



# AMD Workshop Introduction

**Presenter: Bob Robey**  
**Datacenter Solutions Group**  
**AMD Global Training Team**  
**Oct 28-30, 2025**

**AMD**   
together we advance\_

# Training Workshop Guidelines

## – help us create a positive environment

Please help us create a positive collaborative environment for this event. Our top goal is to attract and build the community in machine learning and high performance computing. Please consider not only the technical aspects of your interactions, but also how it affects other participants.


- Start your comments with a positive statement before your comment or question
- Give equal time to all participants
- Please, help others with technical issues and without negative comments
- And most importantly, respect all participants, regardless of background, language, culture

# **Introduction to computing systems for exercises and LLM access**


# Contacts for system issues – aac6, gpu-60 and Open WebUI

- Use the dedicated Discord channel for reporting any issues you are encountering with log in:

## Welcome to # -hpc-on-amd-cloud-resources!

This is the start of the #  -hpc-on-amd-cloud-resources channel. All information about cloud access, course materials, slides, recordings, and reference documents will be shared here.

- Check regularly for updates
- Use this channel to download files and resources
- Get help on accessing the AMD cloud resources
- For questions about the courses, use #ai-on-amd-qa

Happy learning! 

- Bob Robey [Bob.Robey@amd.com](mailto:Bob.Robey@amd.com) and Giacomo Capodaglio [Giacomo.Capodaglio@amd.com](mailto:Giacomo.Capodaglio@amd.com) are in charge of user onboarding and assistance.

# Contacts for questions on the material


- Use the dedicated Discord channel for questions about specific topics (DMs are also OK):

## Welcome to # ? -hpc-on-amd-qa!

This is the start of the # ? -hpc-on-amd-qa channel. This channel is dedicated to questions and answers with the course trainers.

### Guidelines:

- Post clear, specific questions
- Check if your question has already been answered
- Respect other participants and trainers

Trainers will respond as soon as possible. Thank you for keeping it organized! 

# Computing systems access notes

## Account requests

- We'll be using ssh public keys for access to the computing systems
  - Need ssh public keys, preferred username, email, address (business), organization name
  - Information will only be used to create accounts and for account issues
- To generate a ssh key
  - `ssh-keygen -t ed25519`
    - Hit return and set password if preferred
  - `cat $HOME/.ssh/id_ed25519.pub`
    - Send the string for creating account access
  - You can do this from the terminal in Windows, from wsl, from Linux or MacOS
  - There are apps (particularly Windows) that purport to make this easier, but partially obscure the process
- Accounts will stay active for some time after the workshop for participants to try the exercises when it is convenient

# Logging in

- `ssh <username>@aac6.amd.com -i <identityfile> # aac6`
- `ssh <username>@193.143.78.200 # gpu-60`
  - -i option only needed if not using default ssh key
  - Username only needed if different than local username
- If prompted for password, type "Ctrl-C" and contact us for assistance
  - If using a password, be very careful. Three failures will put you IP address in a blacklist.
- To simplify the login to aac6 for instance, you can add the following to your `.ssh/config` file.
- ```
# aac6
Host aac6
User <USERNAME>
Hostname aac6.amd.com
#IdentityFile <path>/<private key file> # customize location of key file
IdentityFile id_ed25519 # file is in .ssh directory
ServerAliveInterval 600
ServerAliveCountMax 30
```
- The `ServerAlive*` lines in the config file may be added to avoid timeouts when idle.
- Now you can login with `ssh aac6` (you can do a similar thing for gpu-60)

# File copying

You should be able to copy files in or out with the `scp` command.

- In: `scp -i <path>/<keyfile> <file> <username>@aac6.amd.com:~/<path>/<file>`
- Out: `scp -i <path>/<keyfile> <username>@aac6.amd.com:~/path/to/your/file ./`

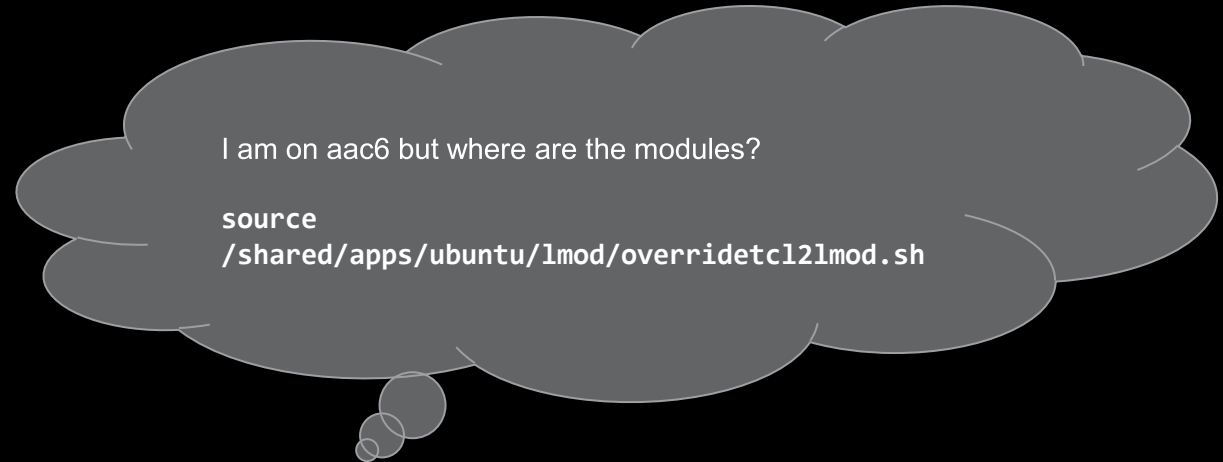
You can also use `rsync` command

- `rsync -avz -e "ssh -i <path>/<keyfile>" <file> <username>@aac6.amd.com:~/path/to/your/files`
- Back-up your files regularly if you think you'll need them after this event

# aac6 environment

- Model Installation Environment: <https://github.com/amd/HPCTrainingDock>
  - Ubuntu® 22.04
  - ROCm version 7.0.1 and 6.4.1
  - GCC is version 11
  - Clang/LLVM is version 19

- Modules using Lua modules
  - Hierarchical module list
  - Initial set on initial login



```
----- /shared/apps/modules/ubuntu/lmodfiles/base -----  
miniconda3/25.3.1  rocm/6.4.1  rocm/7.0.1  rocm-afar/22.20  
gcc/base          miniforge3/24.9.0  turbovnc/3.0.3
```

- rocm/6.4.1 -- ROCm software stack including hip and hip libraries
- Select rocm/6.4.1 for list of software built and dependent on it
- ROCm AFAR drops also available (versions 7.0.5, 22.1.0 and 22.2.0)

# gpu-60 environment

- Model Installation Environment: <https://github.com/amd/HPCTrainingDock>
  - Ubuntu® 22.04
  - ROCm version 6.4.3
  - GCC is version 11
  - Clang/LLVM is version 19

```

----- /etc/lmod/modules/ROCM -----
amdclang/-6.4.3   amdclang/19.0.0-6.4.3 (D)   hipfort/6.4.3   opencl/6.4.3   rocm/6.4.3

----- /etc/lmod/modules/ROCMPlus-AI -----
cupy/13.6.0     hip-python/13.6.0   pytorch/2.8.0_tunableop_enabled   tensorflow/merge-250318
ftorch/dev     jax/0.6.0           pytorch/2.8.0   (D)

----- /etc/lmod/modules/ROCMPlus -----
adios2/2.10.1   hdf5/1.14.6         hypre/2.33.0     netcdf-c/4.9.3     petsc/3.23.1
fftw/3.3.10    hpctoolkit/2024.01.99-next   kokkos/4.7.01   netcdf-fortran/4.6.2   tau/dev

----- /etc/lmod/modules/ROCMPlus-MPI -----
mpi4py/4.1.0   openmpi/5.0.7-ucx1.4.4-ucx1.18.1

----- /etc/lmod/modules/Linux -----
clang/base     gcc/base     gcc/11 (D)   gcc/12

----- /usr/share/lmod/lmod/modulefiles/Core -----
lmod/6.6     settarg/6.6

Where:
D: Default Module

```

# Environment -- modules

## Modules using Lua modules

- ROCm

    amdclang/19.0.0-6.4.1                      rocprofiler-compute/6.4.1 (D)

    rocprofiler-sdk/6.4.1

    hipfort/6.4.1                              rocprofiler-systems/6.4.1 (D)

    opencl/6.4.1

- amdclang/19.0.0-6.4.1 -- OpenMP® compiler
- hipfort/6.4.1-- Fortran wrappers for hip calls
- opencl/6.4.1 – OpenCL™ language
- rocprofiler-compute/6.4.1 Kernel profiler, built-in version
- rocprofiler-systems/6.4.1 Timeline profiler, built-in version
- rocprofiler-sdk/6.4.1 -- Trace decoder and profiler software development toolkit

# Environment – ROCmPlus modules

- ROCmPlus-6.4.1

adios/2.10.1

cupy/13.6.0

fftw/3.3.10

ftorch/dev

hdf5/1.14.6

hip-python/13.6.0

hipfort\_from\_source/6.4.1

hipifly/dev

hpctoolkit/2024.01.99-next

hypre/2.33.0

jax/0.6.0

kokkos/4.6.01

mpi4py/4.0.3

netcdf-c/4.9.3

netcdf-fortran/4.6.2

openmpi/5.0.7-ucc1.4.4-ucx1.18.1

petsc/3.23.1

pytorch/2.7.1

rocprofiler-compute/develop

rocprofiler-systems/develop

scorep/9.0

tau/dev

tensorflow/merge-250318

# Environment -- modules

## Modules using Lua modules

- ROCm
- [amdflang-new](#)

AMD is working on a Next Generation Fortran compiler. An earlier version than used in this training is now part of ROCm 7. During the training, **Participants will have hands-on access to the beta**. The beta itself contains early access to open-sourced pre-release features. Going forward, AMD is able to provide continued early access via AFARs (Advanced Feature Access Releases) as improvements and features are added to the beta. As with any beta, we looking to gather feedback on functionality, usability and user experience.

The latest AFARs containing the next generation Fortran compiler also contains later versions of the C and C++ compiler, much of a later ROCm version and profilers.

**PRE-PRODUCTION SOFTWARE:** The software accessible on this page may be a pre-production version, intended to provide advance access to features that may or may not eventually be included into production version of the software. Accordingly, pre-production software may not be fully functional, may contain errors, and may have reduced or different security, privacy, accessibility, availability, and reliability standards relative to production versions of the software. Use of pre-production software may result in unexpected results, loss of data, project delays or other unpredictable damage or loss. Pre-production software is not intended for use in production, and your use of pre-production software is at your own risk.

# Environment -- modules

## Modules using Lua modules

Compiler modules set the CC, CXX, FC flags. Only one compiler module can be loaded at a time. hipcc is in the path when the rocm module is loaded. These modules also set the OMPI\_CC/CXX/FC environment variables so that the compiler wrappers use the currently set compiler.

## Module commands

- module avail
- module load <package>
- module unload <package>

# Environment -- Slurm

Slurm configuration with single queue LocalQ

sinfo

| PARTITION                    | AVAIL | TIMELIMIT  | NODES | STATE | NODELIST                                   |
|------------------------------|-------|------------|-------|-------|--------------------------------------------|
| 1CN192C4G1H_MI300A_Ubuntu22* | up    | 8-00:00:00 | 5     | idle  | ppac-p11-s24-[16,26,30,35],ppac-p11-s25-40 |
| 1CN48C1G1H_MI300A_Ubuntu22   | up    | 8-00:00:00 | 4     | idle  | sh5-p11-s12-[09,12,15,33,36]               |

25 total GPUs

1CN192C4G1H\_MI300A\_Ubuntu22 - Node with four MI300As

1CN48C1G1H\_MI300A\_Ubuntu22 - Node with one MI300A

Slurm commands

- salloc can be used to schedule a long-term interactive session
  - sbatch is used to submit a job to the batch queue
  - squeue will show the jobs running in the batch queue
- 
- Please allocate only the GPUs that you need and release them after you are done

# aac6 – Total 25 GPUs – your local system may be different

AAC6 is all pre-production parts

Login or Control Node

aac6:  
pl1vm1mi300ctl01

Slurm

salloc  
srun  
sbatch

Queue 1CN192C4G1H\_MI300A\_Ubuntu22

ppac-pl1-s24-16

ppac-pl1-s24-26

ppac-pl1-s24-30

ppac-pl1-s24-35

4 MI300A  
192 CPUs  
512 GB memory

ppac-pl1-s24-16

Queue 1CN48C1G1H\_MI300A\_Ubuntu22

sh5-pl1-s12-09

sh5-pl1-s12-12

sh5-pl1-s12-15

sh5-pl1-s12-33

1 MI300A  
48 CPUs  
128 GB memory

sh5-pl1-s12-36

# Sample slurm commands

- sinfo
- squeue -u <username>
- scancel <jobid>

Most work should be done on the compute nodes

- srun -p 1CN48C1G1H\_MI300A\_Ubuntu22 --exclusive my\_batch\_script.batch
- srun -p 1CN48C1G1H\_MI300A\_Ubuntu22 --ntasks=16 my\_batch\_script.batch
- srun -p 1CN48C1G1H\_MI300A\_Ubuntu22 --ngpus=1 --ntasks=4 my\_batch\_script.batch
- salloc --ntasks 16 --gpus=1 -t
- salloc --nodelist=ppac-pl1-s24-30 --ntasks 1 --gpus=1 --time=01:00:00 --ime=01:00:00
- salloc -N 1 --gpus=4 --cpus-per-gpu=12 --time=02:00:00
- sbatch my\_batch\_script.batch

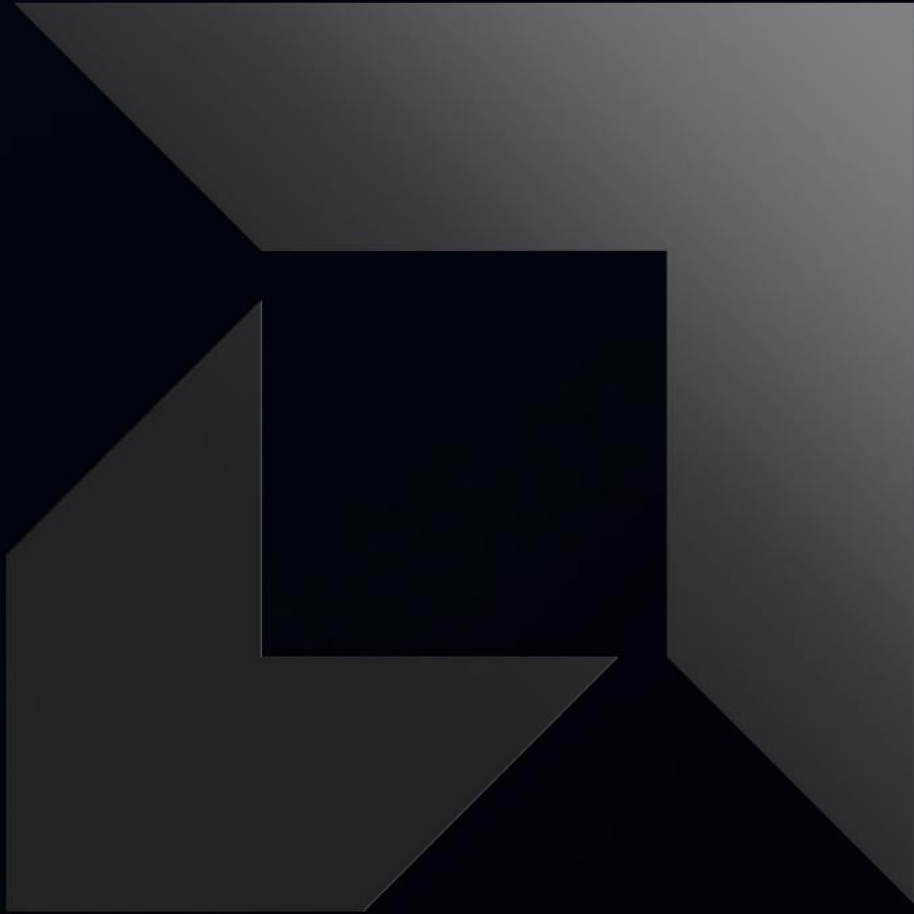
# Sample slurm batch file

- At the top of the batch file
- Bash script commands follow the #SBATCH section

```
#!/bin/bash
#SBATCH --partition=1CN192C4G1H_MI300A_Ubuntu22
#SBATCH --gpus=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --time=01:00:00
#SBATCH -- output=my_job.out
#SBATCH -- error=my_job.out
```

# Installed software

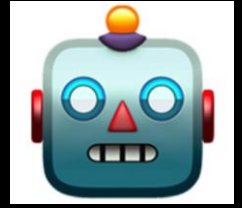
- emacs
- vim
- autotools
- cmake
- tmux
- boost
- eigen
- fftw
- gmp
- gsl
- hdf5-openmpi
- lapack
- magma
- matplotlib
- parmetis
- mpfr
- mpi4py
- openblas
- openssl
- swig
- numpy
- scipy
- h5sparse



# Using the AMD AI Training Assistant

Presenters: Bob Robey  
AMD @ CASTIEL HPC

# Interact with the AMD AI Assistant: USERS BEWARE



Before we even start keep this in mind:

You will be using a large language model to fetch information.

## Pros:

1. Easier than a Google search or deep dive into documentation
2. Has the latest information on AMD software as knowledge base
3. Can iterate to get further details through a conversation
4. Reports the sources it used to provide its answers so you can cross examine

## Cons:

1. Can (and most likely will) provide incorrect information at times
2. The interaction is probabilistic: you will not get the same answer twice with the same prompt (most likely)
3. The model is not fine-tuned to be perfect: some answers will be better than others
4. You cannot consider the answers you receive as gold and will still have to validate them independently

# Interact with the AMD AI Assistant: SSH tunnel

We are prototyping the use of an **AMD AI assistant** that has knowledge of the latest documentation and material on AMD GPU software and programming

You will be able to interact with this AI assistant during this training, to quickly fetch information that you will likely need as you navigate your way through the material

First thing to do: create an SSH tunnel to forward a local port (3000) to a remote server's port (3001) via an intermediate server (gpu-60)

From the terminal window you are using to log in to gpu-60 do:

```
ssh -L 3000:gpu-60:3001 193.143.78.200
```

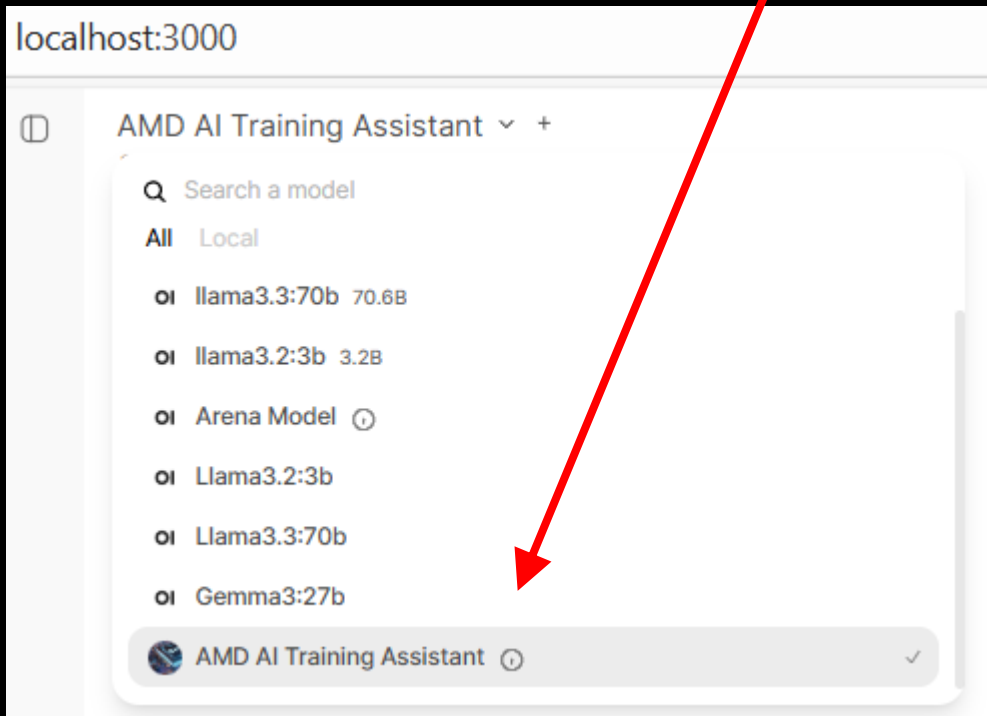
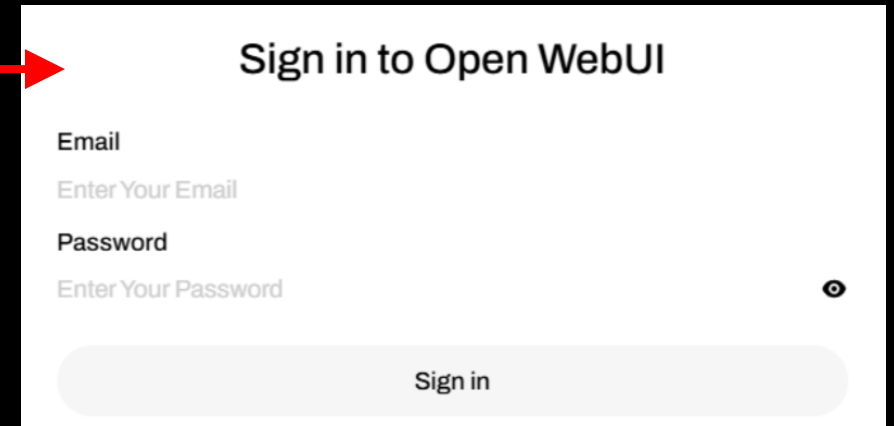
(or alternatively `ssh -L 3000:gpu-60:3001 <username>@193.143.78.200`)

Then open a browser window, and on the address bar type:

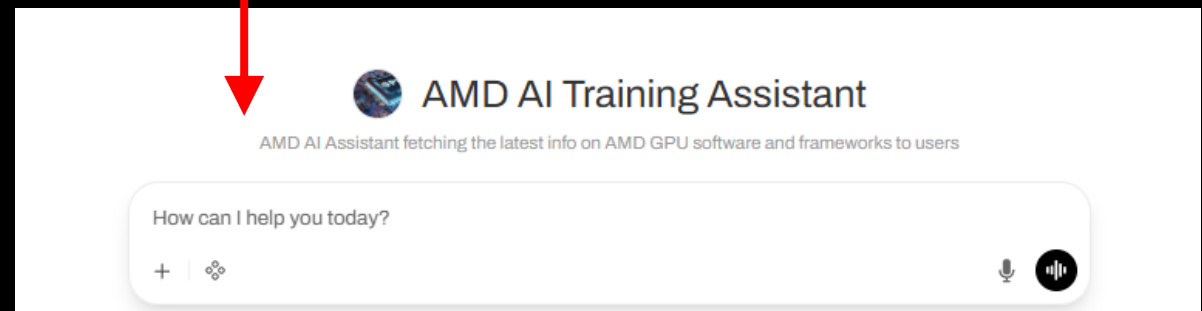
```
http://localhost:3000
```

# Interact with the AMD AI Assistant: logging in

- 1) Type the email and password that have been provided to you
- 2) Select the model from the drop-down list



- 3) Ask a question to the model



# Model based on gemma3:27b



what is this

**OI Gemma3:27b**  
Based on the image, this is a **cat**.  
Specifically, it appears to be a **tabby cat**, as you can see the distinctive striped markings on its fur. It has beautiful green eyes and looks like a sweet and curious feline!

✎ 🗑️ 🔄 🕒 📌 🗣️ 🌐 🔄

**Follow up**

What are some common characteristics of tabby cats?

Can you tell me more about why cats have green eyes?

What breed of cat do you think this is?

How can I tell the difference between different cat breeds?

**OI llama3.3:70b**  
It seems like you're trying to share an image, but as a text-based AI model, I don't have the capability to view or directly access images. If you can describe the image or provide more context about what it depicts or represents, I'd be happy to try and help you understand or discuss it further!

✎ 🗑️ 🔄 🕒 📌 🗣️ 🌐 🔄

**Follow up**

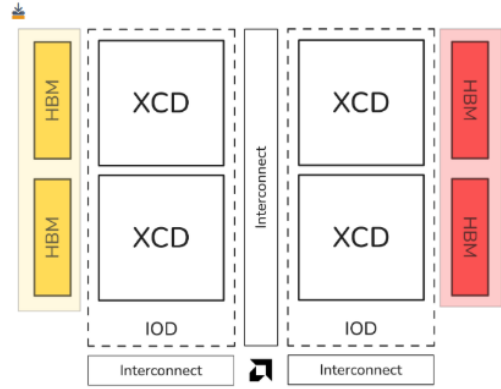
How can I describe an image effectively to get your help?

Are there any specific details I should focus on when...

Can you give me examples of how others have successful...

gemma3 is capable of recognizing images as inputs, llama3.3 is not

# See how RAG knowledge refines the answer



**Gemma3:27b**

Here's a breakdown of the diagram, describing the components and their connections:

**Overall Architecture:**

The diagram illustrates a multi-chip module (MCM) or chiplet-based design, likely targeting high-performance computing (HPC) or AI applications. It's composed of multiple smaller processing units connected together.

**Components:**

- **XCD (Compute Die):** These are the core processing units. Likely standing for Compute Die. They contain the processing logic. There are four of these in the diagram.
- **IOD (I/O Die):** These are Input/Output Dies. They act as interfaces for the Compute Dies, providing connections to external systems, memory, and other devices. There are two IODs here, each connecting to two XCDs.

**AMD AI Training Assistant**

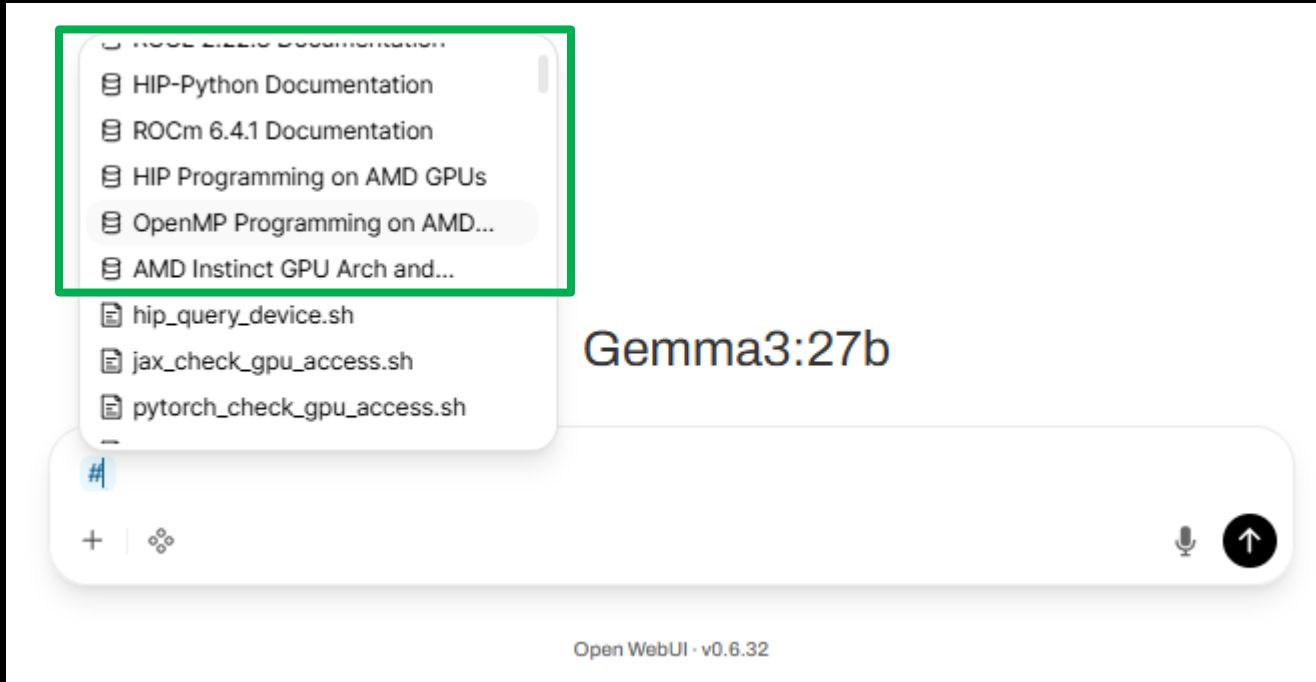
Okay, I've reviewed the provided text corpus. It's a substantial collection of materials detailing AMD's Instinct GPUs, the CDNA architecture, ROCm software stack, and related technologies. Here's a breakdown of the key themes and information, organized for clarity. I'll also highlight what seems to be the *most* important takeaways.

**I. Core Technologies & Architectures**

- **CDNA Architecture:** This is AMD's GPU architecture specifically designed for HPC and AI workloads. Key features include:
  - **Cache Coherency:** A major advantage, enabling simpler application development and potentially better performance, especially when working with both CPU and GPU memory.
  - **High Bandwidth:** Emphasis on fast memory access (HBM) and interconnects (Infinity Fabric).

the knowledge base for the AMD AI Training Assistant made the answer more accurate than the base model

# Use any model and attach a knowledge base with #



You can query a knowledge base or a single document in the knowledge base

Note that these will not have names that are as descriptive so we advise against doing that

# Model output comparison on PyTorch related question

what is the expected value of `torch.cuda.is_available()` if I am running PyTorch on AMD GPUs?

### INCORRECT

**Gemma3:27b**

If you are running PyTorch on AMD GPUs, the expected value of `torch.cuda.is_available()` is **False**.

Here's why:

- `torch.cuda.is_available()` specifically checks for **NVIDIA CUDA compatibility**. The name itself implies this. It's designed to detect the presence of NVIDIA CUDA-enabled GPUs and the necessary drivers.
- AMD GPUs do not use CUDA**. They rely on ROCm (Radeon Open Compute platform) which is a different GPU programming framework.
- PyTorch's support for AMD GPUs:** PyTorch has ROCm support, but `torch.cuda.is_available()` does not check for ROCm. You'll need to use `torch.backends.rocm.is_available()` to check if ROCm is available and PyTorch is configured to use it.

Therefore, running `torch.cuda.is_available()` on a system with only AMD GPUs will consistently return **False**.

How to check for AMD GPU availability in PyTorch:

Use `torch.backends.rocm.is_available()` and `torch.backends.rocm.device_count()`.

### CORRECT

**AMD AI Training Assistant**

Based on the provided test suite and documentation, the expected value of `torch.cuda.is_available()` when running PyTorch on AMD GPUs is **True**.

The test suite explicitly checks for this condition ( `test_pytorch_cuda_available.sh` ) and relies on it being true to confirm a successful ROCm/PyTorch setup. The script aims to verify that PyTorch can see and utilize the AMD GPU.

Several other tests also depend on the assumption that `torch.cuda.is_available()` returns **True** when running on AMD GPUs with ROCm enabled.

61 Sources

Follow up

What specific ROCm versions are supported by this...

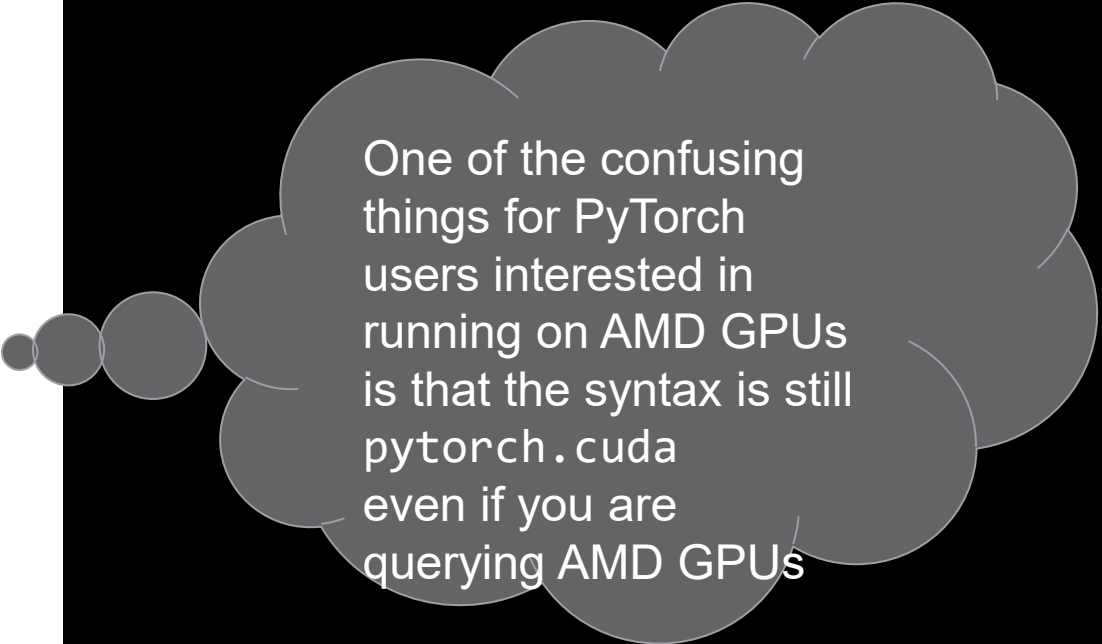
Does `torch.cuda.is_available()` provide any information...

If I'm having issues with PyTorch not recognizing my AMD...

Are there any known compatibility issues between PyTorch...

How does this compare to how `torch.cuda.is_available()`...

Compare the output of the same questions asked to the base model and to the AMD AI Assistant



# Check out sources for more details

By following these steps, you should be able to install PyTorch and run it on your AMD GPU.

- References from 1 pytorch\_check\_gpu\_name.sh pytorch\_check\_gpu\_access.sh and 23 more ^
- 3 README91.md
  - 4 Creating a PyTorch\_TensorFlow code environment on AMD ...
  - 5 README92.md
  - 6 rocm\_setup.sh
  - 7 tensorflow\_check\_gpu\_name.sh
  - 8 pytorch\_mnist\_2gpus.batch
  - 9 README86.md
  - 10 pytorch\_check\_import.sh
  - 11 Programming AMD GPUs with Julia — ROCm Blogs.pdf
  - 12 Accelerating Large Language Models with Flash Attention o...
  - 13 pytorch\_setup.sh
  - 14 pytorch\_mnist\_venv.batch
  - 15 pytorch\_mnist.batch
  - 16 Deep dive into the MI300 compute and memory partition m...
  - 17 Getting to Know Your GPU\_ A Deep Dive into AMD SMI — R...
  - 18 Developing Triton Kernels on AMD GPUs — ROCm Blogs.pdf
  - 19 TorchTune on AMD GPUs How-To Guide\_ Fine-tuning and S...
  - 20 pytorch\_mnist\_venv\_2gpus.batch
  - 21 CMakeLists.txt
  - 22 amd-cdna-3-white-paper.pdf
  - 23 torchvision\_check\_import.sh
  - 24 Day 0 Developer Guide\_ Running the Latest Open Models fr...
  - 25 GEMM Kernel Optimization For AMD GPU — ROCm Blogs.p...

At the end of every response, you can examine the sources that have been used to provide the answer

### Citation

Relevance  
84.88% (0.8488)

Content  
#!/bin/bash

```
# This test checks that the GPU  
# seen by PyTorch is an AMD GPU
```

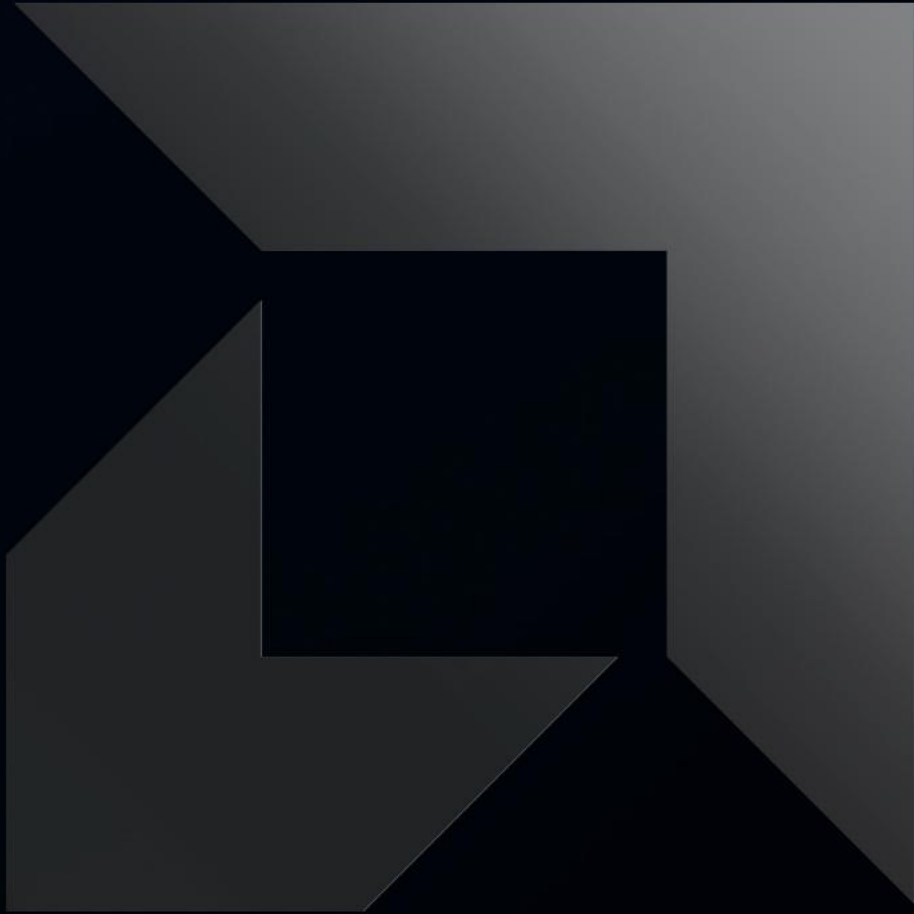
```
# NOTE: this test assumes PyTorch has been installed according  
# to the instructions available in the model installation repo:  
# https://github.com/amd/HPCTrainingDock/blob/main/extras/scripts/pytorch_setup.sh
```

```
module load rocm  
module load rocm pytorch
```

```
python3 -c "import torch; print(f'device name [0]:', torch.cuda.get_device_name(0))"
```

You can click on each source and examine it

Each source has a percentage of relevance that the model has attributed to it



# APU Programming Model

Presenters: Bob Robey  
AMD @ CASTIEL HPC

# Porting code from CPU to GPU

It is **easier than ever** to port to GPUs

- ❖ Accelerated Processing Units (APUs) simplify porting
  - APU architecture and APU Programming model – what are these?
- ❖ Pragma-based language makes porting **quicker and more portable** to run on CPU and GPU
- ❖ **Optimized libraries for the GPU** can help
- ❖ **Interoperability** of different programming models
- ❖ New tools and **improved functionality**

# What AMD offers

- **MI300A is an APU Architecture – in the actual hardware**
  - ❖ Same memory space for CPU and GPU
  - ❖ **APU is an AMD term; others may call it an integrated GPU**
- Commitment to **APU programming model** for GPU products
  - ❖ **APU programming model applies to discrete GPUs as well**
- AMD values customer friendly policies
  - ❖ Open-source
  - ❖ No vendor lock-in
  - ❖ Portability
- Industry leading CPU performance
- Commitment to HPC customers
  - ❖ Leading FP64 (IEEE-754) performance
  - ❖ MI300A and MI300X FP64 performance increased over the MI250X products
  - ❖ No special tricks to get FP64 for general applications



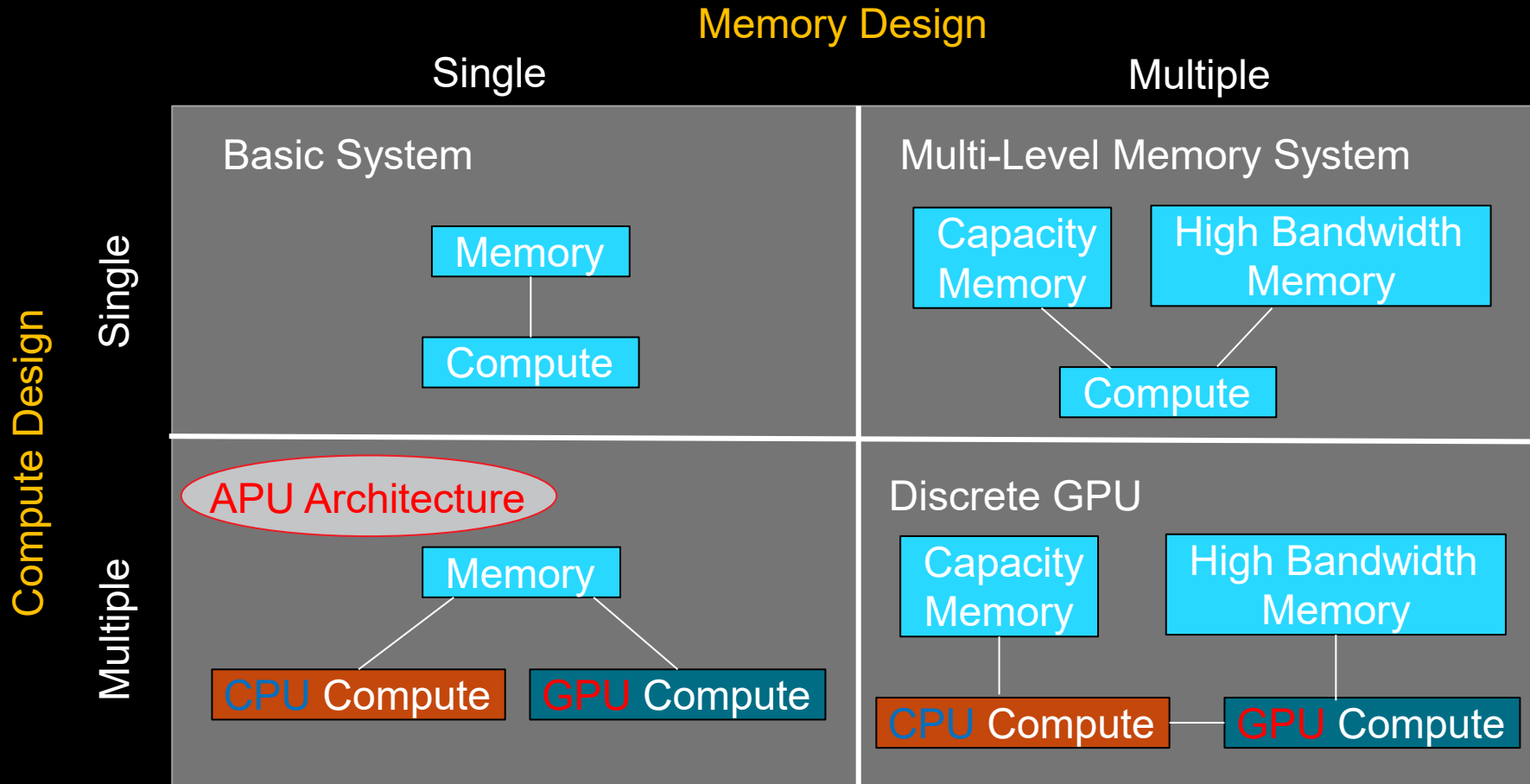
# APU and discrete GPU architectures

Oct 28-30, 2025

AMD @ CASTIEL



# Taxonomy of compute architectures

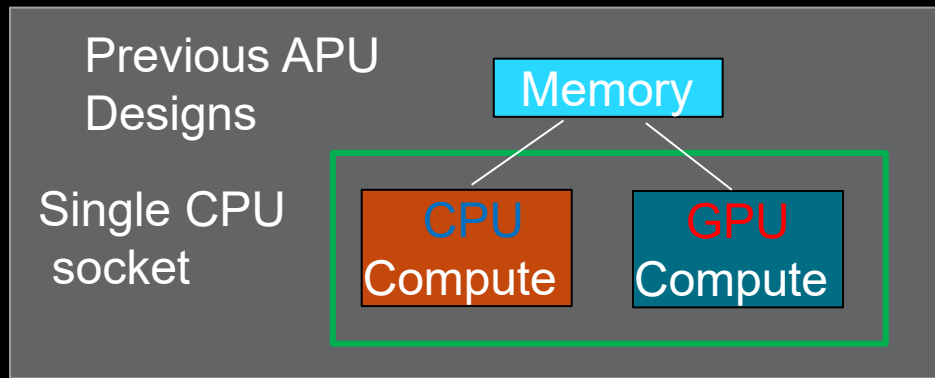


Taxonomy categorizes architecture by dominance of hardware components

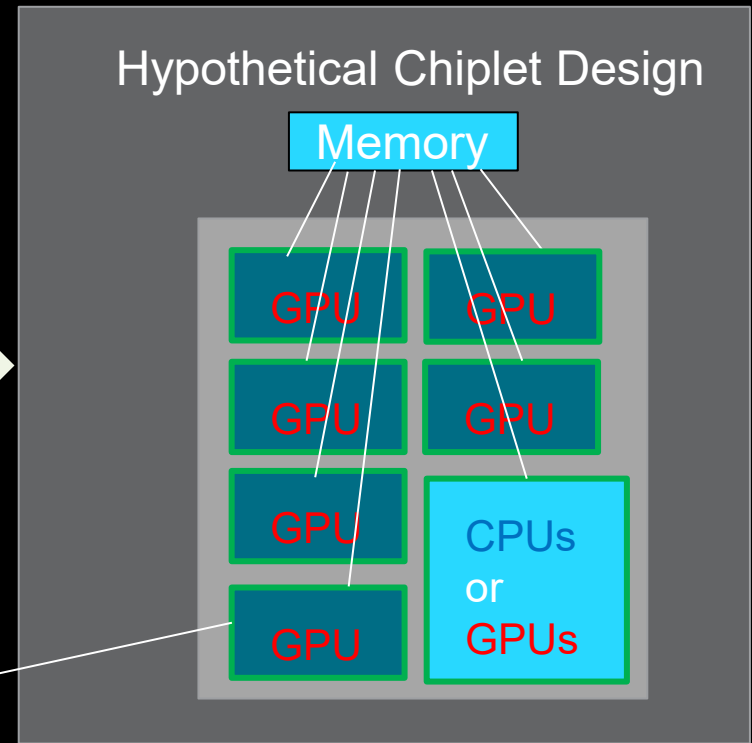
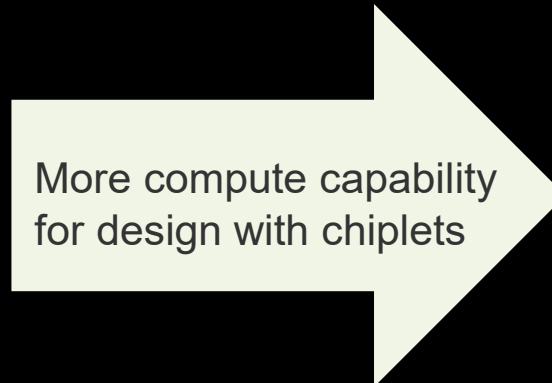
- ❖ Memory Dominant – architecture revolves around a single memory space
- ❖ Compute Dominant – architecture centered around a single compute resource
- ❖ APU is primarily characterized by compute units being able to address all memory

# Breakthrough in compute capability -- Conceptual

- Integrated GPUs have traditionally been limited by how much GPU compute capability can be included
  - ❖ Silicon Chip only has so much space
  - ❖ **Chiplets allow us to expand that space**
- Let's try adding more capability into an APU



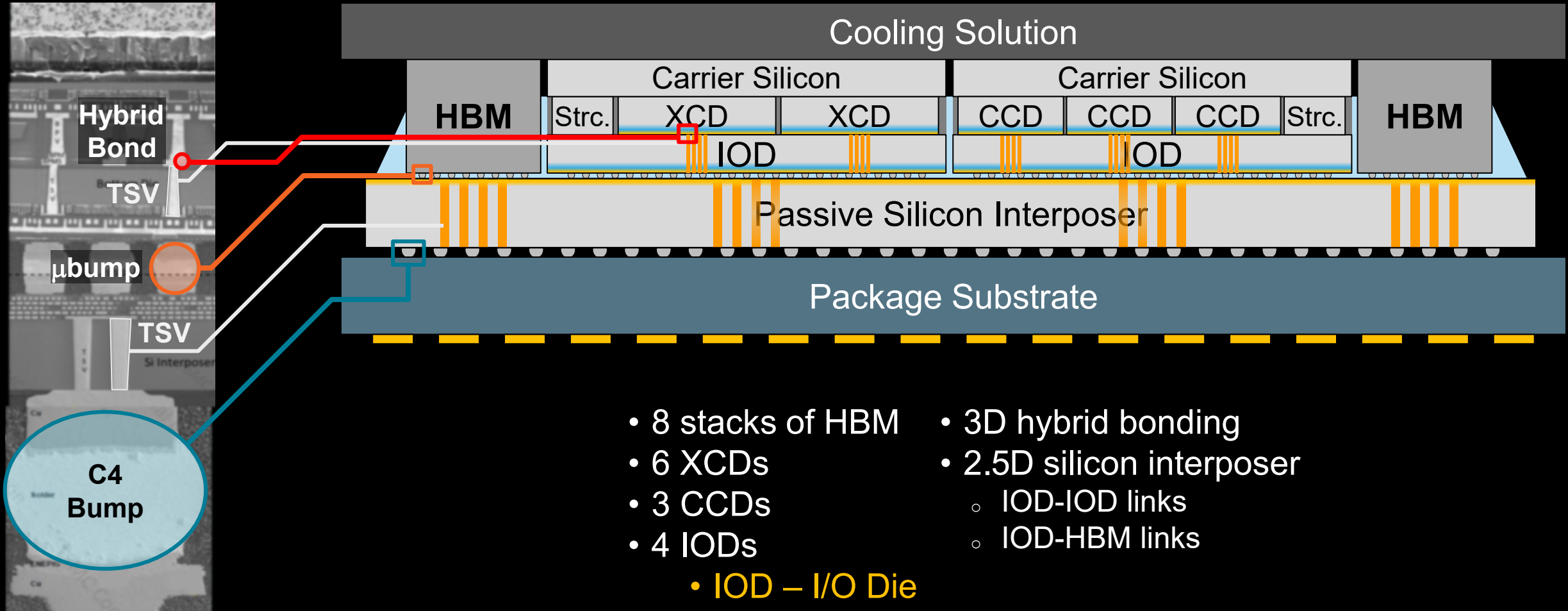
AM5 socket 40x40mm – limited silicon space



Chiplets

Lots more silicon space to work with

# Advanced Heterogeneous Integration Packaging



# Bringing it to AMD Instinct™ Accelerator Products

## MI200 Series

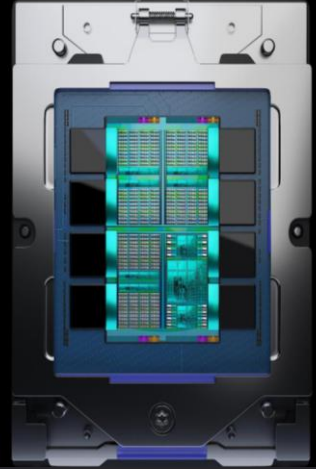
**Extreme Compute Architecture with leading memory capacity and bandwidth**



- Technology in first Exascale systems
- High compute to power ratio
- Tight integration with memory
- Infinity Fabric™ for data transfers

## MI300A

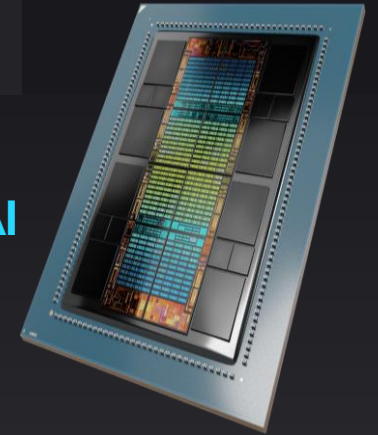
**First True APU Architecture for HPC and AI**



- Memory Bandwidth Workloads
- Hybrid CPU + GPU Capability
- GPUs can drive full bandwidth

## MI300X

**Leadership generative AI accelerator**



- Extreme Compute Workloads
- Suitable for typical AI work
- Other work entirely on GPU



# The APU programming model

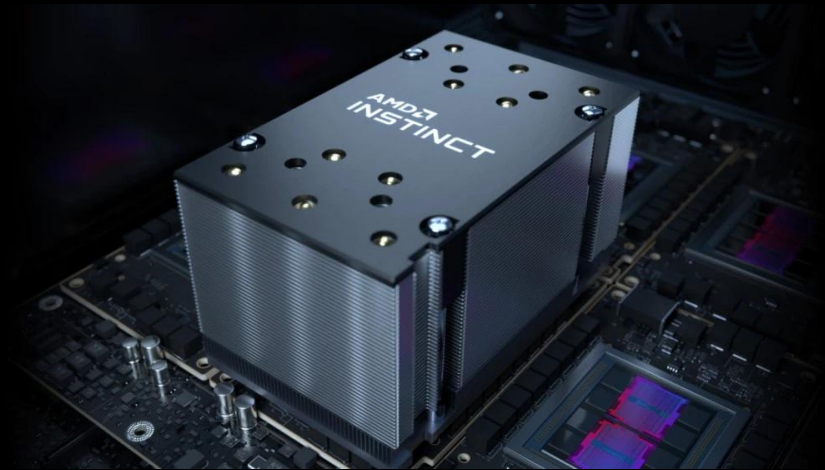
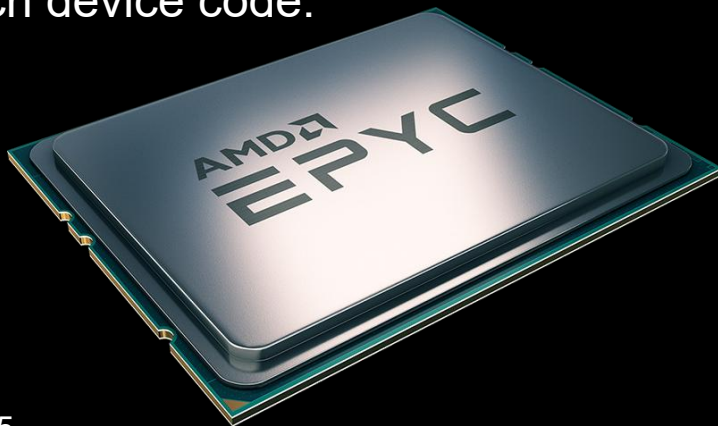
Oct 28-30, 2025

AMD @ CASTIEL

# A tale of Host and Device

Source code for CPU-GPU systems has two flavors: Host code and Device code

- The **Host** is the **CPU**
  - MI300A: 3 “Zen 4” based chiplets
- Host code runs here
- Usual C++ or Fortran syntax and features
- Entry point is the *main* function / PROGRAM
- HIP API or OpenMP® can be used to create device buffers, move between host and device, and launch device code.
- The **Device** is the **GPU**
  - MI300A: 6 XCDs with 38 compute units each (228 CUs total)
- Device code runs here
- C-like syntax (HIP) or OpenMP directives
- Device codes are launched via *kernels*
- Instructions from the Host are enqueued into *streams*



# Memory model

## *Definition*

A memory model defines the rules for the synchronization of memory modifications between threads, compute hardware and cache.

A memory model is critical for parallel computing to help both system developers and application programmers avoid data hazard or race conditions where memory is modified by one entity, but another compute unit fails to get the updated value.

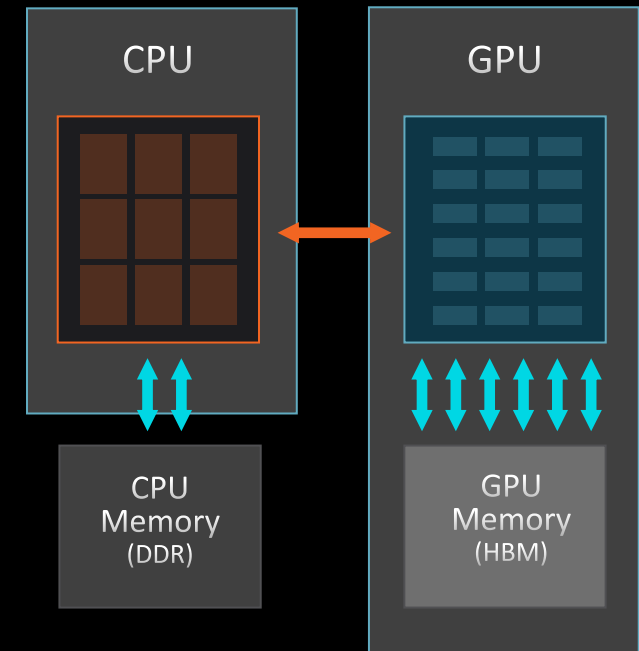
# AMD discrete GPUs and memory addressing

In discrete GPU systems, CPU and GPU memory spaces are **separate** and data needs to be **moved** between the two spaces. This data movement can be performed in two ways:

1. By the **programmer**, explicitly
2. By the **Operating System (OS)**, who helps move pages on access and subsequent page fault
  - We call this managed memory – short for "memory is managed by the operating system"
  - If no corresponding address is found, the program will fail with a segmentation fault

AMD MI200 GPUs (MI210, MI250, MI250X) and MI300X are **discrete GPUs**

- Implement **managed memory**
- To enable managed memory, `export HSA_XNACK=1`



# XNACK -- what is it and how to use it

## Definition

**XNACK** refers to the AMD GPU's ability to retry memory accesses that fail due to a page fault.

## xnack environment variable

On MI200 and MI300 series GPUs, it can be enabled on a per-process basis using the environment variable **HSA\_XNACK=1** and disabled using **HSA\_XNACK=0**. Default decided at boot time.

## xnack compiler flag

Run `rocminfo | grep xnack` to check if xnack is enabled

Compilation mode that can assume three possible values: `xnack+`, `xnack-`, `xnack any`.

To change the xnack compilation mode of a program, `xnack+` or `xnack-` may be appended to the architecture flags:

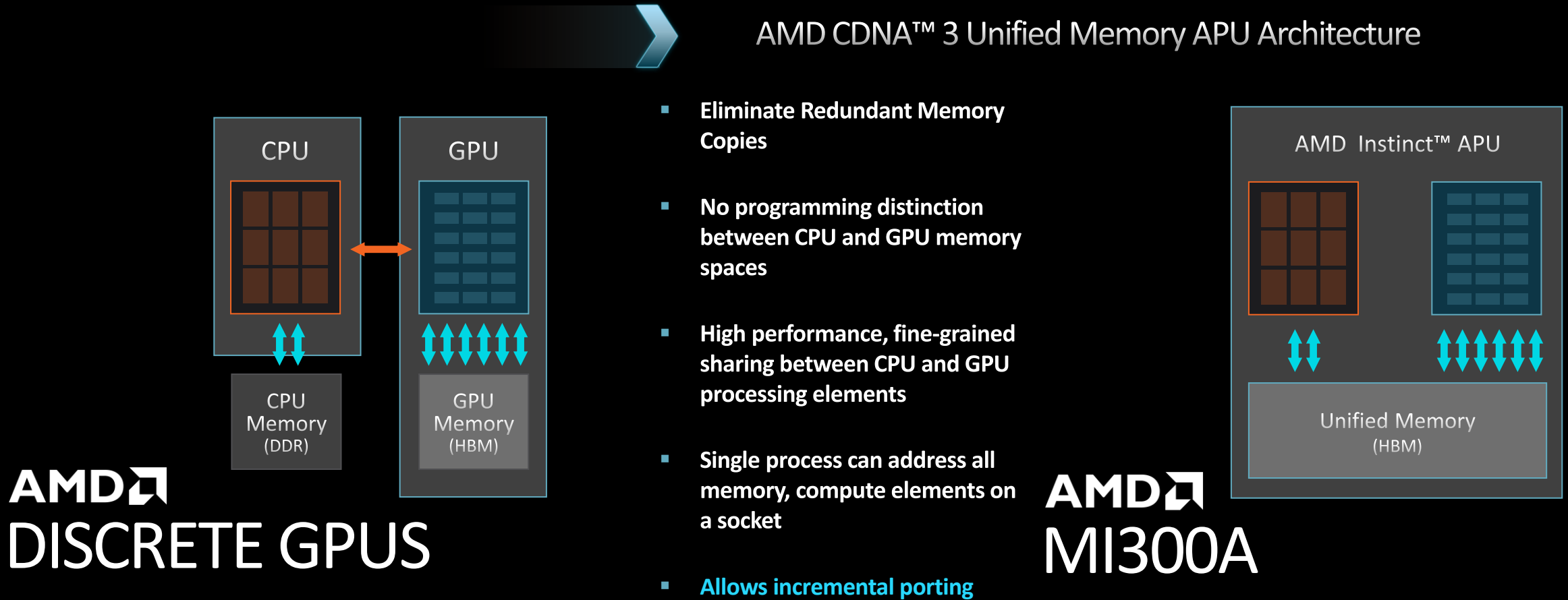
```
--offload-arch=gfx90a:xnack+
```

Supplying multiple xnack options will yield a "fat-binary" with both modes enabled.

When not specified, the default `xnack any` mode will be used.

Code compiled with `xnack any` will run in any case.

# APU ARCHITECTURE BENEFITS FOR CPU TO GPU PORTING



# APU PROGRAMMING MODEL WITH OPENMP®

## CPU CODE

```
!allocation on host
ALLOCATE(var(1:N))

!compute on host
!$omp parallel do &
!$omp private(i), shared(var)
DO i=1,N
    var(i) = ...
END DO
!$omp end parallel do
!sync barrier at omp end ...

...
!deallocation
DEALLOCATE(var)
```

## GPU CODE

```
!allocation on host
ALLOCATE(var(1:N))

!compute on device, expl. mem movement!
!$omp target teams distribute parallel do &
!$omp map(tofrom:var) private(i),shared(var)
DO i=1,N
    var(i) = ...
END DO
!$omp end target teams distribute parallel do
!host-device sync barrier at omp end ...

...
!deallocation
DEALLOCATE(var)
```

Costly to switch  
CPU -> GPU!

## APU CODE

```
!$omp requires unified_shared_memory
!allocation of unified memory
ALLOCATE(var(1:N))

!compute on device, no expl. mem movement!
!$omp target teams distribute parallel do &
!$omp private(i),shared(var)
DO i=1,N
    var(i) = ...
END DO
!$omp end target teams distribute parallel do
!host-device sync barrier at omp end ...

...
!deallocation of unified memory
DEALLOCATE(var)
```

Cheap CPU ->  
GPU with APU

- Compute kernel
- Special directive to enable unified memory
- Explicit memory management between CPU & GPU -> not needed for APU!
- Synchronization Barrier

# APU PROGRAMMING MODEL WITH HIP

## 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

```
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, hipMemcpyHostToDevice);
gpu_func<<< >>>(in_d, out_d, M);
Synchronization implied, hipMemcpy blocks
hipMemcpy(out_h, out_d, Msize, hipMemcpyDeviceToHost);

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

## APU CODE

```
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];
```

- Compute kernel
- GPU memory allocation on Device -> no copies for host and device on APU!
- Explicit memory management between CPU & GPU -> not needed for APU!
- Synchronization Barrier

# PROGRAMMING ACROSS FRAMEWORKS/COMPILERS

## OpenMP® CODE

```
#pragma omp requires unified_shared_memory
```

```
double* in_h, *out_h;
in_h = new (std::align_val_t(128)) double[N];
out_h = new (std::align_val_t(128)) double[N];
```

```
for (int i=0; i<N; i++) // initialize
    in_h[i] = ...;
```

```
#pragma omp target
{ ... }
```

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

```
delete[] in_h; delete[] out_h;
```

## RAJA CODE

```
double* in_h, *out_h;
in_h = new (std::align_val_t(128)) double[N];
out_h = new (std::align_val_t(128)) double[N];
```

```
for (int i=0; i<N; i++) // initialize
    in_h[i] = ...;
```

```
RAJA::forall< exec_policy >(arange, [=]
(int i) { ... } );
```

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

```
delete[] in_h; delete[] out_h;
```

## KOKKOS CODE

```
double* in_h, *out_h;
in_h = new (std::align_val_t(128)) double[N];
out_h = new (std::align_val_t(128)) double[N];
```

```
for (int i=0; i<N; i++) // initialize
    in_h[i] = ...;
```

```
Kokkos::parallel_for(N, [=] (const int
i){ ... });
Kokkos::fence();
```

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

```
delete[] in_h; delete[] out_h;
```

- Compute kernel
- ~~Device specific memory allocator~~
- ~~Explicit memory management between CPU & GPU~~
- Synchronization Barrier



# Choices in programming languages

Oct 28-30, 2025

AMD @ CASTIEL



# Multiple language paths for AMD GPUs -- all offer portability



Native or Low-level languages

ROCm™, HIP, OpenCL



Pragma-based languages

OpenMP, OpenACC (HPE)



Higher Level Performance Portability languages – Frameworks

Kokkos, RAJA



Standard Based Languages

C++ (HIP) Standard Parallelism



# Pragma-based languages

## OpenMP® – primary supported option for AMD GPUs

- Implemented through LLVM™
- Implemented in the cray compilers
- **Recommendation:** Best support for portability, easy to get started with

## OpenACC – Supported through Cray, LLVM™ (CLACC) and GCC compilers

- Not as well supported as other options
- **Recommendation:** Try if your code already has OpenACC implemented
- Also available: source-to-source translation tools from OpenACC to OpenMP (Intel® and CLACC)

## do concurrent (Fortran only)

- requires rewriting the “do”
- Not supported by all compilers yet, but expected very soon
- **Recommendation:** Good option to offload very simple loops, can be considered



# Native or low-level languages

## Heterogeneous Interface for Portability (HIP)

- A portable layer on top of ROCm and CUDA

Requires a different source on CPU and GPU

- Larger effort for porting and overhead
- No equivalent of CUDA Fortran available: Fortran requires C interfaces, thus even larger overhead and two programming languages for same app
- **Recommendation:** HIP for *hottest loops* and complex kernels, if not already CUDA ported
- If already CUDA ported: Converting CUDA to HIP is straightforward
  - Hipify scripts do majority of the work
  - Still requires optimization effort to get best performance
    - e.g. a **wavefront has 64 threads executing** the same instruction (different compared to 32 threads per warp on NVIDIA hardware)
- There are other low-level languages such as OpenCL™

Portable HIP C++ (Host & Device Code)

#include  
"hip\_runtime.h"

hipcc

AMD GPU

CUDA equivalents available make  
this portable to NVIDIA hardware



# Higher level performance portability frameworks

## Kokkos – Sandia National Lab (SNL) C++ performance portable programming model

- ❖ The Kokkos team has aggressively developed support for AMD GPUs via a HIP backend
- ❖ Kokkos handles many of the unique attributes of the AMD GPUs for you
- ❖ Parts being integrated into the C++ standard

## RAJA – Lawrence Livermore National Lab (LLNL) C++ performance portability layer

- ❖ Modular in structure with separation of compute and data management
- ❖ Supports AMD GPUs
- ❖ Key kernel patterns have been optimized by AMD

## Advantages of Performance Portability Frameworks

- ❖ True single-source application code (at least if you restrict yourself to C++)
- ❖ Many of these framework support both CPUs and GPUs



# C++ standard based languages

With the C++ 17 standard, support for parallelism was introduced. The application developer specifies parallelism as the first parameter to a C++ algorithm

- `std::execution::seq` – Sequential execution  
All operations on the thread that invoked the algorithm
  - `std::execution::unseq` – Vectorized execution (C++20)  
Indicate that a parallel algorithm's execution may be vectorized, e.g., executed on a single thread using instructions that operate on multiple data items
  - `std::execution::par` – Parallel multithreaded execution  
Parallel execution allowed. Operations are indeterminately sequenced within a thread
  - `std::execution::par_unseq` – Parallel multithreaded and vectorized execution  
The various operations can be interleaved with each other on the same thread. Any given operation may start on a thread and end on a different thread
- With the release of ROCm 6.1, C++ standard parallelism is available for AMD GPUs.
- ❖ To enable, use the `--hipstdpar` compile flag
  - ❖ The ROCm 6.1 release only supports the `par_unseq` execution policy
  - ❖ **Recommendation:** Consider this option only for experimental use and with experienced C++ developers, not best performance at the moment

# Other languages

Many other languages also work on AMD GPUs to some level and are continually improving

- ❖ SYCL
- ❖ Julia: [instructions](#) on how to add Julia as a module.
- ❖ Python™

ML/AI -- Support for these languages is excellent and portable

- ❖ PyTorch – AMD is a founding member of the PyTorch Foundation
- ❖ JAX
- ❖ Cupy
- ❖ TensorFlow
- ❖ And many other ML/AI packages
  - See <https://github.com/ROCm>

# Hands-on exercises

Located in our HPC Training Examples repo:

<https://github.com/amd/HPCTrainingExamples>

A table of contents for the READMEs if available at the top-level README in the repo

Relevant exercises for this presentation located in ManagedMemory directory.

Link to instructions on how to run the tests: ManagedMemory/README.md

Log into the AAC node and clone the repo:

```
ssh <username>@aac6.amd.com -p <port number>  
git clone https://github.com/amd/HPCTrainingExamples.git
```

# Disclaimer

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

THIS INFORMATION IS PROVIDED ‘AS IS.’ AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Third-party content is licensed to you directly by the third party that owns the content and is not licensed to you by AMD. ALL LINKED THIRD-PARTY CONTENT IS PROVIDED “AS IS” WITHOUT A WARRANTY OF ANY KIND. USE OF SUCH THIRD-PARTY CONTENT IS DONE AT YOUR SOLE DISCRETION AND UNDER NO CIRCUMSTANCES WILL AMD BE LIABLE TO YOU FOR ANY THIRD-PARTY CONTENT. YOU ASSUME ALL RISK AND ARE SOLELY RESPONSIBLE FOR ANY DAMAGES THAT MAY ARISE FROM YOUR USE OF THIRD-PARTY CONTENT.

© 2025 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo, AMD CDNA, AMD ROCm, AMD Instinct, and combinations thereof are trademarks of Advanced Micro Devices, Inc. in the United States and/or other jurisdictions. Other names are for informational purposes only and may be trademarks of their respective owners.

LLVM is a trademark of LLVM Foundation

Git and the Git logo are either registered trademarks or trademarks of Software Freedom Conservancy, Inc., corporate home of the Git Project, in the United States and/or other countries

OpenCL is a trademark of Apple Inc. used by permission by Khronos Group, Inc.

Intel® is a trademark of Intel Corporation or its subsidiaries

The OpenMP name and the OpenMP logo are registered trademarks of the OpenMP Architecture Review Board

**AMD** 