# AMD Instinct™ GPUs: Hardware and Software

Dr.-Ing. Michael Klemm
Principal Member of Technical Staff
HPC Center of Excellence

Derek Bouius
Sr. Product Manager
GPU Compute Software

Justin Chang
Member of Technical Staff
A + A Solutions Group

# Cautionary Statement

This presentation contains forward-looking statements concerning Advanced Micro Devices, Inc. (AMD) such as the features, functionality, performance, availability, timing and expected benefits of AMD products and product roadmaps, which are made pursuant to the Safe Harbor provisions of the Private Securities Litigation Reform Act of 1995. Forward-looking statements are commonly identified by words such as "would," "may," "expects," "believes," "plans," "intends," "projects" and other terms with similar meaning. Investors are cautioned that the forward-looking statements in this presentation are based on current beliefs, assumptions and expectations, speak only as of the date of this presentation and involve risks and uncertainties that could cause actual results to differ materially from current expectations. Such statements are subject to certain known and unknown risks and uncertainties, many of which are difficult to predict and generally beyond AMD's control, that could cause actual results and other future events to differ materially from those expressed in, or implied or projected by, the forward-looking information and statements. Investors are urged to review in detail the risks and uncertainties in AMD’s Securities and Exchange Commission filings, including but not limited to AMD’s most recent reports on Forms 10-K and 10-Q.

AMD does not assume, and hereby disclaims, any obligation to update forward-looking statements made in this presentation, except as may be required by law.

# Agenda

- AMD Instinct™ Architecture
- AMD ROCm™ Software Stack
- OpenMP Offload Programming
- HIP and HIPifying Code
- Q&A

# AMD Instinct™ Architecture

# Industry’s First GPU to Break 10 TF (FP64) Barrier

2000

ASCI White, #1 Supercomputer

6 Megawatts, 212K Pounds (106 Tons), 12.3 Teraflops Peak

2020

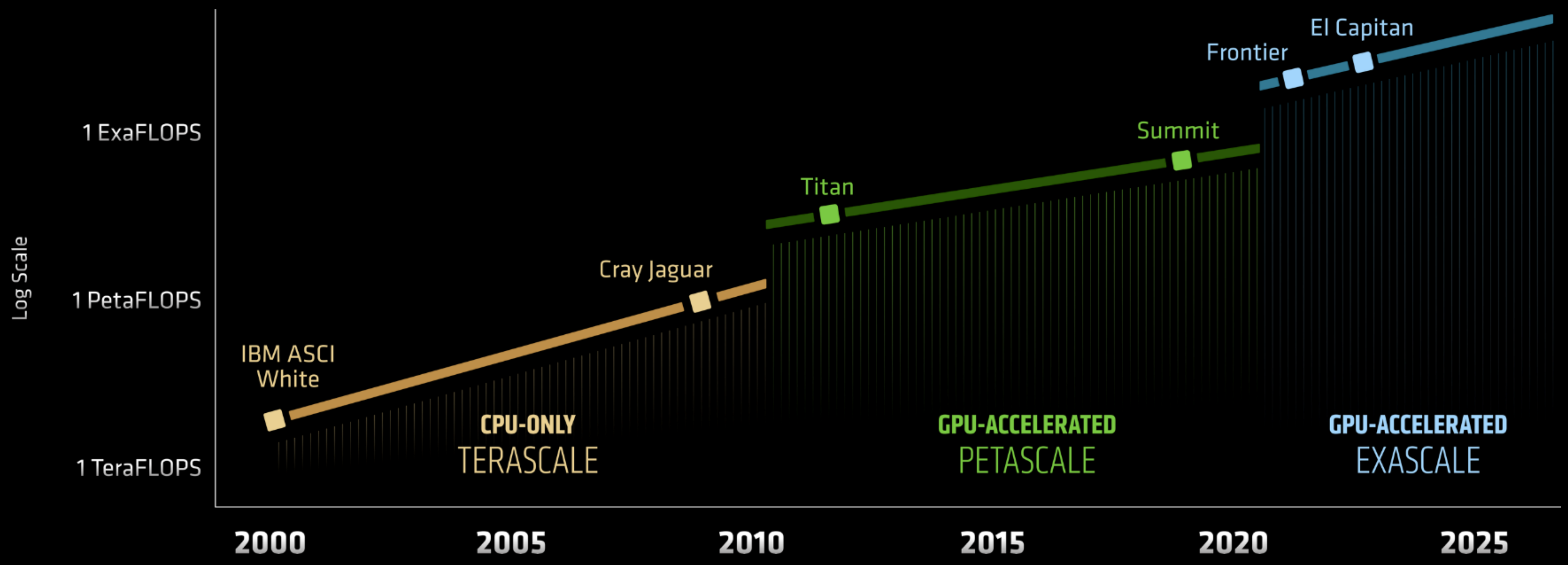
AMD Instinct™ MI100 GPU

300 Watts, 2.56 Pounds (1.16Kg), 11.5 Teraflops Peak

SEE ENDNOTES MI100-05

# The Dawn of GPU-Accelerated Exascale: Major Leaps in Performance Driving Three Phases of Supercomputing

Performance of Top Supercomputers



SOURCE: HTTPS://OPENAI.COM/BLOG/AI-AND-COMPUTE/ (MACHINE INTELLIGENCE) AND HTTPS://WWW.TOP500.ORG/ (HIGH PERFORMANCE COMPUTING)

# Application Optimized Architectures

## HIGHEST EFFICIENCY THROUGH DOMAIN SPECIFIC OPTIMIZATION

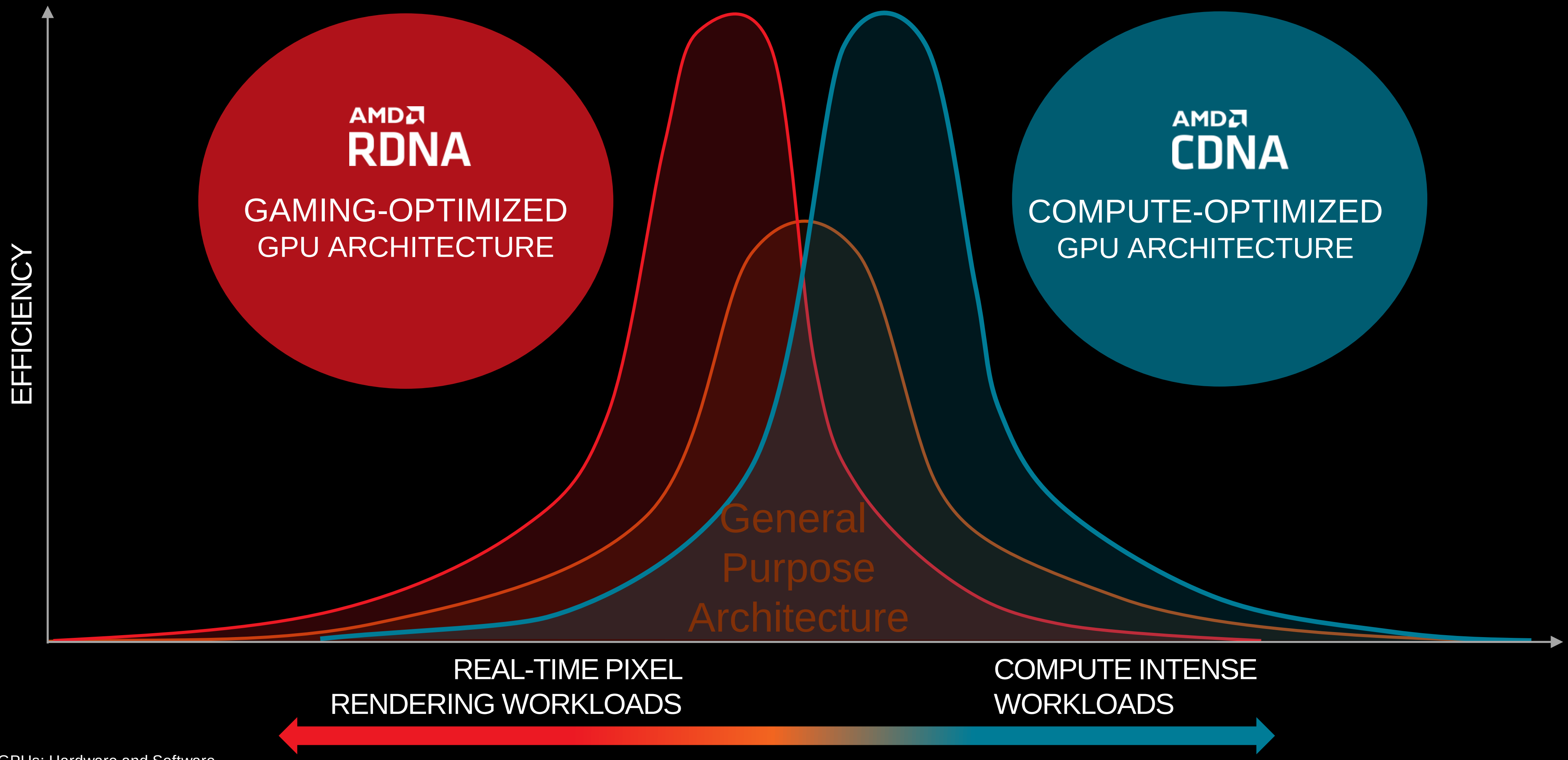
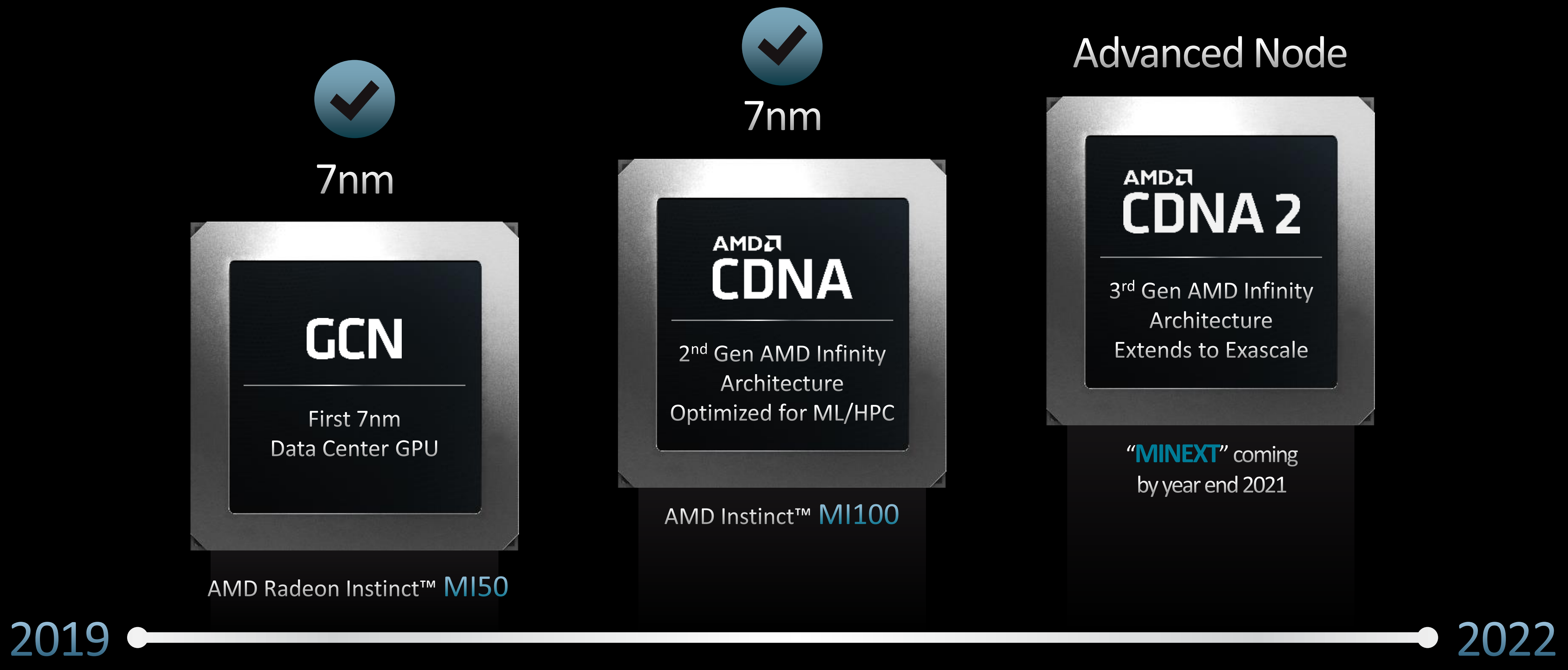


CHART FOR ILLUSTRATIVE PURPOSES

# Data Center GPU Architecture Roadmap



✓ 7nm

GCN

First 7nm
Data Center GPU

AMD Radeon Instinct™ MI50

✓ 7nm

AMD CDNA

2nd Gen AMD Infinity
Architecture
Optimized for ML/HPC

AMD Instinct™ MI100

Advanced Node

AMD CDNA 2

3rd Gen AMD Infinity
Architecture
Extends to Exascale

“MINEXT” coming
by year end 2021

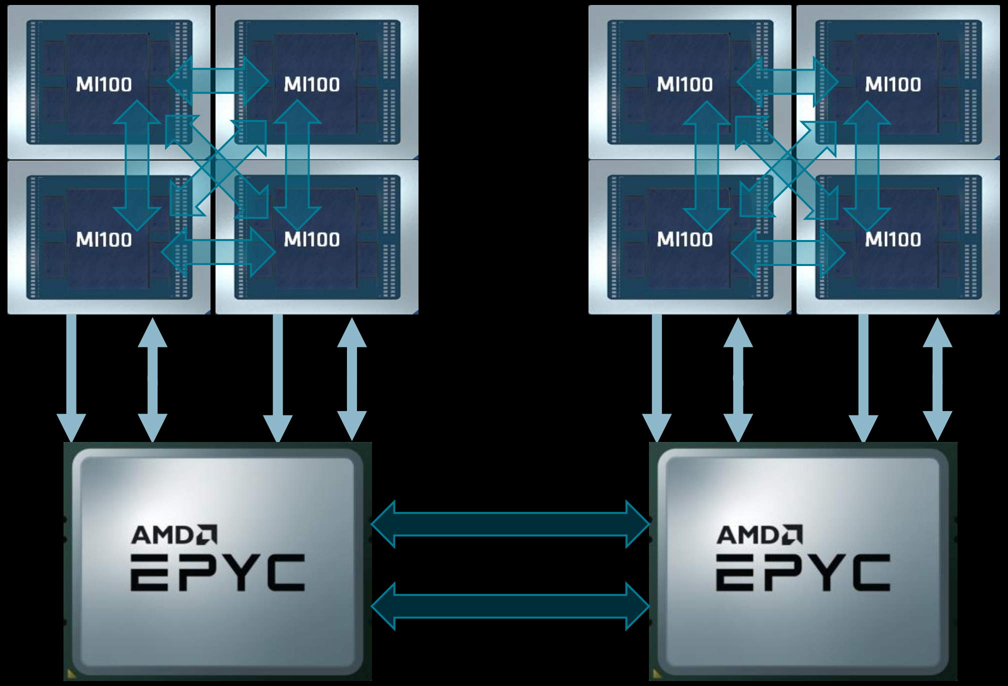
2019

2022

Roadmaps Subject to Change

# Node-level Design – GPU Hives

- GPUs can form “hives” of four GPUs
- One hives associated to one processor
- High-speed AMD Infinity Fabric™ connections in the hives (fully connected).

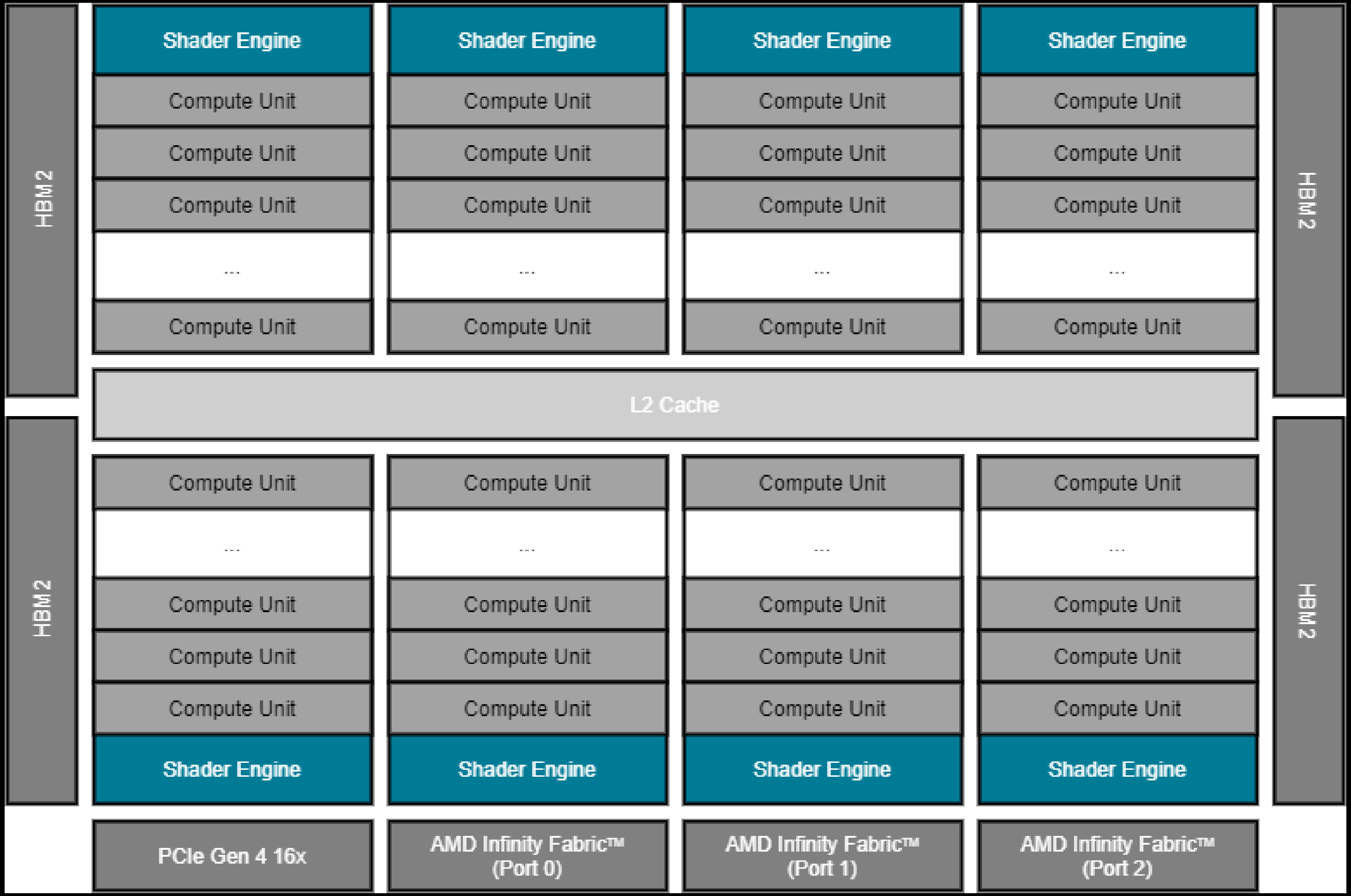


⟷ PCIe® Gen 4 link

⟷ AMD Infinity Fabric™ technology

# AMD CDNA™ Architecture – Made for Performance

- GPU is composed from several main blocks using an on-die fabric
- 120 Compute Units (CU)
  - Four Compute Engines w/ SIMD
  - SIMD pipelines execute 16-wide instructions
- Support for int8, FP16, FP32, FP64, bfloat16



HBM2

Shader Engine | Shader Engine | Shader Engine | Shader Engine

Compute Unit | Compute Unit | Compute Unit | Compute Unit

Compute Unit | Compute Unit | Compute Unit | Compute Unit

Compute Unit | Compute Unit | Compute Unit | Compute Unit

... | ... | ... | ...

Compute Unit | Compute Unit | Compute Unit | Compute Unit

HBM2

L2 Cache

HBM2

Compute Unit | Compute Unit | Compute Unit | Compute Unit

... | ... | ... | ...

Compute Unit | Compute Unit | Compute Unit | Compute Unit

Compute Unit | Compute Unit | Compute Unit | Compute Unit

Compute Unit | Compute Unit | Compute Unit | Compute Unit

Shader Engine | Shader Engine | Shader Engine | Shader Engine

HBM2

PCIe Gen 4 16x | AMD Infinity Fabric™ (Port 0) | AMD Infinity Fabric™ (Port 1) | AMD Infinity Fabric™ (Port 2)

# AMD ROCm™ Software Stack

# Evolution of the AMD ROCm™ Software Stack

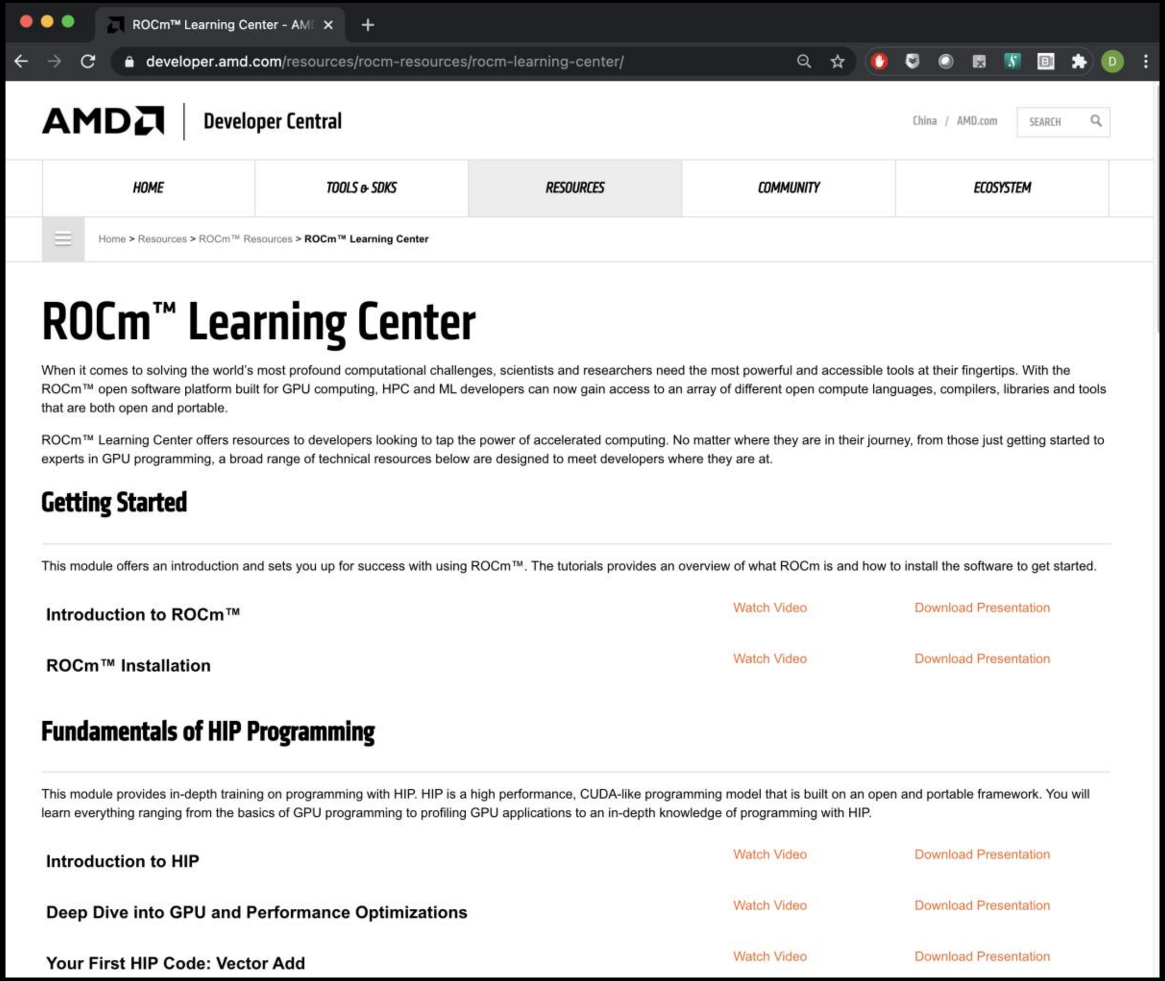
| | | | | |
|---|---|---|---|---|
| Applications | HPC Apps | | ML Frameworks | |
| Cluster Deployment | Singularity | SLURM | Docker | Kubernetes |
| Tools | Debugger | Profiler, Tracer | System Valid. | System Mgmt. |
| Portability Frameworks | Kokkos | RAJA | GridTools | ONNX |
| Math Libraries | RNG, FFT | Sparse | BLAS, Eigen | MIOpen |
| Scale-Out Comm. Libraries | OpenMPI | UCX | MPICH | RCCL |
| Programming Models | OpenMP | HIP | OpenCL™ | Python |
| Processors | CPU + GPU | | | |

ECLIPSE FOUNDATION · Red Hat · slurm workload manager · docker · TensorFlow · PYTORCH · MLIR · LLVM COMPILER INFRASTRUCTURE · tvm · CentOS · SUSE · ECP EXASCALE COMPUTING PROJECT · mindtech · ubuntu · OpenMP

Future | Beta/Early | Production

AMD

# Hands on Training Material at ROCm$^{TM}$ Learning Center



ROCm™ Learning Center - AMD
developer.amd.com/resources/rocm-resources/rocm-learning-center/
AMD | Developer Central
China / AMD.com
SEARCH
HOME
TOOLS & SDKS
RESOURCES
COMMUNITY
ECOSYSTEM
Home > Resources > ROCm™ Resources > ROCm™ Learning Center

## ROCm™ Learning Center

When it comes to solving the world's most profound computational challenges, scientists and researchers need the most powerful and accessible tools at their fingertips. With the ROCm™ open software platform built for GPU computing, HPC and ML developers can now gain access to an array of different open compute languages, compilers, libraries and tools that are both open and portable.

ROCm™ Learning Center offers resources to developers looking to tap the power of accelerated computing. No matter where they are in their journey, from those just getting started to experts in GPU programming, a broad range of technical resources below are designed to meet developers where they are at.

### Getting Started

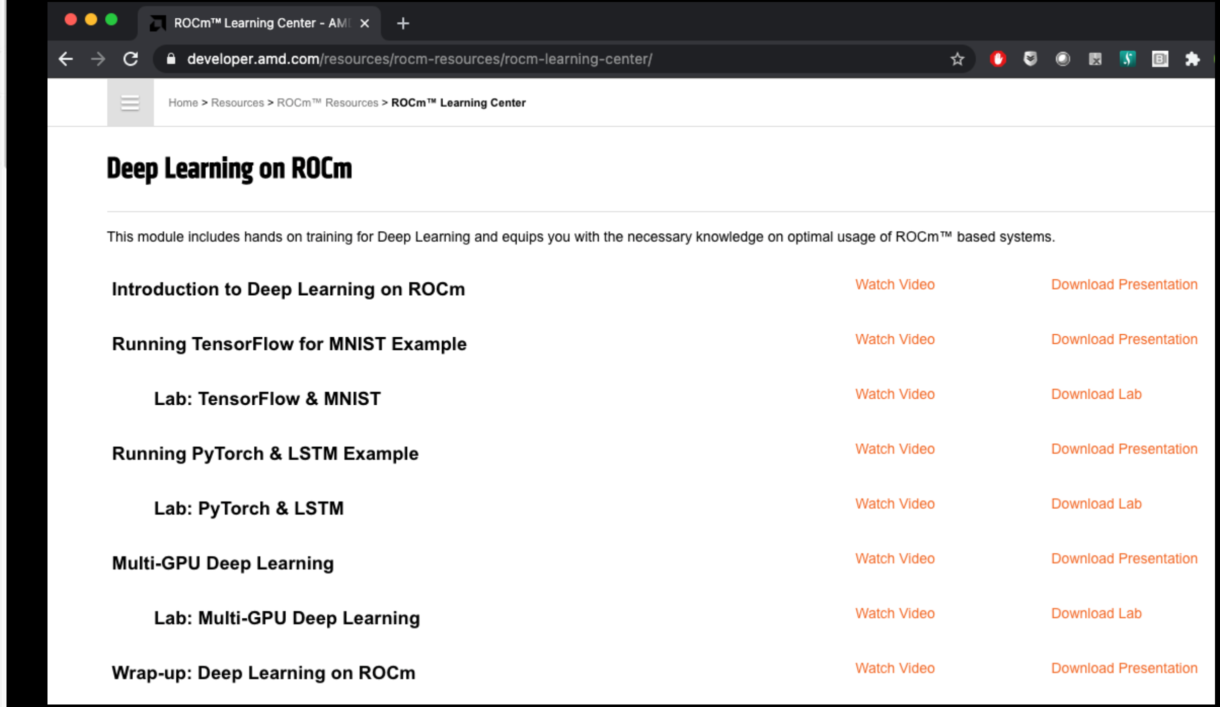
This module offers an introduction and sets you up for success with using ROCm™. The tutorials provides an overview of what ROCm is and how to install the software to get started.

| | | |
|---|---|---|
| Introduction to ROCm™ | Watch Video | Download Presentation |
| ROCm™ Installation | Watch Video | Download Presentation |

### Fundamentals of HIP Programming

This module provides in-depth training on programming with HIP. HIP is a high performance, CUDA-like programming model that is built on an open and portable framework. You will learn everything ranging from the basics of GPU programming to profiling GPU applications to an in-depth knowledge of programming with HIP.

| | | |
|---|---|---|
| Introduction to HIP | Watch Video | Download Presentation |
| Deep Dive into GPU and Performance Optimizations | Watch Video | Download Presentation |
| Your First HIP Code: Vector Add | Watch Video | Download Presentation |



ROCm™ Learning Center - AMD
developer.amd.com/resources/rocm-resources/rocm-learning-center/
Home > Resources > ROCm™ Resources > ROCm™ Learning Center

### Deep Learning on ROCm

This module includes hands on training for Deep Learning and equips you with the necessary knowledge on optimal usage of ROCm™ based systems.

| | | |
|---|---|---|
| Introduction to Deep Learning on ROCm | Watch Video | Download Presentation |
| Running TensorFlow for MNIST Example | Watch Video | Download Presentation |
| Lab: TensorFlow & MNIST | Watch Video | Download Lab |
| Running PyTorch & LSTM Example | Watch Video | Download Presentation |
| Lab: PyTorch & LSTM | Watch Video | Download Lab |
| Multi-GPU Deep Learning | Watch Video | Download Presentation |
| Lab: Multi-GPU Deep Learning | Watch Video | Download Lab |
| Wrap-up: Deep Learning on ROCm | Watch Video | Download Presentation |

- https://developer.amd.com/resources/rocm-resources/rocm-learning-center/

# AMD Math Libraries for GPU (1/2)

| | |
|---|---|
| rocBLAS | Basic Linear Algebra Subroutines |
| rocFFT | Fast Fourier Transforms |
| rocRAND | Random Number Generation |
| rocTHRUST | C++ Parallel Algorithms |
| rocPRIM | Optimized Parallel Primitives |

# AMD Math Libraries for GPU (2/2)

| | |
|---|---|
| **rocSPARSE** | Sparse BLAS, SpMV, etc. |
| **rocSOLVER** | LAPACK Routines |
| **rocALUTION** | Solvers and preconditioners for sparse linear systems |

**See github.com/ROCm-Developer-Tools/HIP → hip_porting_guide.md for a complete list**

# Example: Calling BLAS Level 3 Routines (SGEMM)

## Calling standard math library (host):

```
void example_sgemm_host() {
    // Declarations omitted.



    cblas_sgemm(transa, transb,
                m, n, k,
                alpha, A, lda,
                B, ldb,
                beta, C, ldc);


}
```

## Calling rocBLAS math library (GPU):

```
void example_sgemm_gpu() {
    // Declarations omitted.
    // Assume matrix on GPU.
    rocblas_handle handle;
    rocblas_create_handle(&handle);
    rocblas_sgemm(handle,
                  transa, transb,
                  m, n, k,
                  &alpha, A, lda,
                  B, ldb,
                  &beta, C, ldc);
    rocblas_destroy_handle(handle);
}
```

Library interface almost identical and easy to port from host usage to GPU usage.

# OpenMP Productive Programming for GPUs

# Example: saxpy() – Very Common Operation in HPC Codes

```
void saxpy(size_t n, float a,
           float * x, float * y) {
    double t = 0.0;
    double tb, te;
    tb = omp_get_wtime();
    #pragma omp parallel for firstprivate(a)
    for (int i = 0; i < n; i++) {
        y[i] = a * x[i] + y[i];
    }
    te = omp_get_wtime();
    t = te - tb;
    printf("Time of kernel: %lf\n", t);
}
```

Timing code (not needed, just to have a bit more code to show ☺)

This is the code we want to execute on a target device (i.e., GPU)

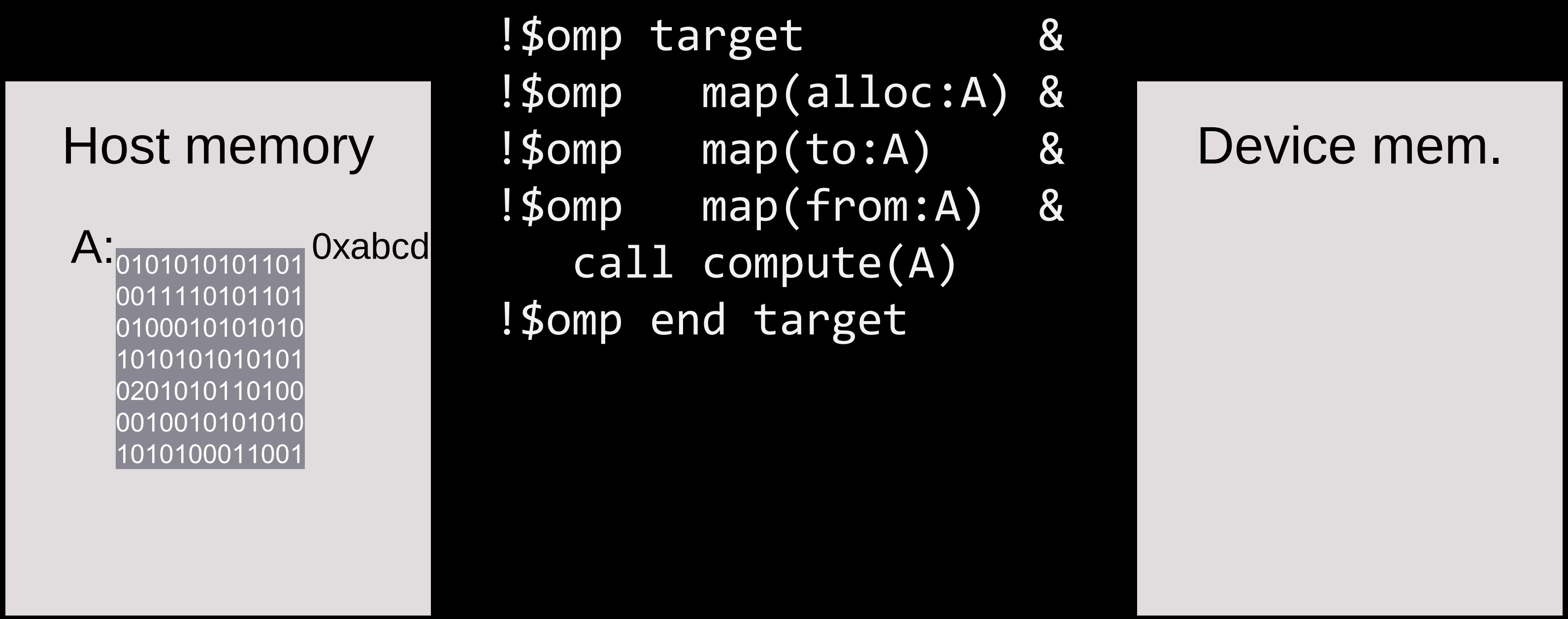
Timing code (not needed, just to have a bit more code to show ☺)

Don’t do this at home!
Use a math library for this!

```
clang -fopenmp <other compiler flags> -o saxpy.o -c saxpy.c
```

# OpenMP: Heterogenous Programming (aka Offloading)

- As of version 4.0, the OpenMP API supports offloading computation to GPUs.
- Similar device model compared to other heterogenous programming models:
  - One host for “traditional” multi-threading.
  - Multiple GPUs of the same kind for offloading.
  - GPU devices are accessible though a device ID (from 0 to *n*-1 for *n* devices).



```
!$omp target          &
!$omp   map(alloc:A) &
!$omp   map(to:A)     &
!$omp   map(from:A)   &
   call compute(A)
!$omp end target
```

Copyright OpenMP ARB, used with permission

# Example: saxpy() on a GPU

```
void saxpy(size_t n, float a,
           float * x, float * y) {
    double t = 0.0;
    double tb, te;
    tb = omp_get_wtime();
    #pragma omp target \
                teams distribute parallel for \
                map(to:x[0:SZ]) map(tofrom:y[0:SZ])
    for (int i = 0; i < SZ; i++) {
        y[i] = a * x[i] + y[i];
    }
    te = omp_get_wtime();
    t = te - tb;
    printf("Time of kernel: %lf\n", t);
}
```

- No need for boilerplate code to
  - allocate memory,
  - transfer data, and
  - synchronize GPU execution.
- Tightly integrates with multi-threaded execution on the host
- Directive-based language
  - Fortran!
  - (No need to switch to a different base language.)
- Descriptive and prescriptive model

```
clang -fopenmp --offload-arch=gfx908 <other compiler flags> -o saxpy.o -c saxpy.c
```

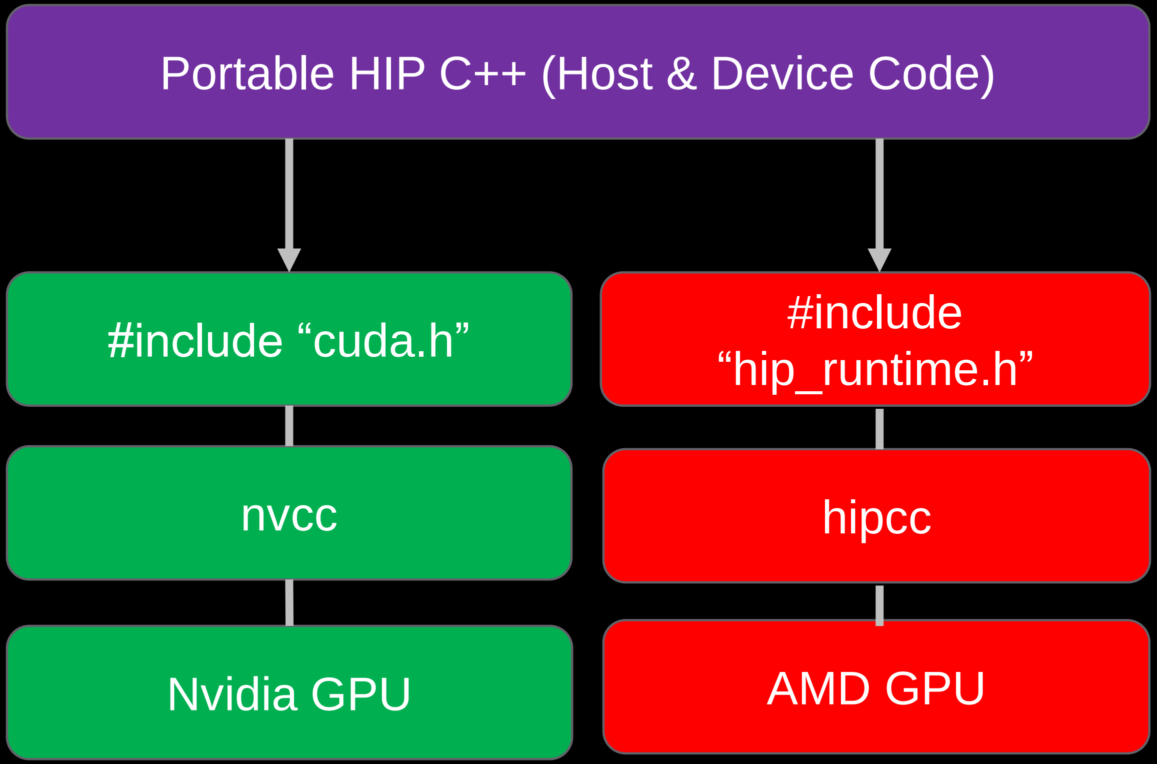
# HIP Programming and HIPifying Code

# What is HIP?

AMD **Heterogeneous-compute Interface for Portability**, or **HIP**, is a 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.

HIP:

- Is open-source!
- Provides an API for an application to leverage GPU acceleration for the hardware of your choice.
- Syntactically similar to the CUDA® API enabling developers familiar with CUDA programming to easily extend their knowledge to new hardware platforms.
- Most CUDA API calls can be converted in place.
- Supports a strong subset of CUDA runtime functionality and enables creative developers to innovate on multiple hardware platforms.

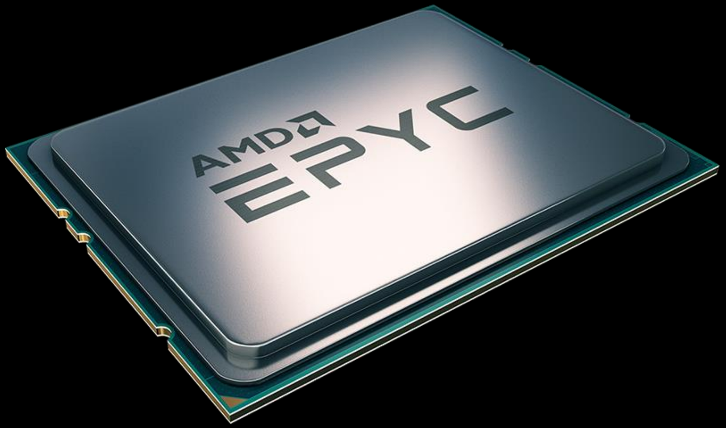


# A Tale of Host and Device

Source code in HIP has two flavors: Host code and Device code

- The host is the CPU.
- Host code runs here.
- Usual C++ syntax and features.
- Entry point is the ‘main’ function.
- HIP API can be used to create device buffers, move between host and device, and launch device code.

- The device is the GPU.
- Device code runs here.
- Device codes are launched via “kernels”
- Instructions from the Host are enqueued into “streams”.



# HIP API

- Device Management:
  - `hipSetDevice(), hipGetDevice(), hipGetDeviceProperties()`
- Memory Management
  - `hipMalloc(), hipMemcpy(), hipMemcpyAsync(), hipFree(), hipHostMalloc()`
- Streams
  - `hipStreamCreate(), hipSynchronize(), hipStreamSynchronize(), hipStreamFree()`
- Events
  - `hipEventCreate(), hipEventRecord(), hipStreamWaitEvent(), hipEventElapsedTime()`
- Device Kernels
  - `__global__, __device__, hipLaunchKernelGGL()`
- Device code
  - `threadIdx, blockIdx, blockDim, __shared__`
  - 200+ math functions covering entire CUDA math library.
- Error handling
  - `hipGetLastError(), hipGetErrorString()`

# HIP Kernel for saxpy()

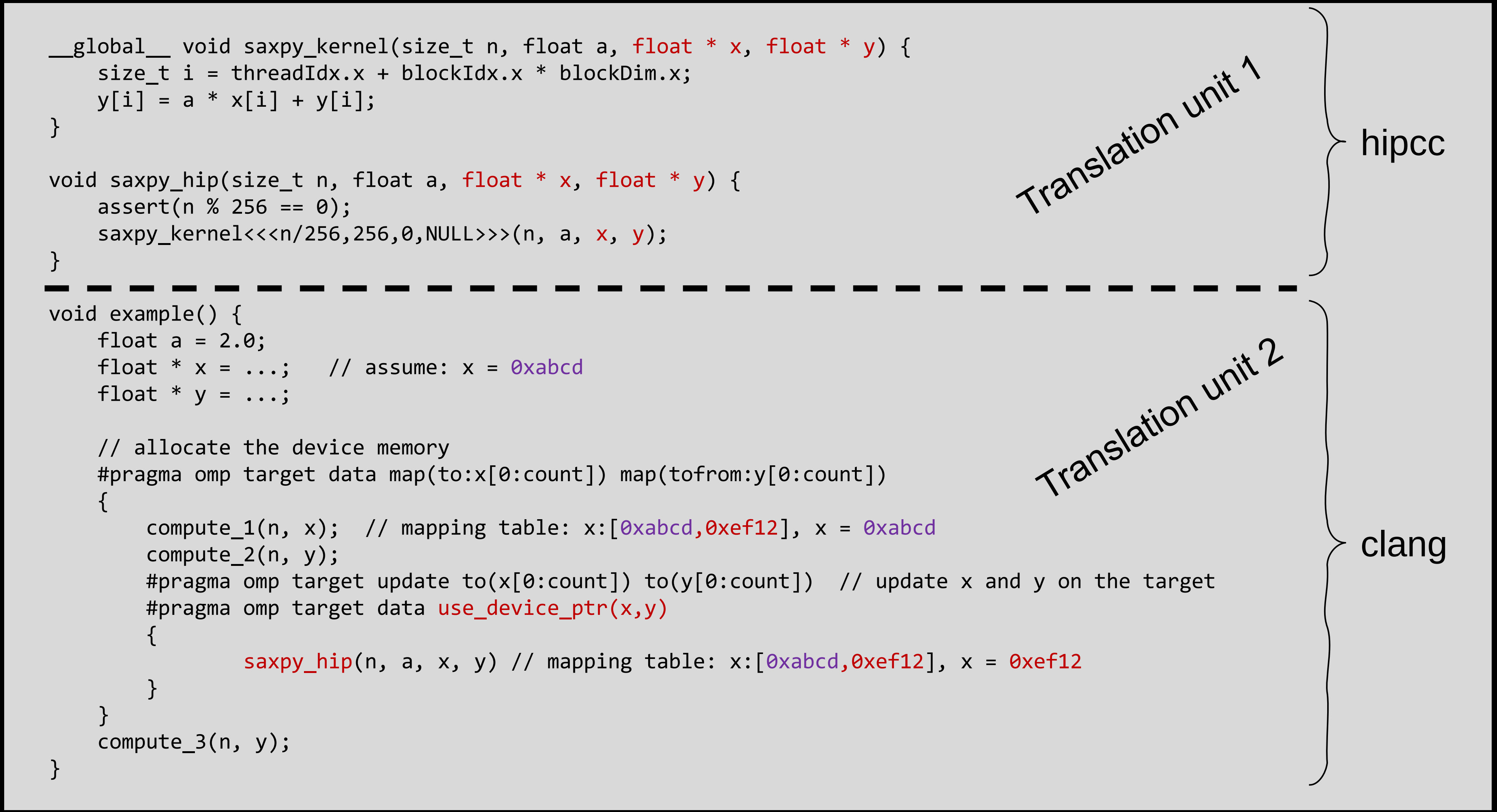
```
__global__ void saxpy_kernel(size_t n, float a, float * x, float * y) {
    size_t i = threadIdx.x + blockIdx.x * blockDim.x;
    y[i] = a * x[i] + y[i];
}

void saxpy(size_t n, float a, float * x, float * y) {
    assert(n % 256 == 0);
    saxpy_kernel<<<n/256,256,0,NULL>>>(n, a, x, y);
}
```

# AOMP Implementation Status

- Call HIP kernel with OpenMP-managed buffers (`use_device_ptr`) ✓
- Call OpenMP kernels with HIP-managed buffers (`is_device_ptr`) ✓
- HIP and OpenMP kernels co-existence in same translation unit ✗

# Mixing OpenMP Offload and HIP Kernels



```
__global__ void saxpy_kernel(size_t n, float a, float * x, float * y) {
    size_t i = threadIdx.x + blockIdx.x * blockDim.x;
    y[i] = a * x[i] + y[i];
}

void saxpy_hip(size_t n, float a, float * x, float * y) {
    assert(n % 256 == 0);
    saxpy_kernel<<<n/256,256,0,NULL>>>(n, a, x, y);
}
```

Translation unit 1

hipcc

```
void example() {
    float a = 2.0;
    float * x = ...;   // assume: x = 0xabcd
    float * y = ...;

    // allocate the device memory
    #pragma omp target data map(to:x[0:count]) map(tofrom:y[0:count])
    {
        compute_1(n, x);  // mapping table: x:[0xabcd,0xef12], x = 0xabcd
        compute_2(n, y);
        #pragma omp target update to(x[0:count]) to(y[0:count])  // update x and y on the target
        #pragma omp target data use_device_ptr(x,y)
        {
                saxpy_hip(n, a, x, y) // mapping table: x:[0xabcd,0xef12], x = 0xef12
        }
    }
    compute_3(n, y);
}
```

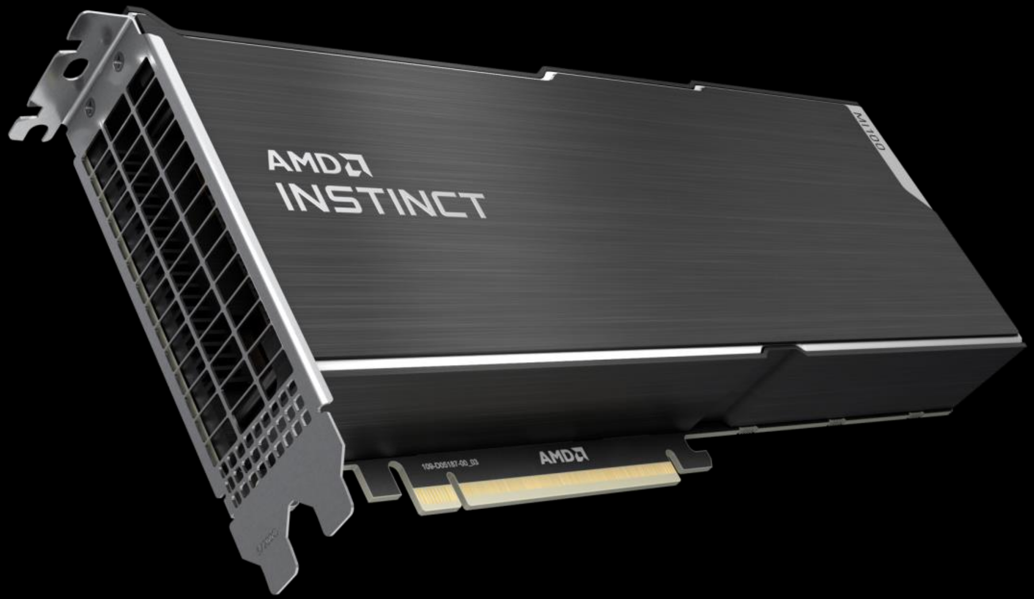
Translation unit 2

clang

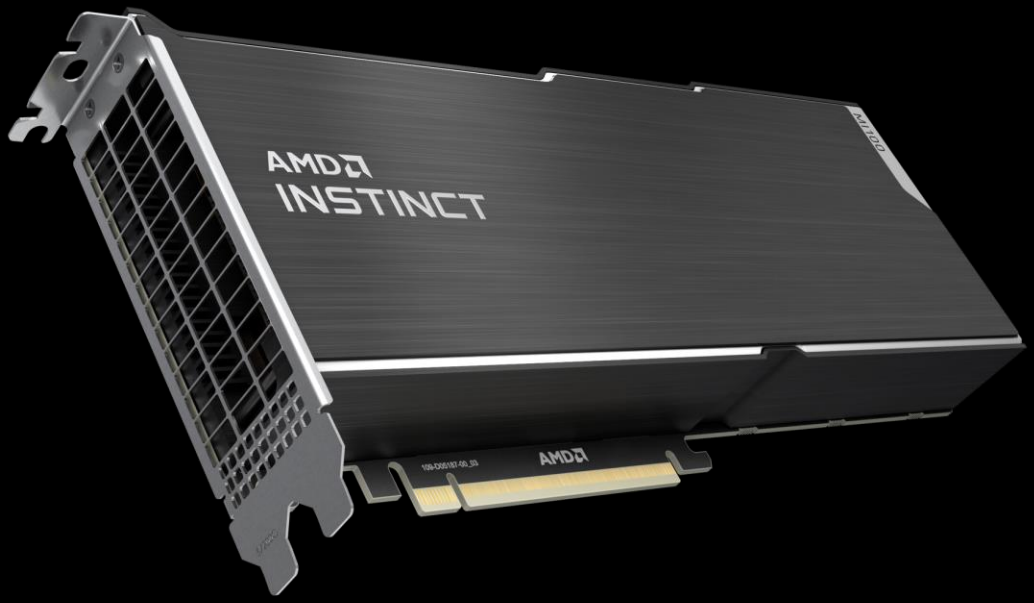
# Summary

- AMD Instinct™ GPUs
  - High-performance GPU architecture designed for HPC and AI/ML
  - >10TF FP64 performance

- AMD ROCm™ Software
  - Open-source!
  - Standards based: OpenMP
  - Portable: OpenMP, HIP
  - Easy to port: HIPification



# Q&A



# Endnotes

**MI100-05** – Slide 5

Calculations performed by AMD Performance Labs as of Sep 18, 2020 for the AMD Instinct™ MI100 accelerator at 1,502 MHz peak boost engine clock resulted in 11.535 TFLOPS peak theoretical double precision (FP64) floating-point performance. The results calculated for Radeon Instinct™ MI50 GPU at 1,725 MHz peak engine clock resulted in 6.62 TFLOPS FP64. Server manufacturers may vary configuration offerings yielding different results. MI100-05

**MI100-14**

Testing Conducted by AMD performance labs as of October 30th, 2020, on three platforms and software versions typical for the launch dates of the Radeon Instinct MI25 (2018), MI50 (2019) and AMD Instinct MI100 GPU (2020) running the benchmark application Quicksilver. MI100 platform (2020): Gigabyte G482-Z51-00 system comprised of Dual Socket AMD EPYC™ 7702 64-Core Processor, AMD Instinct™ MI100 GPU, ROCm™ 3.10 driver, 512GB DDR4, RHEL 8.2 MI50 platform (2019): Supermicro® SYS-4029GP-TRT2 system comprised of Dual Socket Intel Xeon® Gold® 6132, Radeon Instinct™ MI50 GPU, ROCm 2.10 driver, 256 GB DDR4, SLES15SP1 MI25 platform (2018): Supermicro SYS-4028GR-TR2 system comprised of Dual Socket Intel Xeon CPU E5-2690, Radeon Instinct™ MI25 GPU, ROCm 2.0.89 driver, 246GB DDR4 system memory, Ubuntu 16.04.5 LTS. MI100-14

**MI100-15**

Testing Conducted by AMD performance labs as of October 30th, 2020, on three platforms and software versions typical for the launch dates of the Radeon Instinct MI25 (2018), MI50 (2019) and AMD Instinct MI100 GPU (2020) running the benchmark application TensorFlow ResNet 50 FP 16 batch size 128. MI100 platform (2020): Gigabyte G482-Z51-00 system comprised of Dual Socket AMD EPYC™ 7702 64-Core Processor, AMD Instinct™ MI100 GPU, ROCm™ 3.10 driver, 512GB DDR4, RHEL 8.2 MI50 platform (2019): Supermicro® SYS-4029GP-TRT2 system comprised of Dual Socket Intel Xeon® Gold® 6254, Radeon Instinct™ MI50 GPU, ROCm 3.0.6 driver, 338 GB DDR4, Ubuntu® 16.04.6 LTS MI25 platform (2018): a Supermicro SYS-4028GR-TR2 system comprised of Dual Socket Intel Xeon CPU E5-2690, Radeon Instinct™ MI25 GPU, ROCm 2.0.89 driver, 246GB DDR4 system memory, Ubuntu 16.04.5 LTS. MI100-15

# Disclaimer and Attributions

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. 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.

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 MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY 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.

AMD