = out_h[i];
```

On the APU:

- ~~GPU memory allocation on Device~~
- ~~Explicit memory management between CPU & GPU~~
- Synchronization Barrier

Shared Memory => Porting of codes from CPU to APU is easier

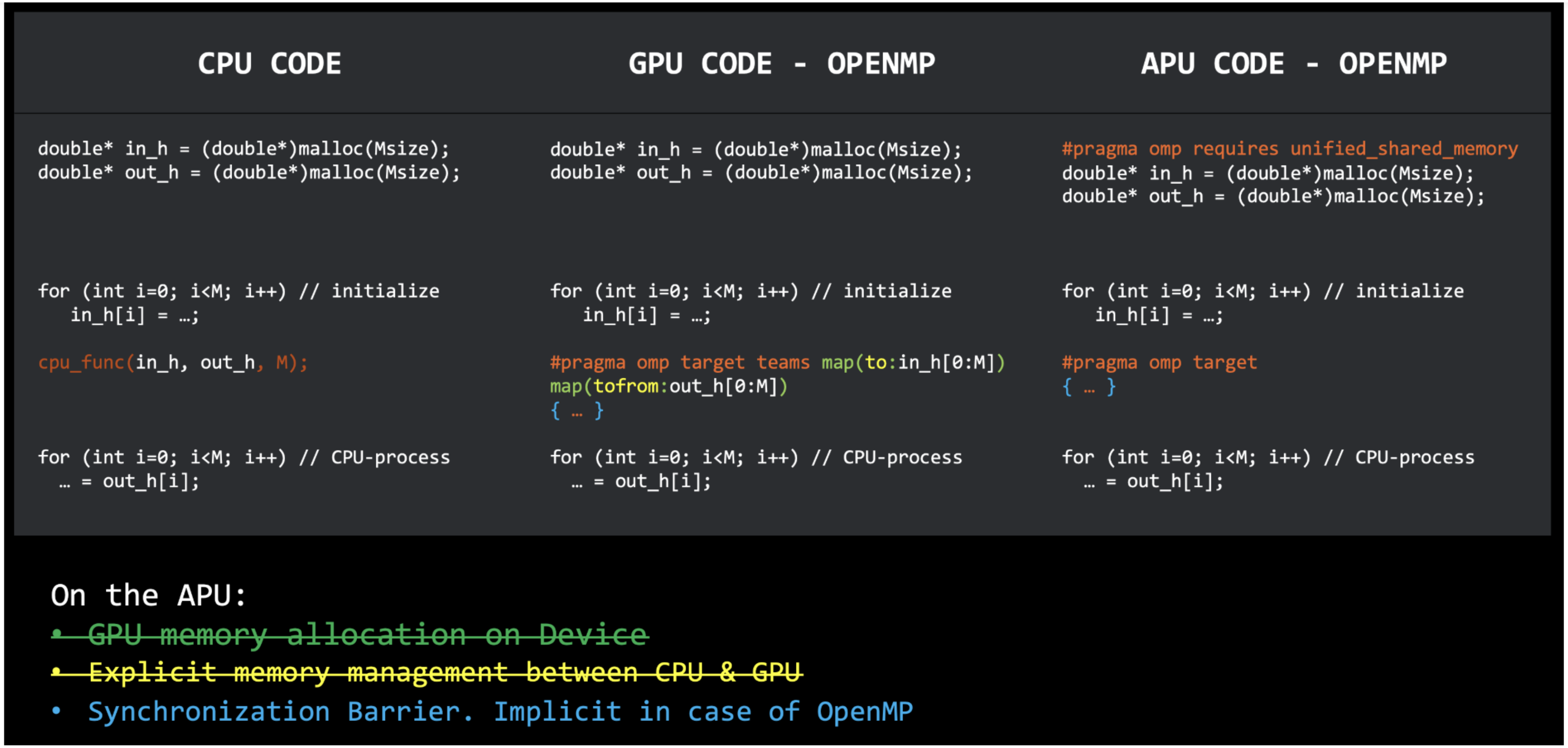
Reference: https://rocm.blogs.amd.com/software-tools-optimization/mi300a-programming/README.html

MI300A: APU Programming Model

- Don’t need to explicitly manage memory as in the case of heterogenous architectures
- Can incrementally change your code, starting with most compute intensive areas. Easier to port complex code regions as memory management is not needed
- Can reuse most of the code – unified code base possible in OpenMP offload approach
- Portable – managed memory support is available on other devices

Although with different performance characteristics

MI300A: APU Programming Model – OpenMP Offload



CPU CODE

```
double* in_h = (double*)malloc(Msize);
double* out_h = (double*)malloc(Msize);


for (int i=0; i<M; i++) // initialize
   in_h[i] = …;

cpu_func(in_h, out_h, M);


for (int i=0; i<M; i++) // CPU-process
  … = out_h[i];
```

GPU CODE - OPENMP

```
double* in_h = (double*)malloc(Msize);
double* out_h = (double*)malloc(Msize);


for (int i=0; i<M; i++) // initialize
   in_h[i] = …;

#pragma omp target teams map(to:in_h[0:M])
map(tofrom:out_h[0:M])
{ … }

for (int i=0; i<M; i++) // CPU-process
  … = out_h[i];
```

APU CODE - OPENMP

```
#pragma omp requires unified_shared_memory
double* in_h = (double*)malloc(Msize);
double* out_h = (double*)malloc(Msize);


for (int i=0; i<M; i++) // initialize
   in_h[i] = …;

#pragma omp target
{ … }

for (int i=0; i<M; i++) // CPU-process
  … = out_h[i];
```

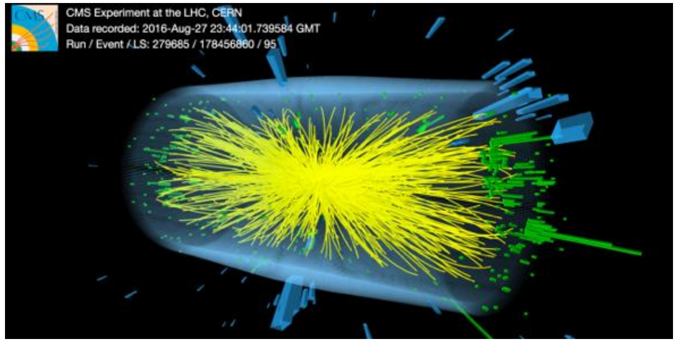
On the APU:

- ~~GPU memory allocation on Device~~
- ~~Explicit memory management between CPU & GPU~~
- Synchronization Barrier. Implicit in case of OpenMP

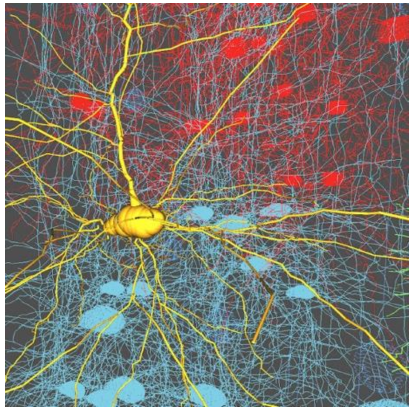
Reference: https://rocm.blogs.amd.com/software-tools-optimization/mi300a-programming/README.html

Cosmos will support a wide range of applications

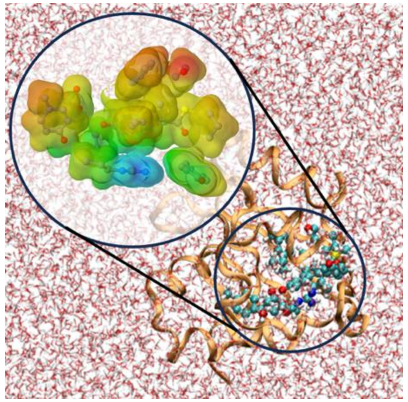
- APU memory architecture will remove need for host/device memory management that often makes porting to accelerators difficult
- The shared memory between CPUs and GPUs will allow for an incremental approach to porting new codes and provide an easier programming path => *no code left behind!*
- Contacted several research groups and communities during proposal development who will collaborate on porting and optimization efforts during testbed phase. Spans several domain science and AI/ML application areas including:
  - Heavily used community codes, such as AMBER (molecular dynamics), Enzo (cosmology), and NEURON (neuroscience);
  - Science gateways, e.g. I-TASSER and the Neuroscience Gateway;
  - Middleware, such as HTCondor and MVAPICH;
  - Software to analyze data from large instruments, such as LIGO and IceCube.


CMS Experiment at the LHC, CERN
Data recorded: 2016-Aug-27 23:44:01.739584 GMT
Run / Event / LS: 279685 / 178456860 / 95

Collision readout from CMS at LHC



3D representation of M1 cortical network, output from NEURON



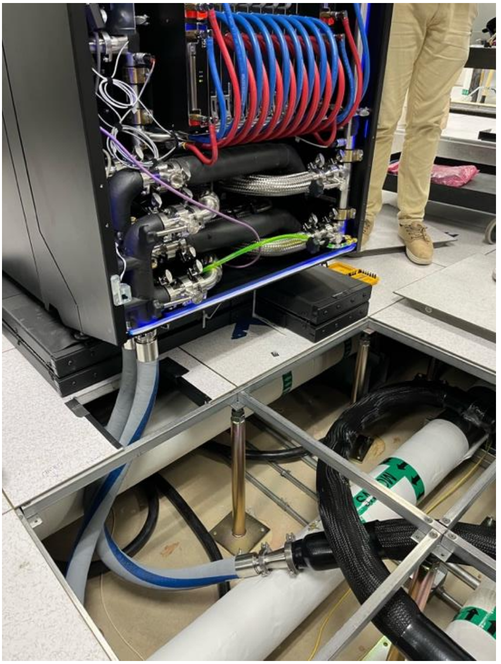
QM electron density at the protein active, AMBER/Quick Code

The project is structured in two phases: a 3-year testbed, followed by a 2-year allocations phase

- **Testbed Phase**
  - Collaborate with research teams (~20 already contacted) to port applications to the MI300A APU
  - Collaborations that specifically target community codes, science gateways, and enabling middleware
  - Project workshops and participation in the AMD User Forum share lessons learned, develop knowledge and best practices
  - External Advisory Board to help recruit research groups, provide guidance to project
- **Allocations Phase**
  - Allocate via NSF-approved process (ACCESS)
  - Lessons learned from Testbed phase inform documentation and training
  - Regular and advanced user support
  - Industry engagement for similar technology evaluation

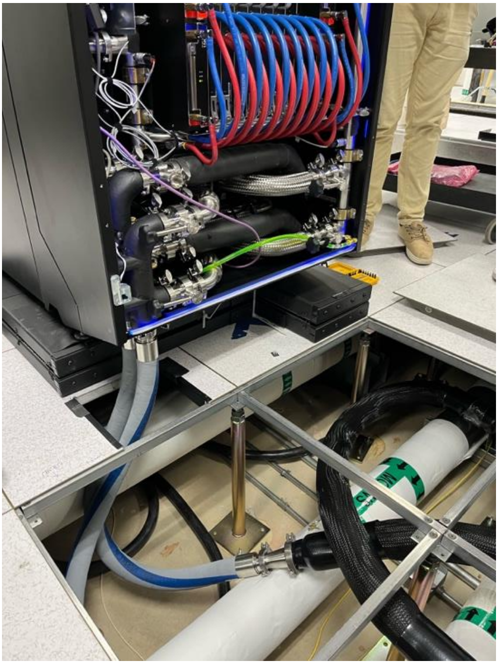
Cosmos: Current Status

- System delivered and installed
- Acceptance benchmarks completed
- Codes/benchmarks tested include:
  - HPL
  - AMBER – compiled from source w/ ROCm/HIP
  - UniFrac – OpenMP offload approach
  - CGYRO – Fortran OpenMP offload + hipFFT
  - IceCube benchmark – OpenCL
  - Pytorch/ML – MegatronLM, VLLM inference
  - MPI latency/bandwidth
- Early user phase completed and transitioning to testbed phase
- NSF review completed



Cosmos: Current Status

- System delivered and installed
- Acceptance benchmarks completed
- Codes/benchmarks tested include:
  - HPL
  - AMBER – compiled from source w/ ROCm/HIP
  - UniFrac – OpenMP offload approach
  - CGYRO – Fortran OpenMP offload + hipFFT
  - IceCube benchmark – OpenCL
  - Pytorch/ML – MegatronLM, VLLM inference
  - MPI latency/bandwidth
- Early user phase completed and transitioning to testbe
- NSF review completed



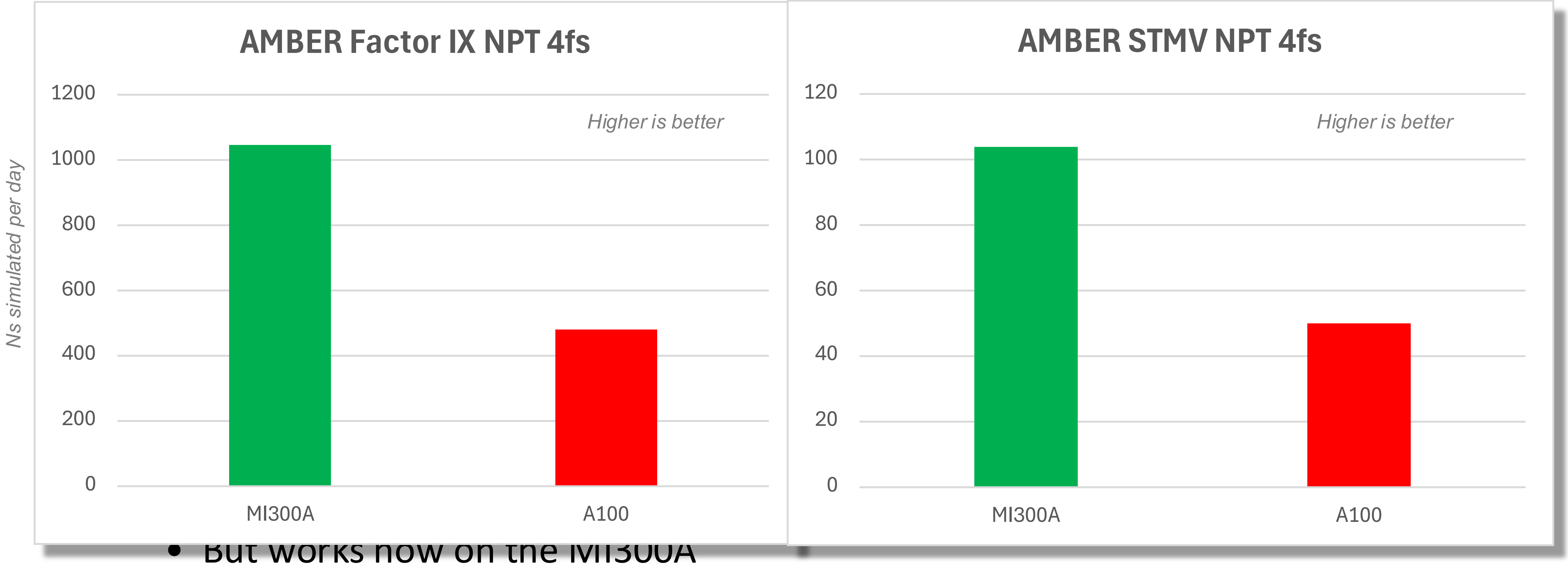
All “true GPU” applications. Will explore APU-optimized avenues in the near future.

Example app - AMBER

- Popular biomolecular simulation tool
- Already had a GPU port
  - Based on NVIDIA CUDA (with significant optimizations)
- The code was thus “hippified”
  - Converting CUDA code into HIP
  - AMD provides several tools to do this
- Was not exactly trivial
  - Several assumptions about NVIDIA GPUs did not cleanly translate
  - But works now on the MI300A

Example app - AMBER

Results from early user benchmarks. Results may change/improve as we better understand the system



- But works now on the MI300A

Acknowledgement: AMBER was benchmarked on the PNRP system.

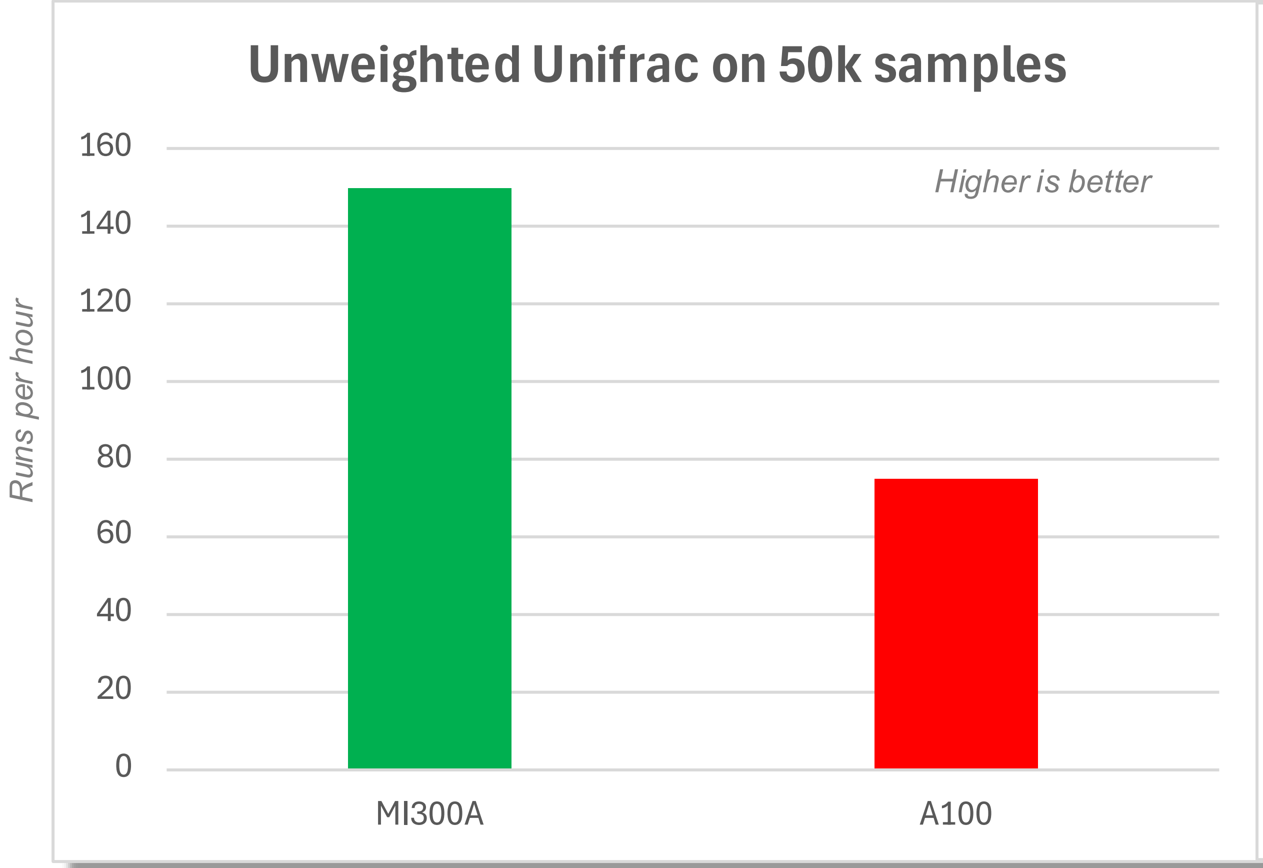
Example app - Unifrac

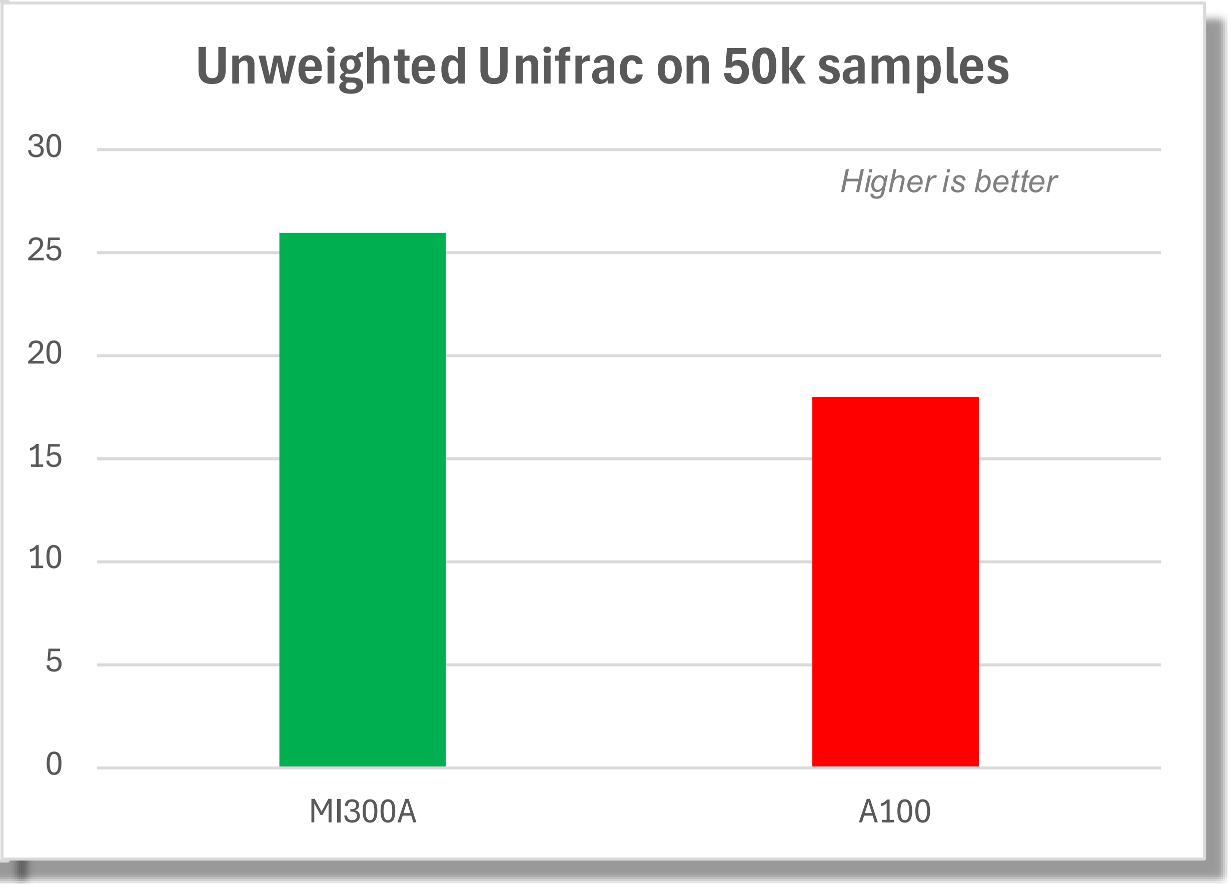
- Popular microbiome distance metric library
  - C++ based
- Already had a GPU port
  - Originally based on OpenACC
  - Recently added OpenMP offload
    - Almost identical semantics as OpenACC
    - Just different syntax
- Cosmos first AMD GPU-based target
  - But worked out-of-the-box on the MI300A, with a simple recompilation

AMD does not support OpenACC

Example app - Unifrac

Results from early user benchmarks. Results may change/improve as we better understand the system





Acknowledgement: Unifrac was benchmarked on the PNRP system.

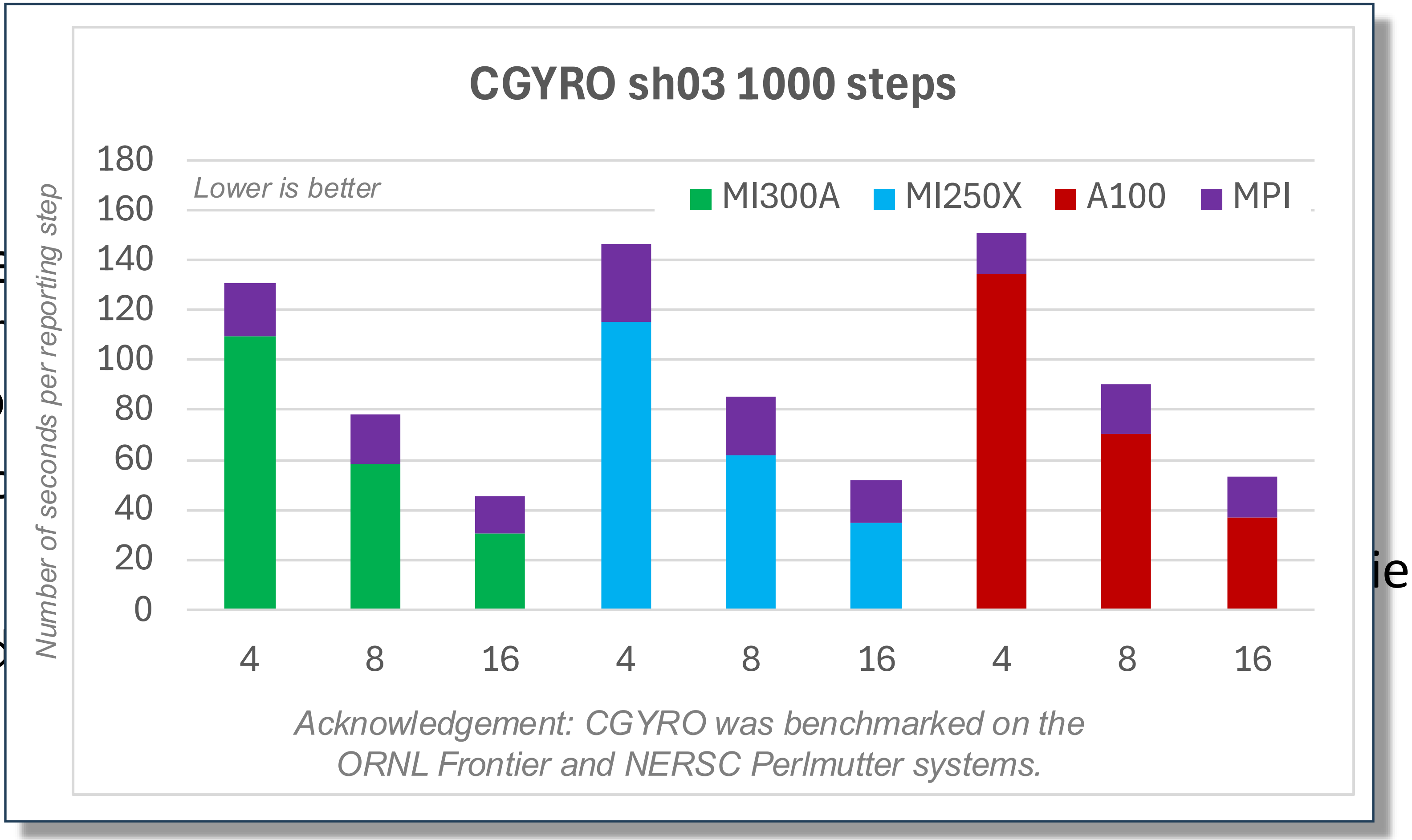
Example app - CGYRO

- Popular Fusion simulation HPC tool
  - Fortran + vendor FFT + MPI
- Many years of experience on HPC GPU-based systems
  (going all the way back to OLCF Titan)
  - GPU offload originally OpenACC based
  - With cuFFT for batched-FFT processing
- Already had experience on AMD GPUs (MI250X on OLCF Frontier)
  - Added OpenMP offload support + hipFFT as part of that
- Worked on Cosmos/MI300A out-of-the-box

Example app - CGYRO

Results from early user benchmarks. Results may change/improve as we better understand the system

- Popular
  - Fortra
- Many ye
  (going all th
  - GPU o
  - With c
- Already ier)
  - Added
- Worked



Acknowledgement: CGYRO was benchmarked on the ORNL Frontier and NERSC Perlmutter systems.

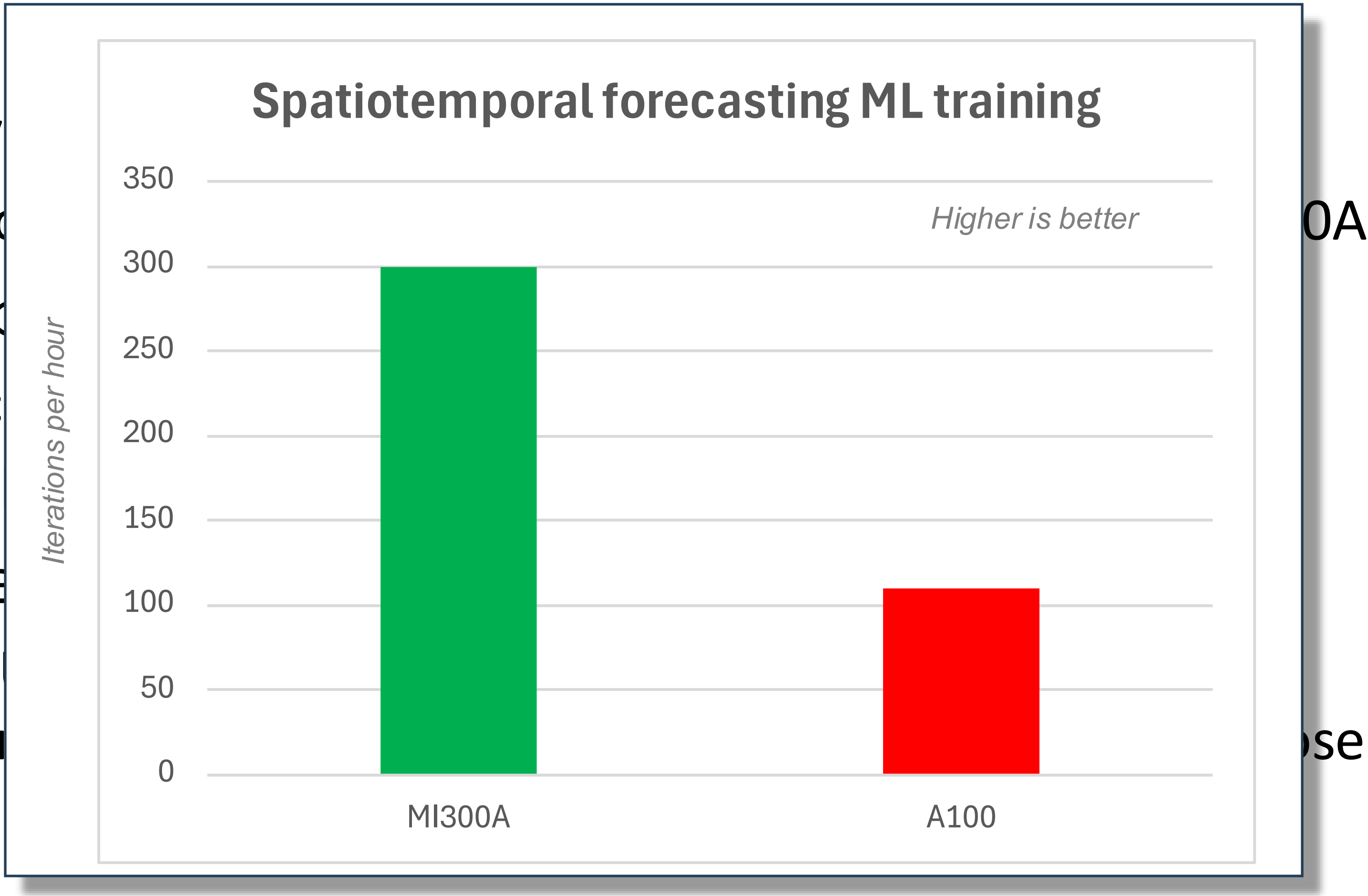
AI/ML applications

- AMD provides both PyTorch and Tensorflow packages
  - Already optimized for the AMD hardware, including the MI300A
- Available both as
  - Docker containers
  - Pip install
- No changes needed in the user code
  - AMD GPUs advertise “CUDA support”
    (ROCm under the hood, but applications are not directly exposed to it)

AI/ML applications

Results from early user benchmarks. Results may change/improve as we better understand the system

- AMD prov
  - Already c 0A
- Available b
  - Docker c
  - Pip insta
- No change
  - AMD GP (ROCm u sed to it)



Acknowledgement: The ML training was benchmarked on the PNRP system.

Summary

- MI300A is the world’s first high-performance APU. *Cosmos* will be the first to introduce this architecture to the NSF community
- *Cosmos* will enable the open research community exploit this innovative and powerful accelerator technology
- The testbed is also bringing in new storage technologies to SDSC with VAST. Both VAST and Ceph have potential to be shared across multiple systems and we will be exploring options in the testbed phase.
- System has completed reliability testing and early user phase. In transition to testbed phase after NSF review
- Thanks to the co-PIs and many others at SDSC who contributed to the proposal to make this possible!