COSMOS: INNOVATIVE SYSTEM FEATURING AMD MI300A APUs
Experiences Running Applications Using MVAPICH-PLUS

Mahidhar tatineni
San Diego Supercomputer Center (SDSC)
MVAPICH USER GROUP MEETING
August 20, 2025

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

Reference: PEARC25 workshop presentation

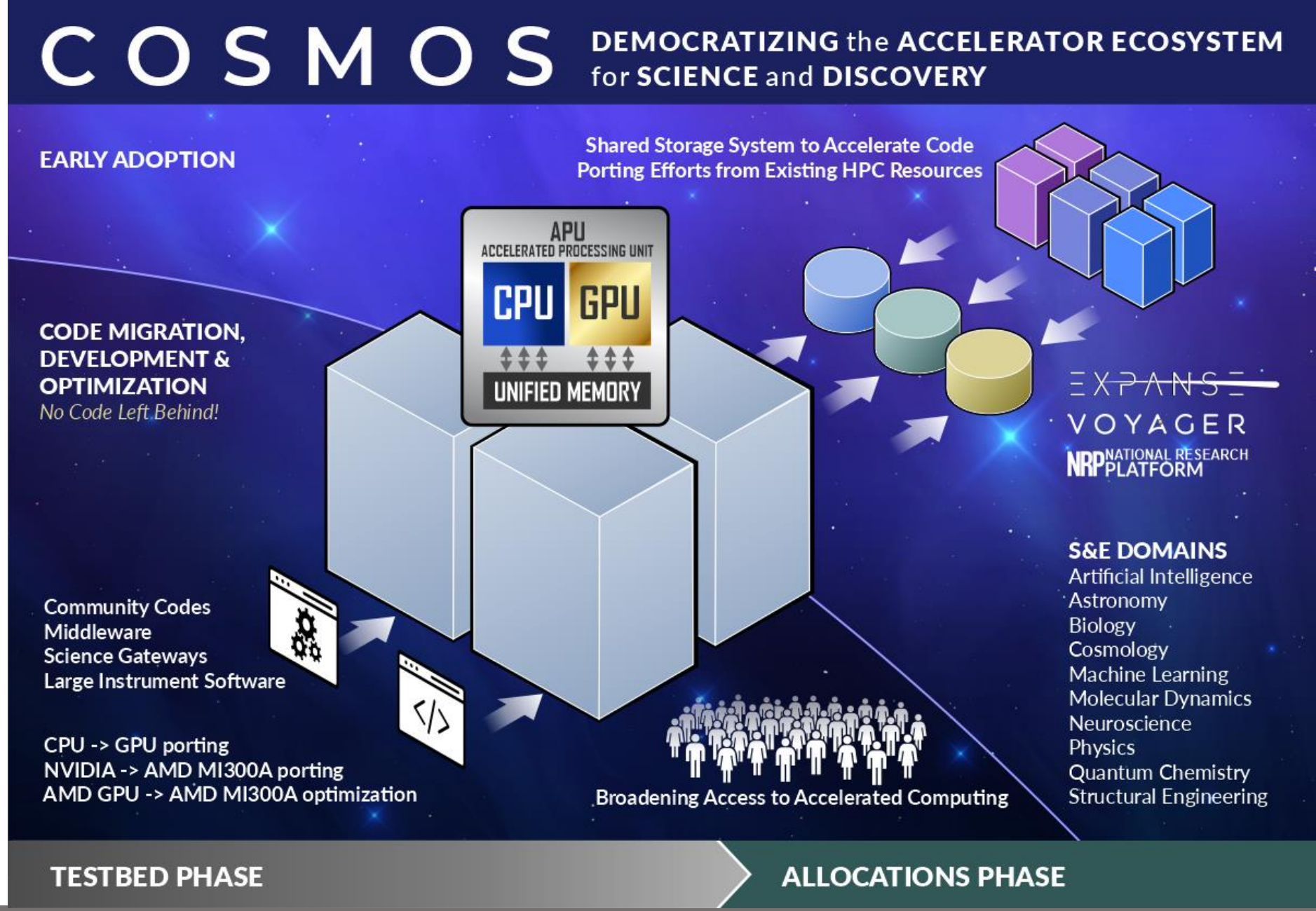
NSF Award 2404323

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

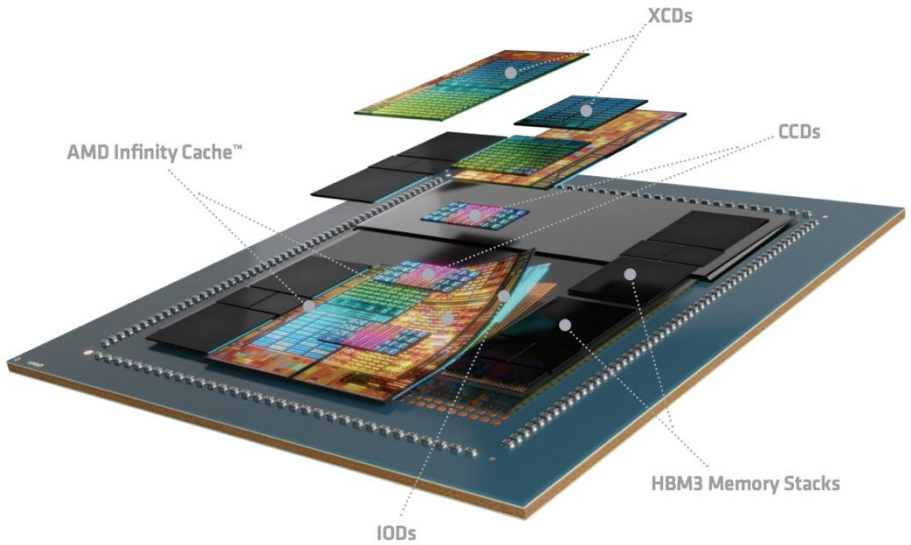
Senior Personnel:
Robert Sinkovits
Mary Thomas

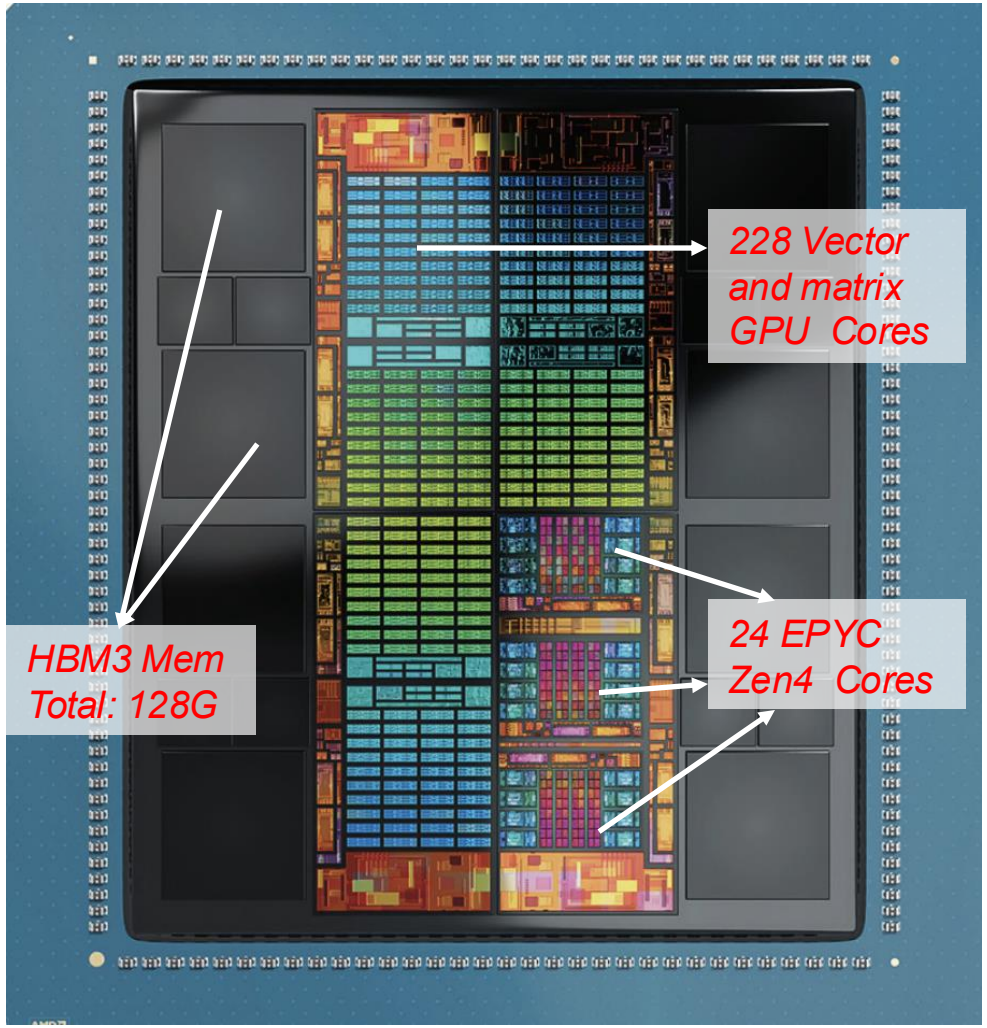


Cosmos: Democratizing the Accelerator Ecosystem for Science and Discovery

- 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 (~500TB) 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.

AMD MI300A Architecture

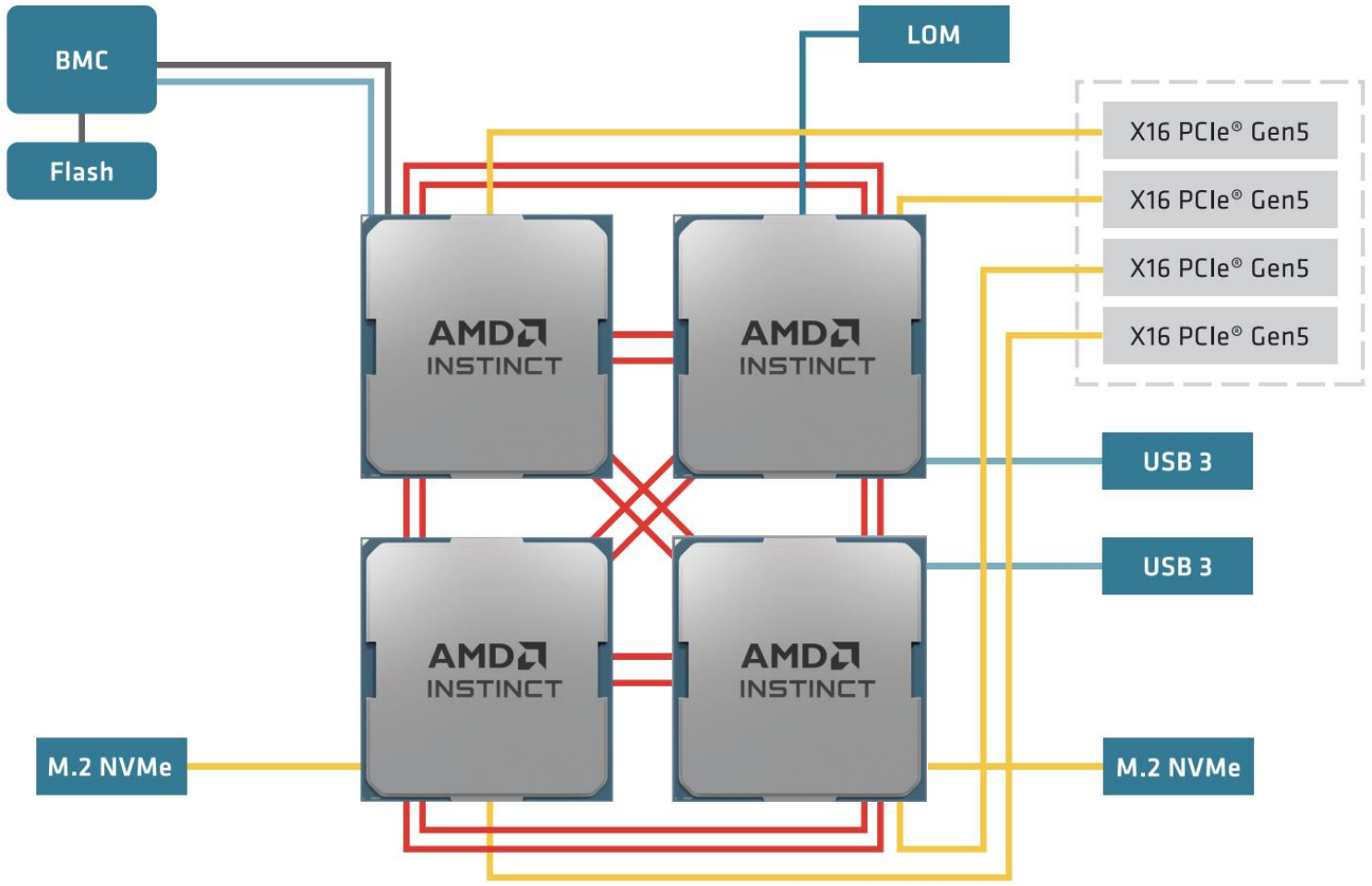




- *MI300A APU combines CPU, GPU, and memory on one package. Six accelerated compute dies (XCDs) combined with 3 chiplets with 8 zen4 cores each.*
- *128 GB of HBM3 memory shared coherently between CPUs and GPUs. 5.3 TB/s peak throughput.*
- *256 MB AMD Infinity Cache (last level) shared between XCDs 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!)*

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*

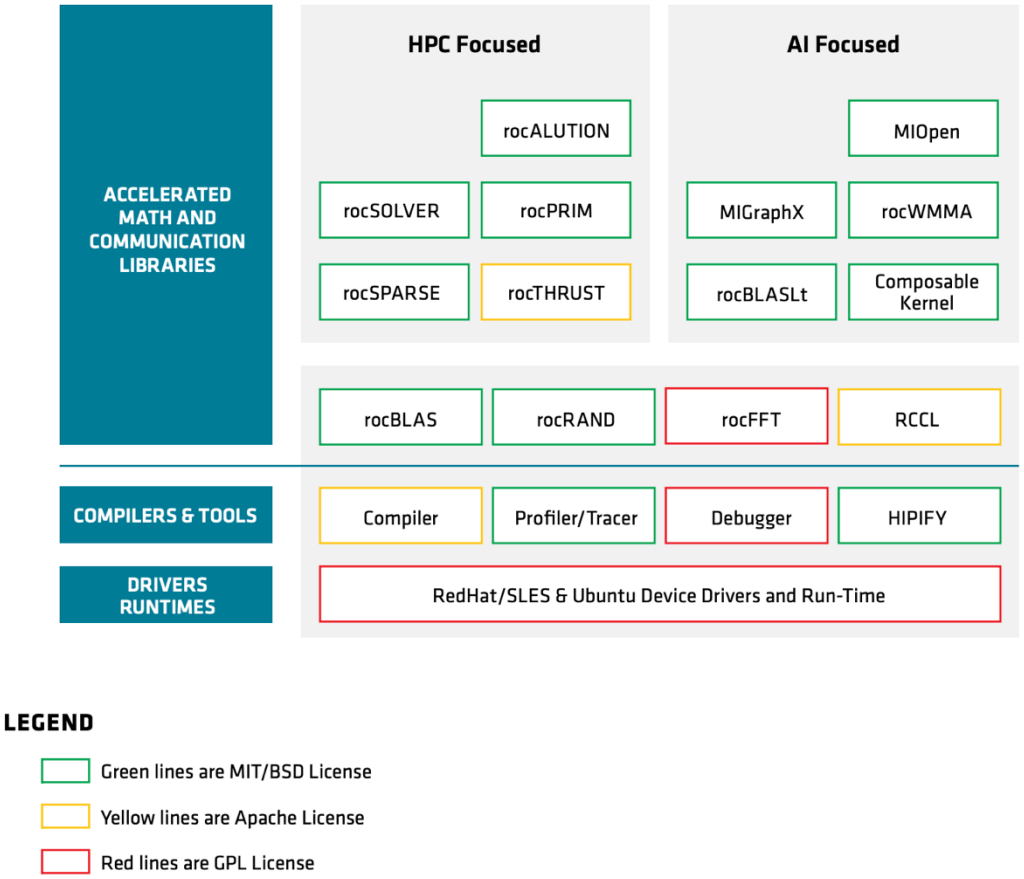
Reference: https://www.hpe.com/psnow/doc/a00094635enw

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.



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	GPU CODE - HIP	APU CODE - HIP

`double* in_h = (double*)malloc(Msize);`
`double* out_h = (double*)malloc(Msize);`
	`double* in_h = (double*)malloc(Msize);`
	`double* out_h = (double*)malloc(Msize);`
	`hipMalloc(&in_d, Msize);`
	`hipMalloc(&out_d, Msize);`
		`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++) // 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++) // initialize`
		`  in_h[i] = …;`
		`gpu_func<< >>(in_h, out_h, M);`
		`hipDeviceSynchronize();`

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

`double* in_h = (double*)malloc(Msize);`
`double* out_h = (double*)malloc(Msize);`
	`double* in_h = (double*)malloc(Msize);`
	`double* out_h = (double*)malloc(Msize);`
		`#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] = …;`	`for (int i=0; i<M; i++) // initialize` `  in_h[i] = …;`	`for (int i=0; i<M; i++) // initialize` `  in_h[i] = …;`

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

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

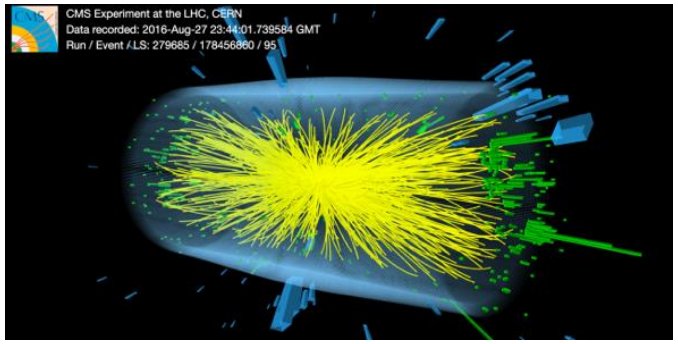
On the APU:

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

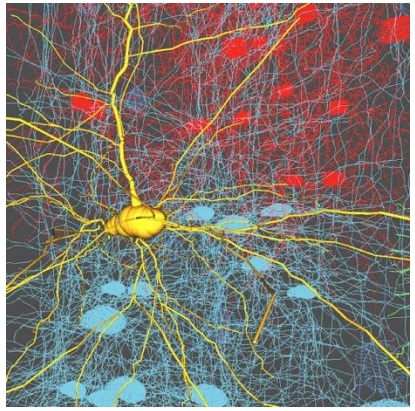
Reference: https://rocm.blogs.amd.com/software-tools-optimization/mi300a-programming/README.html

Cosmos will support a wide range of applications

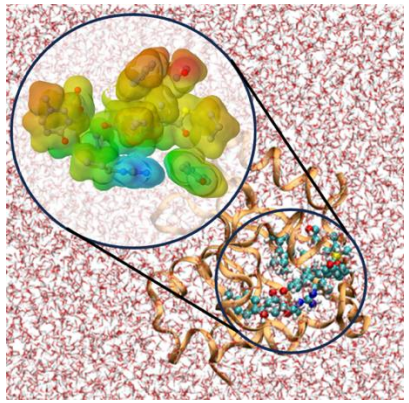
- APU memory architecture will remove need for host/device memory management that often makes porting to accelerators difficult
- The shared memory between CPUs and GPUs will allow for an incremental approach to porting new codes and provide an easier programming path => *no code left behind!*
- Contacted 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.



Collision readout from CMS at LHC



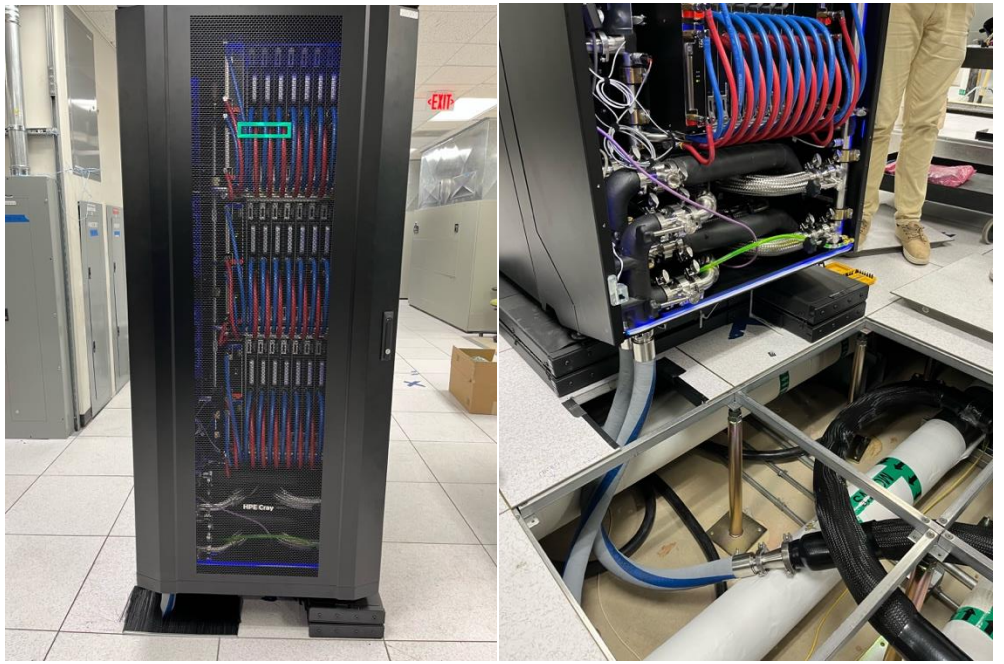
3D representation of M1 cortical network, output from NEURON



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

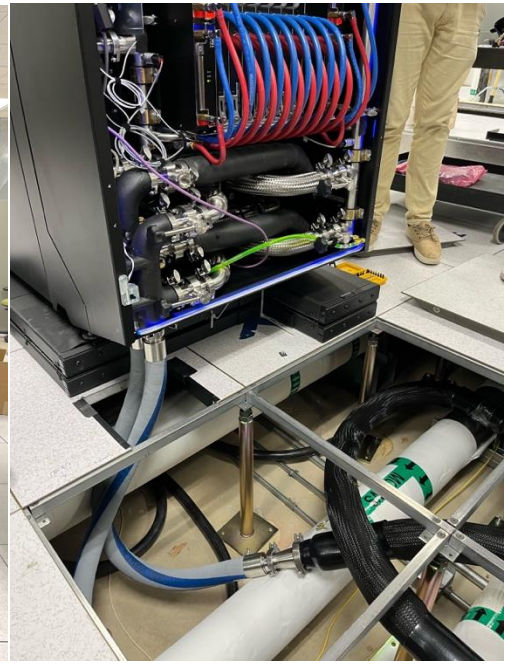
Cosmos: Current Status

- 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
- NSF review completed
- Early user phase completed. Transitioned to testbed phase



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
- NSF review completed
- Early user phase completed. Transitioned to testbed phas



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

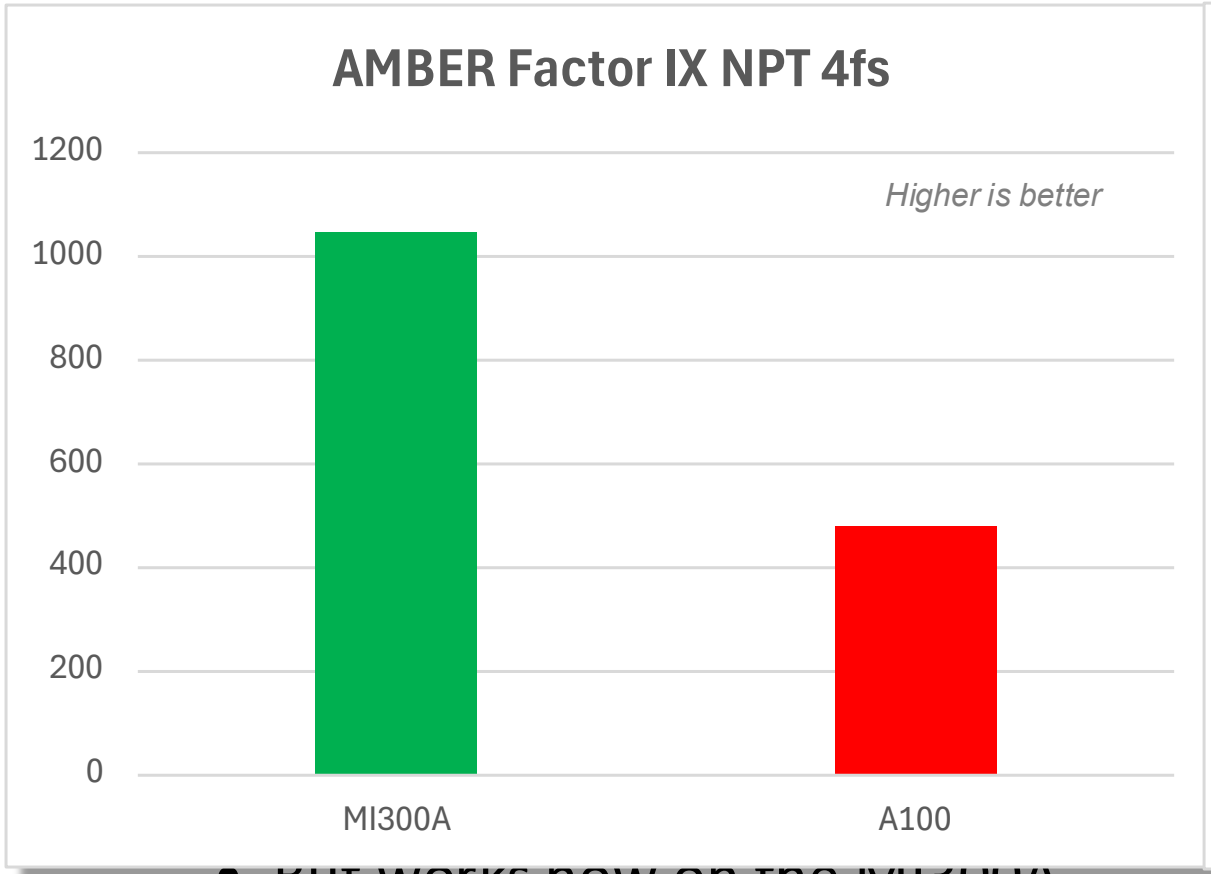
Applications on Cosmos

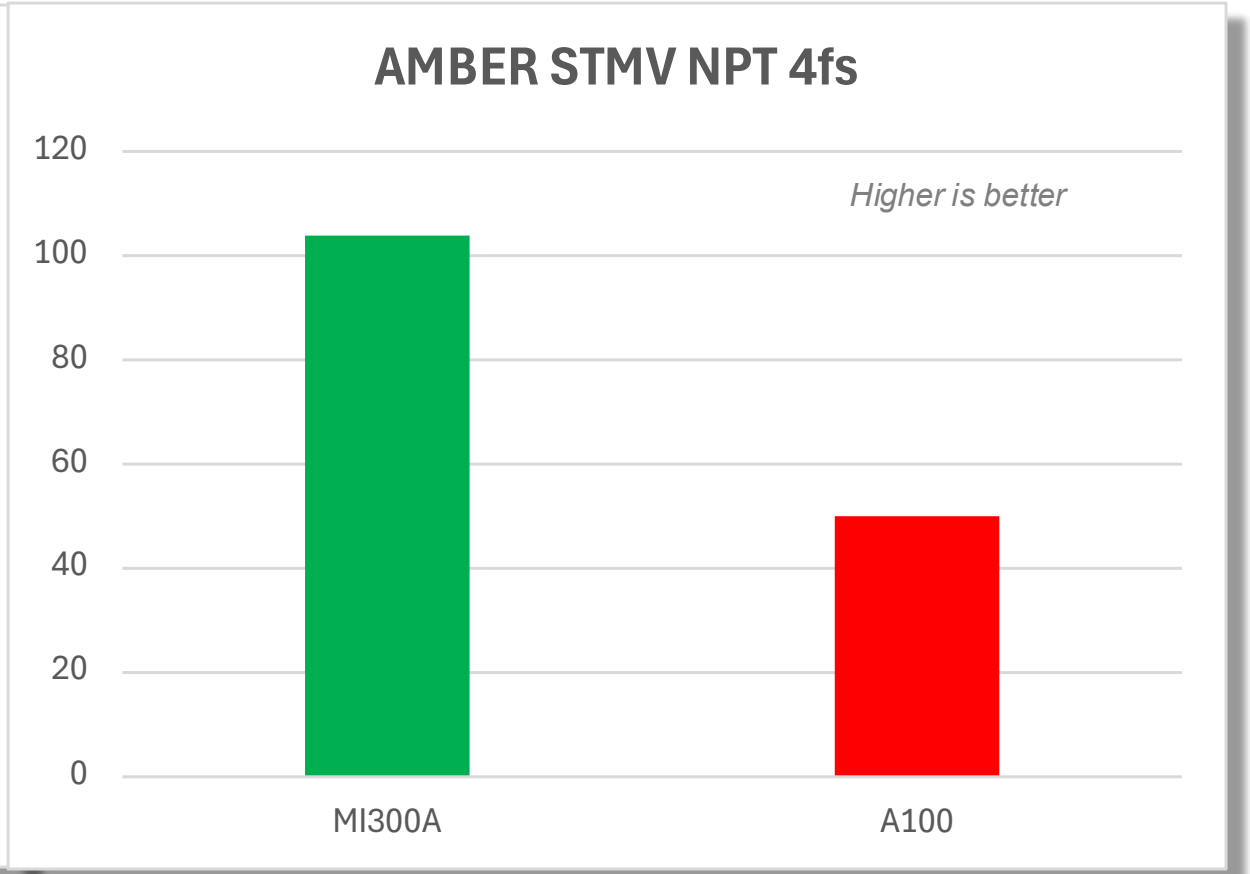
Example app - AMBER

- 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



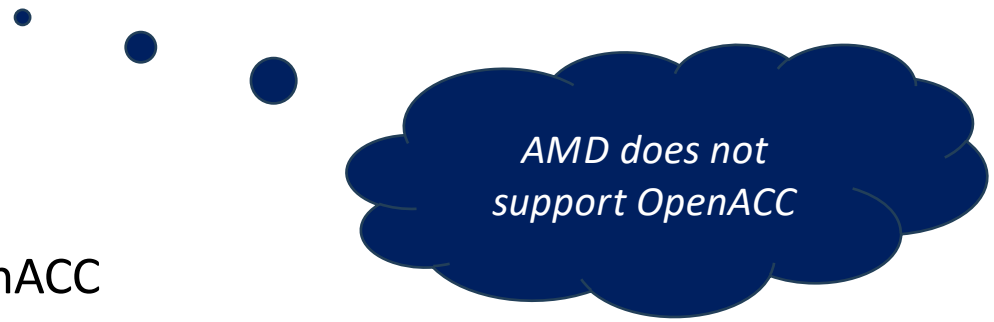


- But works now on the MI300A

Acknowledgement: AMBER was benchmarked on the PNRP system.

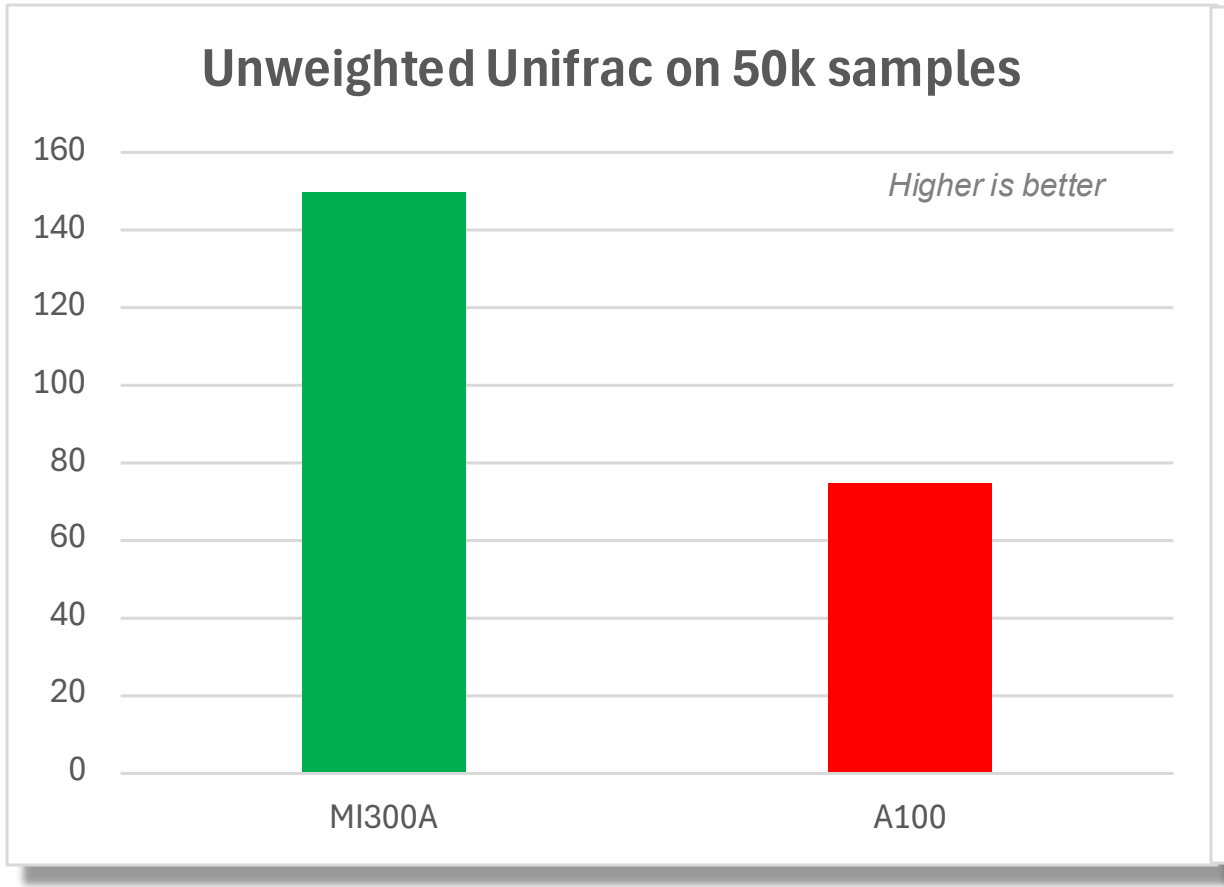
Example app - Unifrac

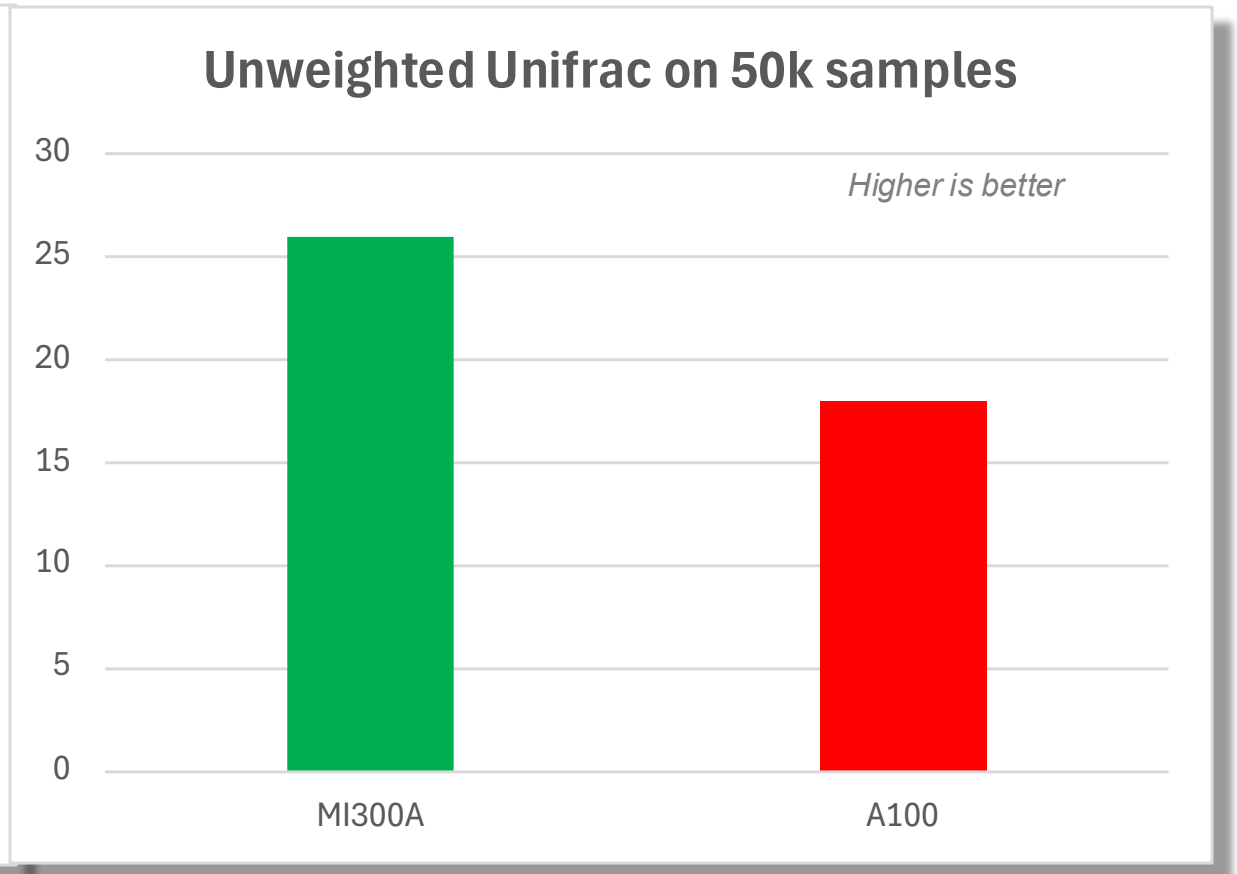
- Popular microbiome distance metric library
  - C++ based
- Already had a GPU port
  - Originally based on OpenACC
  - Recently added OpenMP offload
    - Almost identical semantics as OpenACC
    - Just different syntax
- Cosmos first AMD GPU-based target
  - But worked out-of-the-box on the MI300A,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



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

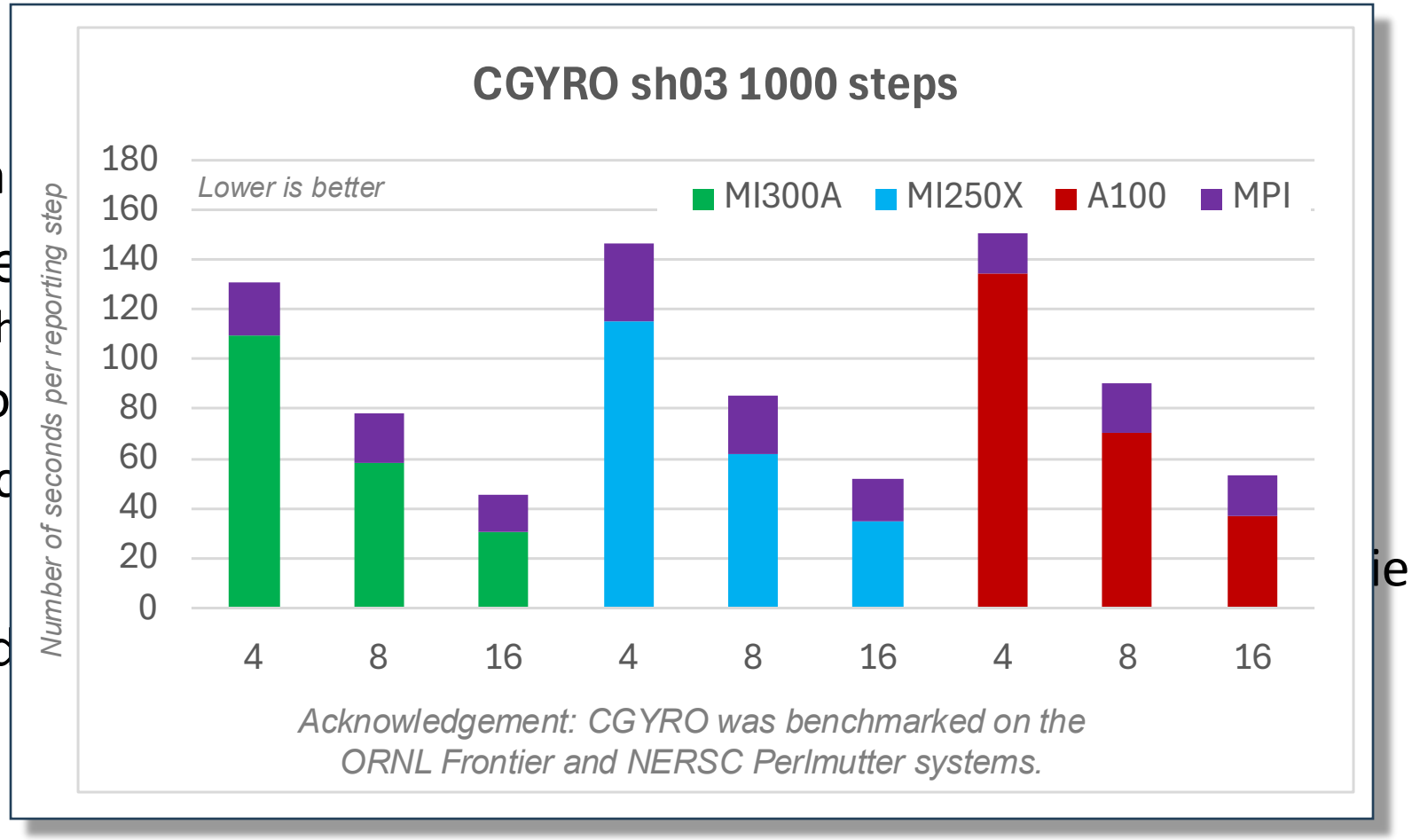
- Popular Fusion simulation HPC tool
  - Fortran + vendor FFT + MPI
- Many years of experience on HPC GPU-based systems (going all the way back to OLCF Titan)
  - GPU offload originally OpenACC based
  - With cuFFT for batched-FFT processing
- Already had experience on AMD GPUs (MI250X on OLCF Frontier)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

ier)

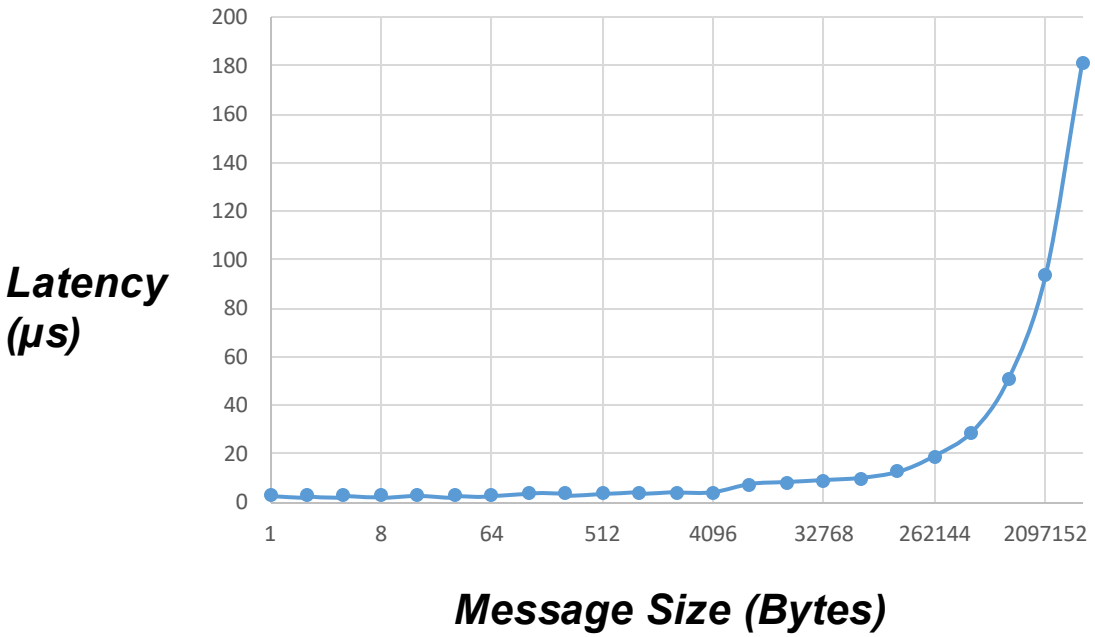


MVAPICH-PLUS on Cosmos

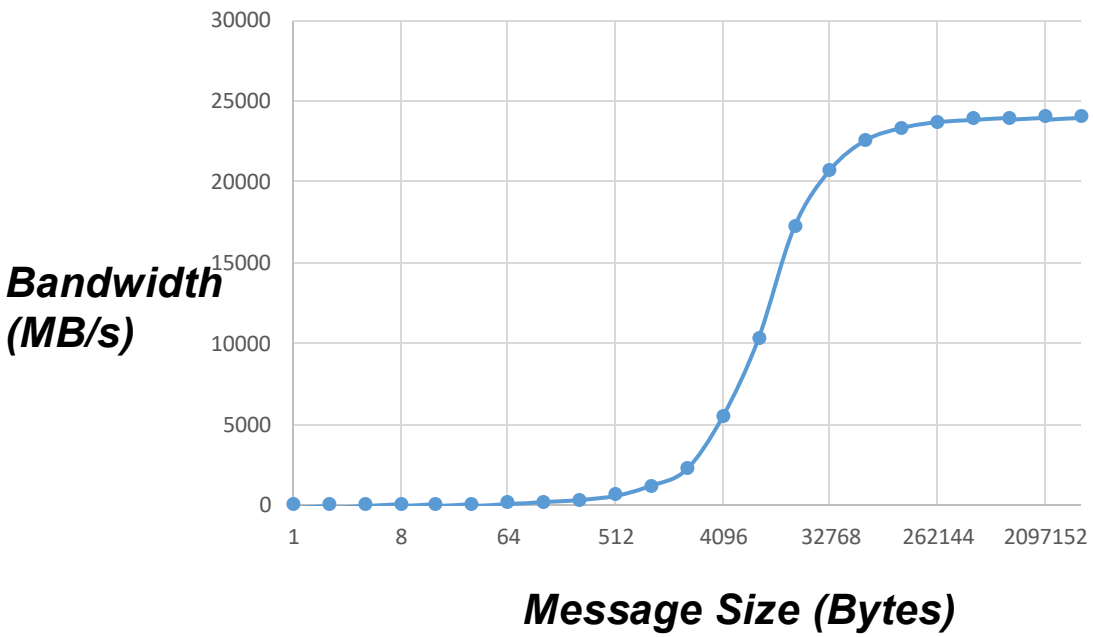
MVAPICH-PLUS on Cosmos

- MVAPICH-PLUS developers were part of the early user testing on Cosmos
- Current experiences are testing with GNU compilers and MVAPICH-PLUS 4.1.
- The Cray programming environments already have MPI integrated compilers, so it makes it tricky to separate the MVAPICH-PLUS builds.
- For GNU Compilers, the build picked up the default compilers in the OS (v7.5.0). This does result in an older gnu version being used.
- Looking into workarounds that will pick up module provided version of both gnu and cray compilers without conflict with existing Cray MPI implementations
- Very preliminary testing results! Tested functionality with OSU benchmarks and LAMMPS application

OSU Benchmark Results (pt2pt)



OSU pt2pt Latency Test



OSU pt2pt Bandwidth Test

LAMMPS build using MVAPICH-PLUS

- Initial build process was picking up Cray MPI automatically! Explicit cmake options were needed to work around this
- Two different builds attempted and tested
  - LAMMPS with MPI, and GPU (HIP based) packages
  - LAMMPS with OpenMP, MPI, and Kokkos packages
- CMAKE options

```
cmake -DCMAKE_C_COMPILER=`which mpicc` -DCMAKE_CXX_COMPILER=hipcc -
DCMAKE_Fortran_COMPILER=gfortran -D BUILD_MPI=yes -D PKG_GPU=on -D GPU_API=HIP -D
HIP_ARCH=gfx942 -DMPI_C_COMPILER=`which mpicc` -DMPI_CXX_COMPILER=`which mpicxx`
../cmake
```

```
cmake -DCMAKE_C_COMPILER=gcc -DCMAKE_CXX_COMPILER=hipcc -
DCMAKE_Fortran_COMPILER=gfortran -D BUILD_MPI=yes -D Kokkos_ARCH_AMD_GFX942_APU=yes -
D Kokkos_ENABLE_HIP=yes -D Kokkos_ENABLE_OPENMP=yes -D Kokkos_ARCH_ZEN4=yes -D
PKG_KOKKOS=on -DMPI_C_COMPILER=`which mpicc` -DMPI_CXX_COMPILER=`which mpicxx` ../cmake
```

LAMMPS Testing Details

- Testing modes:
  - SPX (Single Partition X-celerator): The default mode on Cosmos. All six GPU XCDs are grouped as a single logical GPU device with all compute and memory resources available => 4 “GPUs” per node
  - TPX (Triple Partition X-celerator): Divides the GPU complex into three partitions, each containing two XCDs and providing three logical GPU devices per socket => 12 “GPUs” per node
- The LAMMPS GPU package build was tested in both SPX and TPX modes. The Kokkos based build failed on initialization and further debugging is needed. Workaround suggested (MPIR_CVAR_CH4_CMA_ENABLE=0) worked.
- These are very preliminary results! With proper binding and MPI/OpenMP and HIP device usage we expect to see performance improve further
- Benchmarking will continue in testbed phase
- The LAMMPS benchmark case chosen is atomic fluid with Lennard-Jones potential with 512,000 atoms run for 100,000 steps.

LAMMPS Output SPX and TPX modes

SPX Mode
All 4 APUs in use
=> 4 GPU devices

```
--------------------------------------------------------------------------
- Using acceleration for lj/cut:
-  with 1 proc(s) per device.
-  with 1 thread(s) per proc.
-  Horizontal vector operations: ENABLED
-  Shared memory system: No
--------------------------------------------------------------------------
Device 0: AMD Instinct MI300A, 228 CUs, 1.3e+02/1.3e+02 GB, 2.1 GHZ (Mixed Precis
ion)
Device 1: AMD Instinct MI300A, 228 CUs, 2.1 GHZ (Mixed Precision)
Device 2: AMD Instinct MI300A, 228 CUs, 2.1 GHZ (Mixed Precision)
Device 3: AMD Instinct MI300A, 228 CUs, 2.1 GHZ (Mixed Precision)
--------------------------------------------------------------------------
```

TPX Mode
2 APUs in use
=> 6 GPU devices

```
--------------------------------------------------------------------------
- Using acceleration for lj/cut:
-  with 1 proc(s) per device.
-  with 1 thread(s) per proc.
-  Horizontal vector operations: ENABLED
-  Shared memory system: No
--------------------------------------------------------------------------
Device 0: AMD Instinct MI300A, 76 CUs, 42/43 GB, 2.1 GHZ (Mixed Precision)
Device 1: AMD Instinct MI300A, 76 CUs, 2.1 GHZ (Mixed Precision)
Device 2: AMD Instinct MI300A, 76 CUs, 2.1 GHZ (Mixed Precision)
Device 3: AMD Instinct MI300A, 76 CUs, 2.1 GHZ (Mixed Precision)
Device 4: AMD Instinct MI300A, 76 CUs, 2.1 GHZ (Mixed Precision)
Device 5: AMD Instinct MI300A, 76 CUs, 2.1 GHZ (Mixed Precision)
--------------------------------------------------------------------------
```

LAMMPS Runs using Kokkos based Build

- Run using MVAPICH-PLUS failed with internal MPI error. Needs further investigation (have not provided info to MVAPICH team yet)
- We will also test with Cray MPI compilers to see if similar error occurs
- Error:

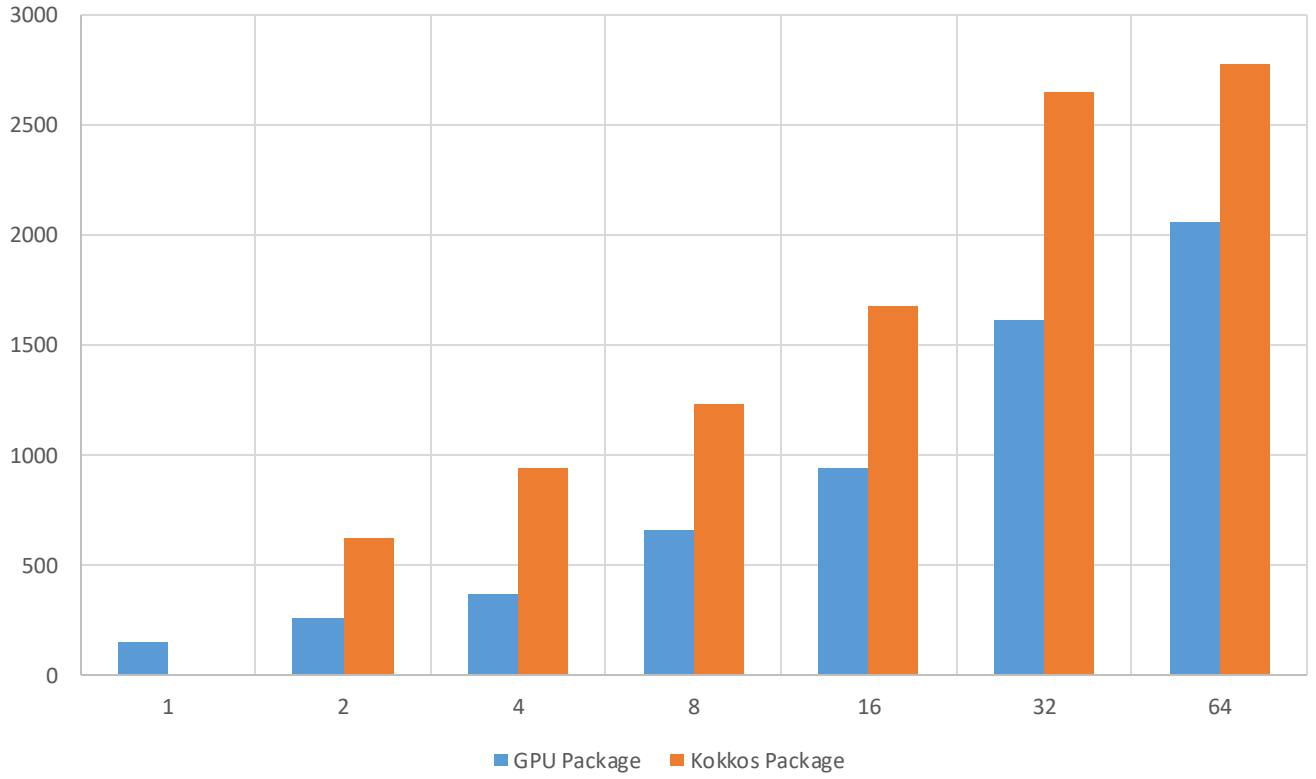
```
Abort(472044815) on node 0 (rank 0 in comm 0): Fatal error in internal_Wait: Oth
er MPI error, error stack:
internal_Wait(104).............: MPI_Wait(request=0x7ffc3c70427c, status=0x1) fa
iled
MPIR_Wait(755).................:
MPIDI_IPCI_handle_lmt_recv(345):
MPIDI_CMA_copy_data(84)........:
copy_iovs(420).................: process_vm_readv failed (errno 14)
```

- Workaround suggested (MPIR_CVAR_CH4_CMA_ENABLE=0) worked!

LAMMPS Results – SPX Mode

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

Performance #Steps/s



#GPU Devices (=#APUs in SPX mode)

The LAMMPS benchmark case chosen is atomic fluid with Lennard-Jones potential with 512,000 atoms run for 100,000 steps.

LAMMPS Results SPX vs TPX, Single Node

The LAMMPS benchmark case chosen is atomic fluid with Lennard-Jones potential with 512,000 atoms run for 100,000 steps.

#APUs (#GPUs)	SPX Mode	TPX Mode (3 GPUs per APU)	A100 Results (PNRP)
1 (SPX:1; TPX:3)	153.458	376.606	-
2 (SPX:2, TPX:6)	262.088	463.543	-
3 (TPX: 8)	-	**492.670**	-
4 (SPX:4, TPX:12)	374.849	192.498	286

- Benchmarks run with only 1 MPI task per GPU device
- Very preliminary results! Lot of optimizations feasible with more MPI tasks per APU and also OpenMP threads
- Saw low GPU device utilization during SPX runs => room for improvement with more MPI tasks. Probably explains better TPX results.

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)

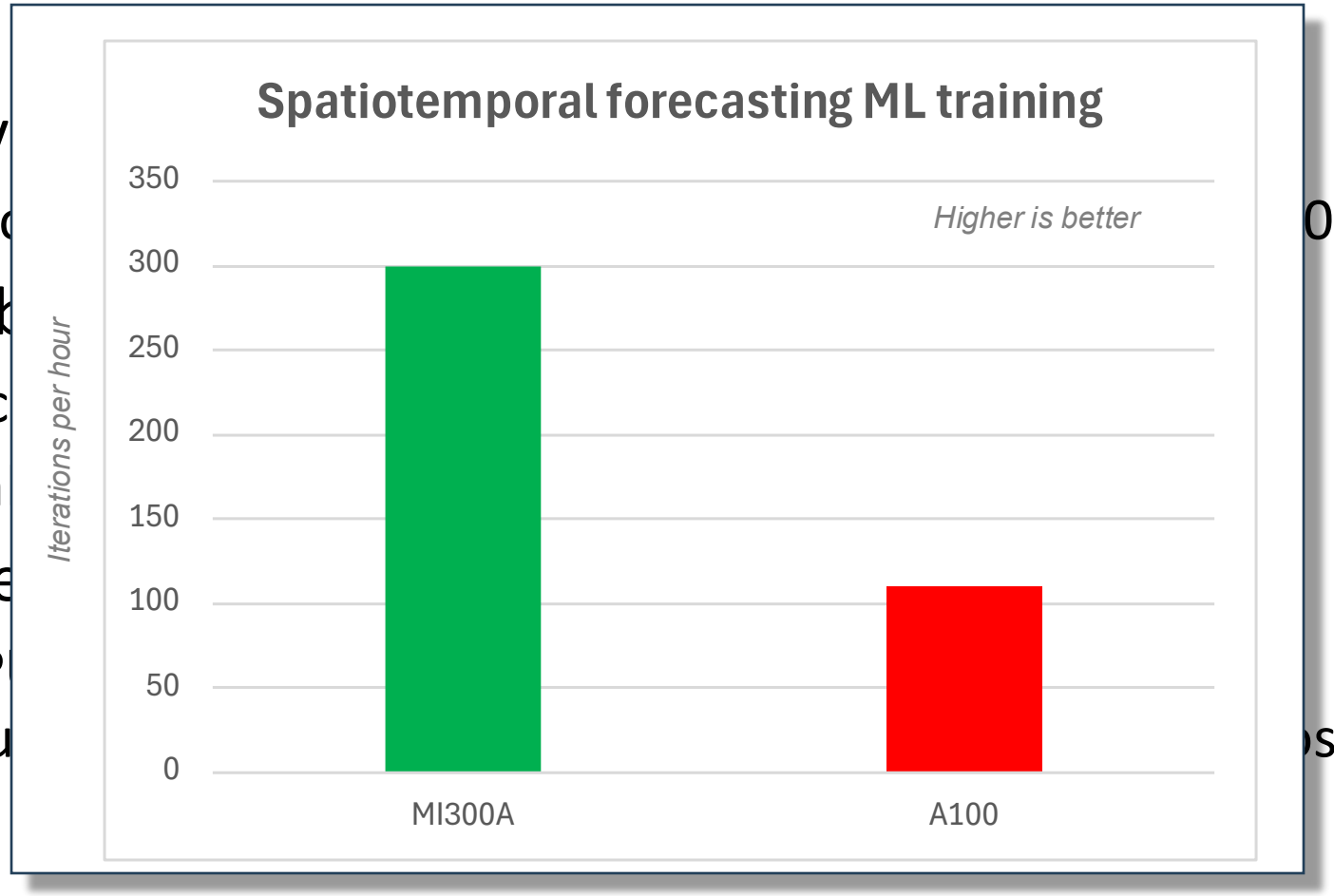
AI/ML applications

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

- AMD prov
  - Already
- Available
  - Docker c
  - Pip insta
- No change
  - AMD GP
    (ROCm u

0A

sed to it)



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

Summary

- MI300A is an innovative 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, early user phase, and is currently in testbed phase.
- Early results from MVAPICH-PLUS are promising. Lot of testing and benchmarking to come! Hope to enable integration with latest versions of both gnu and cray compilers.
- Thanks to the co-PIs and many others at SDSC who contributed to the proposal to make this possible!
- Thanks to Dr. Panda and the MVAPICH team for continuing to provide performant MPI versions for any innovative hardware we make available to them ☺