

IRIS

An Intelligent Task-Based Runtime System
for Extremely Heterogeneous Architectures

Jungwon Kim

IRIS mini workshop

January 13, 2021

An Intelligent Task-Based Runtime System *for Extremely Heterogeneous Architectures*

Systems	Snapdragon	Jetson	Zynq	DGX	Oswald	Summit	Frontier
CPU	ARM	ARM	ARM	Intel	Intel	IBM	AMD
GPU	Qualcomm	NVIDIA		NVIDIA	NVIDIA	NVIDIA	AMD
FPGA			Xilinx		Intel		
DSP	Qualcomm						



- Target ExCL systems
- Our question: How can we write portable programs for the target extremely heterogeneous architectures?

OpenCL is *NOT* Portable Enough

Systems	Snapdragon	Jetson	Zynq	DGX	Oswald	Summit	Frontier
CPU	ARM	ARM	ARM	Intel	Intel	IBM	AMD
GPU	Qualcomm	NVIDIA		NVIDIA	NVIDIA	NVIDIA	AMD
FPGA			Xilinx		Intel		
DSP	Qualcomm						



- OpenCL is an open cross-platform standard for accelerator programming.
- Accelerator programming model: host + kernel
 - Kernel: time-consuming and compute-intensive code to be offloaded to accelerators
 - Host: orchestration of kernel executions
 - H2D memory copy → kernel execution → D2H memory copy

NVIDIA CUDA

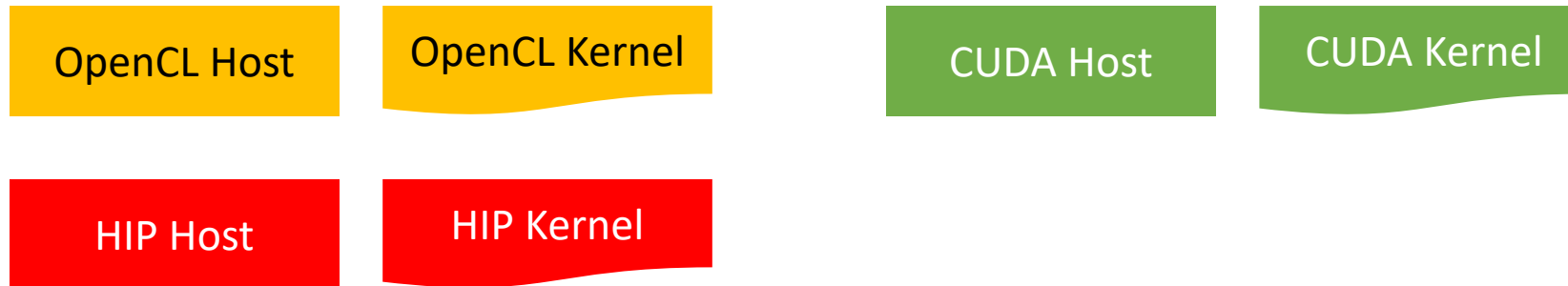
Systems	Snapdragon	Jetson	Zynq	DGX	Oswald	Summit	Frontier
CPU	ARM	ARM	ARM	Intel	Intel	IBM	AMD
GPU	Qualcomm	NVIDIA		NV	NV	NVIDIA	AMD
FPGA			Xilinx		Intel		
DSP	Qualcomm						



- NVIDIA CUDA presents more device-specific optimizations than NVIDIA OpenCL for their devices.

AMD HIP

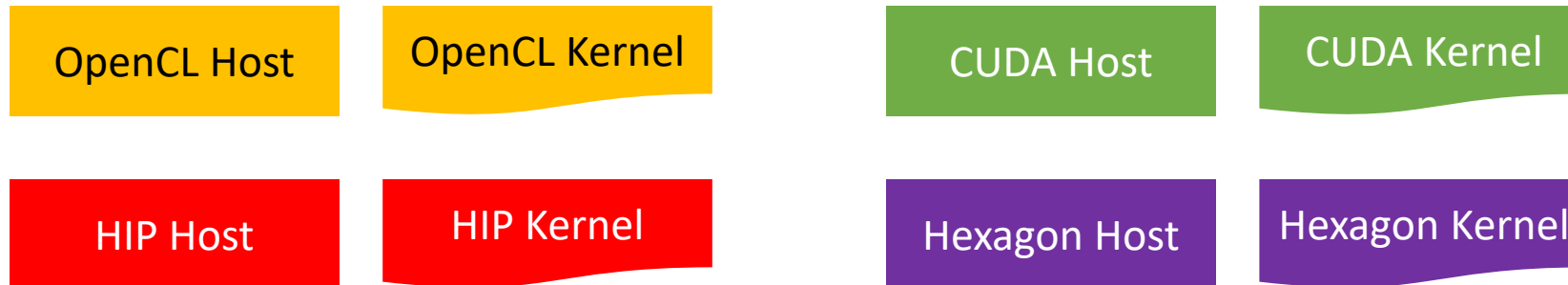
Systems	Snapdragon	Jetson	Zynq	DGX	Oswald	Summit	Frontier	
CPU	ARM	ARM	ARM	Intel	Intel	IBM	AMD	
GPU	Qualcomm	NVIDIA		NV	NV	NVIDIA	AMD	AMD
FPGA			Xilinx		Intel			
DSP	Qualcomm							



- AMD HIP presents more device-specific optimizations than AMD OpenCL for their devices.

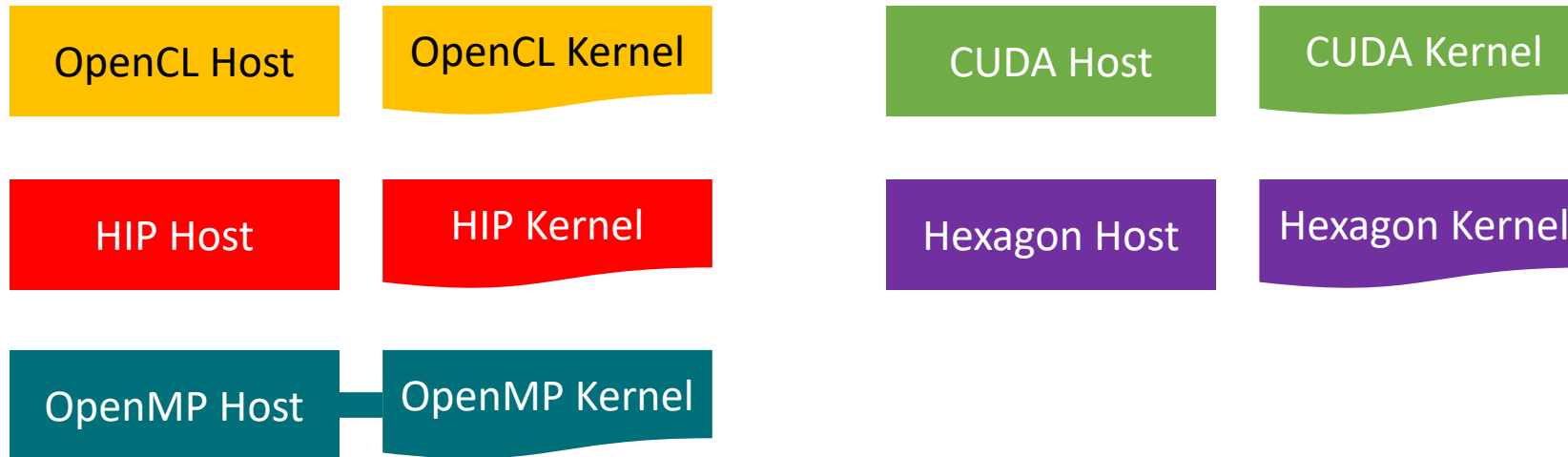
Qualcomm Hexagon

Systems	Snapdragon	Jetson	Zynq	DGX	Oswald	Summit	Frontier	
CPU	ARM	ARM	ARM	Intel	Intel	IBM	AMD	
GPU	Qualcomm	NVIDIA		NV	NV	NVIDIA	AMD	AMD
FPGA			Xilinx		Intel			
DSP	Qualcomm							



OpenMP w/o Offloading

Systems	Snapdragon	Jetson	Zynq	DGX		Oswald		Summit	Frontier	
CPU	ARM	ARM	ARM	Intel	Intel	Intel	Intel	IBM	AMD	
GPU	Qualcomm	NVIDIA		NV	NV	NV	NV	NVIDIA	AMD	AMD
FPGA			Xilinx			Intel				
DSP	Qualcomm									

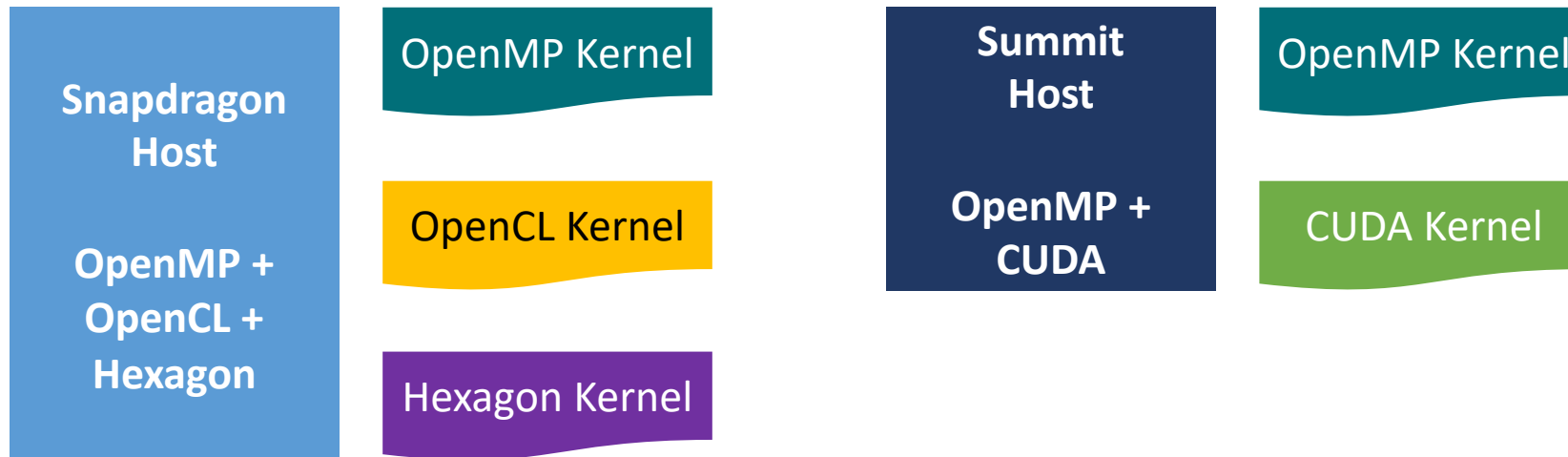


Single source code:

Host (sequential code) + Kernel (parallel loop)

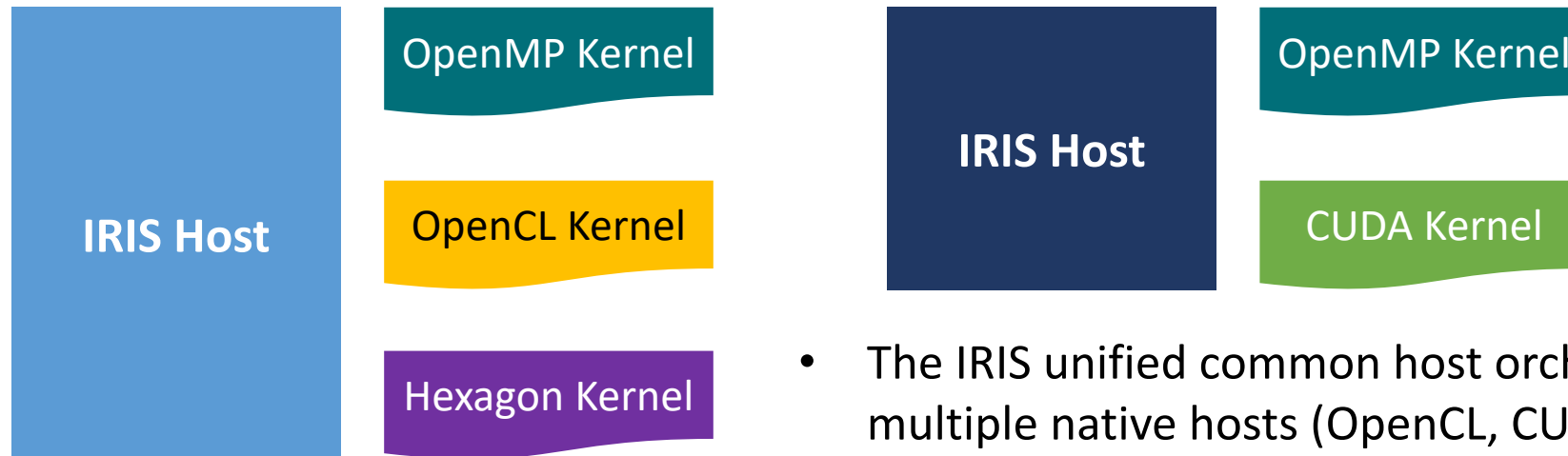
Not Portable Programs

Systems	Snapdragon	Jetson	Zynq	DGX		Oswald		Summit	Frontier	
CPU	ARM	ARM	ARM	Intel	Intel	Intel	Intel	IBM	AMD	
GPU	Qualcomm	NVIDIA		NV	NV	NV	NV	NVIDIA	AMD	AMD
FPGA			Xilinx			Intel				
DSP	Qualcomm									



IRIS Unified Common Host

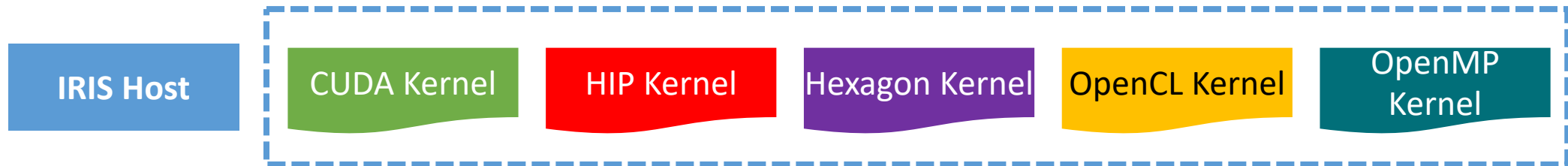
Systems	Snapdragon	Jetson	Zynq	DGX		Oswald		Summit	Frontier	
CPU	ARM	ARM	ARM	Intel	Intel	Intel	Intel	IBM	AMD	
GPU	Qualcomm	NVIDIA		NV	NV	NV	NV	NVIDIA	AMD	AMD
FPGA			Xilinx			Intel				
DSP	Qualcomm									



- The IRIS unified common host orchestrates the multiple native hosts (OpenCL, CUDA, HIP, Hexagon, and OpenMP) into a single context.
- C/C++/Fortran/Python APIs

IRIS Unified Common Host + Native Kernels

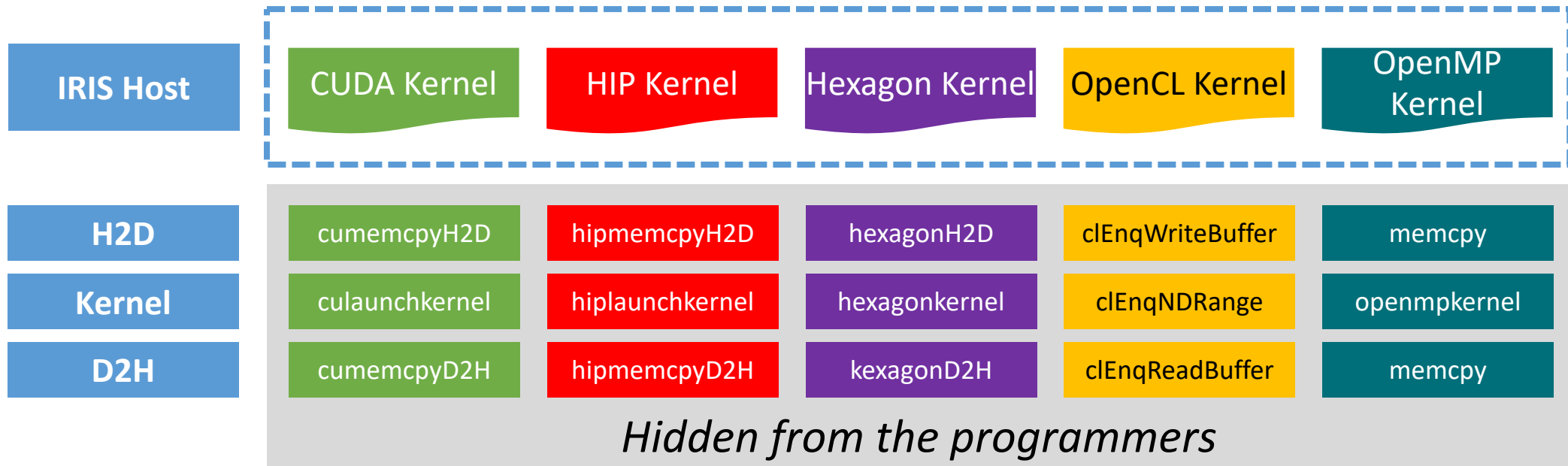
Systems	Snapdragon	Jetson	Zynq	DGX		Oswald		Summit	Frontier	
CPU	ARM	ARM	ARM	Intel	Intel	Intel	Intel	IBM	AMD	
GPU	Qualcomm	NVIDIA		NV	NV	NV	NV	NVIDIA	AMD	AMD
FPGA			Xilinx			Intel				
DSP	Qualcomm									



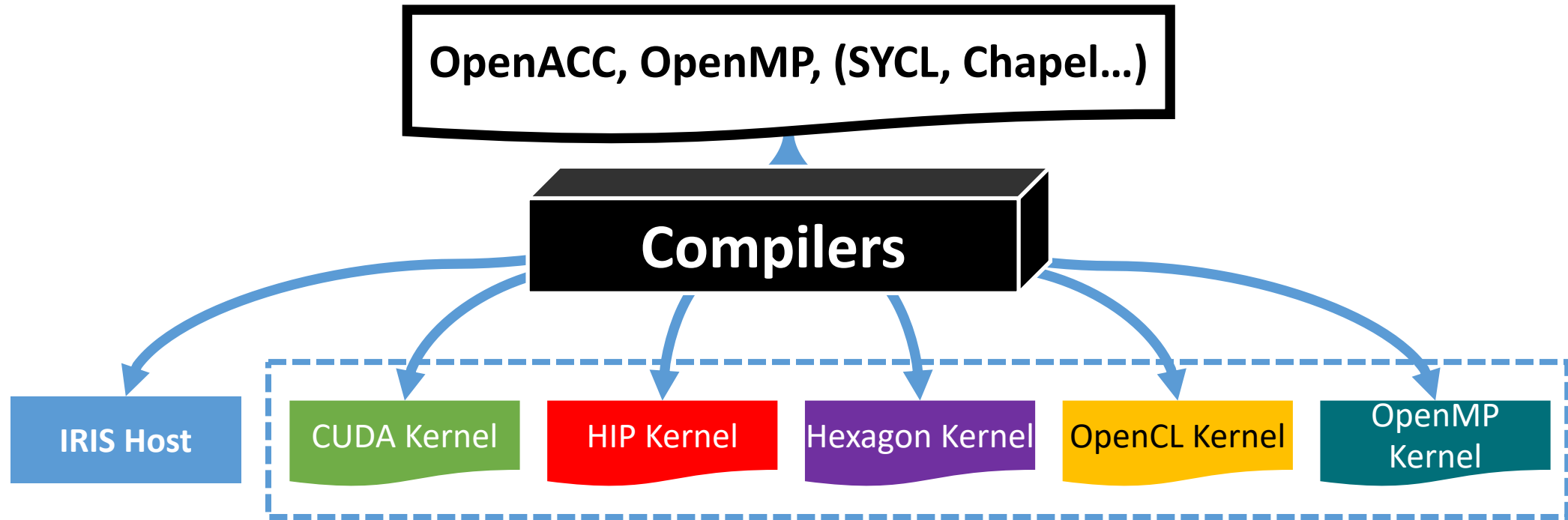
- An IRIS application: a single IRIS host + a set of native kernels
- IRIS applications run on all the target systems.
- How does the IRIS host run the native kernels on the target devices?

IRIS Host *Internally* Calls the Native AP Hosts

Systems	Snapdragon	Jetson	Zynq	DGX		Oswald		Summit	Frontier	
CPU	ARM	ARM	ARM	Intel	Intel	Intel	Intel	IBM	AMD	
GPU	Qualcomm	NVIDIA		NV	NV	NV	NV	NVIDIA	AMD	AMD
FPGA			Xilinx			Intel				
DSP	Qualcomm									



High-Level Programming Models



- Writing code in a **high-level**, portable, parallelizable style
 - Compilers generate IRIS host and native kernels from the HL source code.
- Device-specific optimized kernels generation is the key for high performance.
 - E.g., OpenCL kernels are not performance portable.
 - Tradeoff b/w performance and productivity

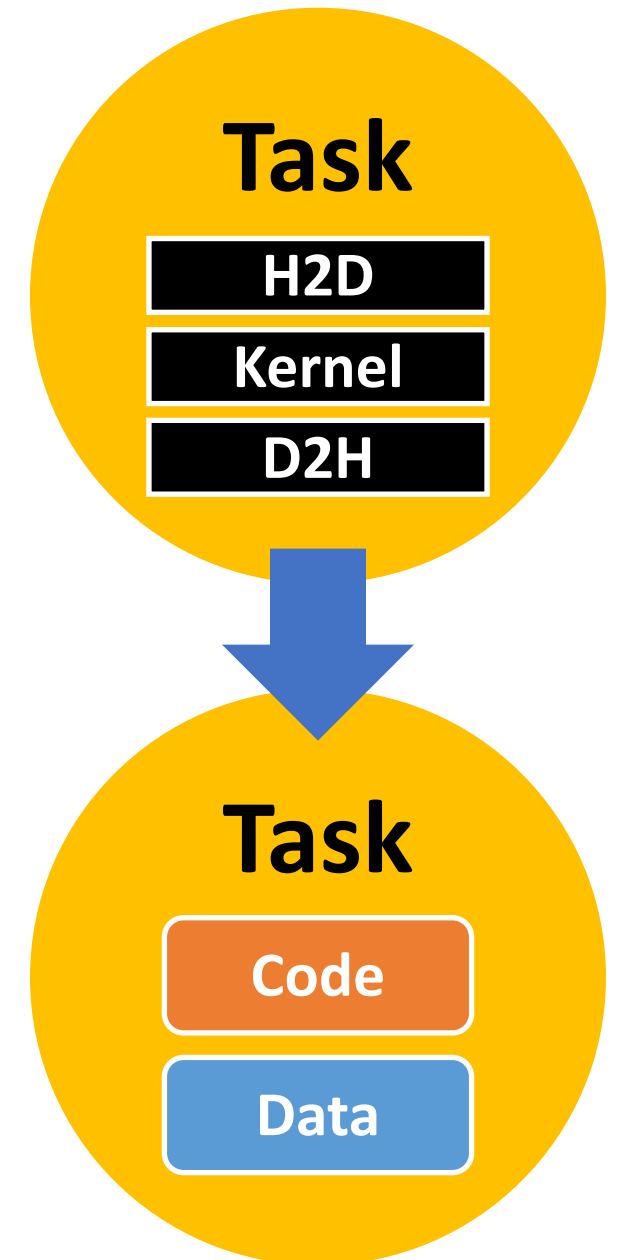
An Intelligent *Task-Based* Runtime System for Extremely Heterogeneous Architectures

Systems	Snapdragon	Jetson	Zynq	DGX		Oswald		Summit	Frontier	
CPU	ARM	ARM	ARM	Intel	Intel	Intel	Intel	IBM	AMD	
GPU	Qualcomm	NVIDIA		NV	NV	NV	NV	NVIDIA	AMD	AMD
FPGA			Xilinx			Intel				
DSP	Qualcomm									

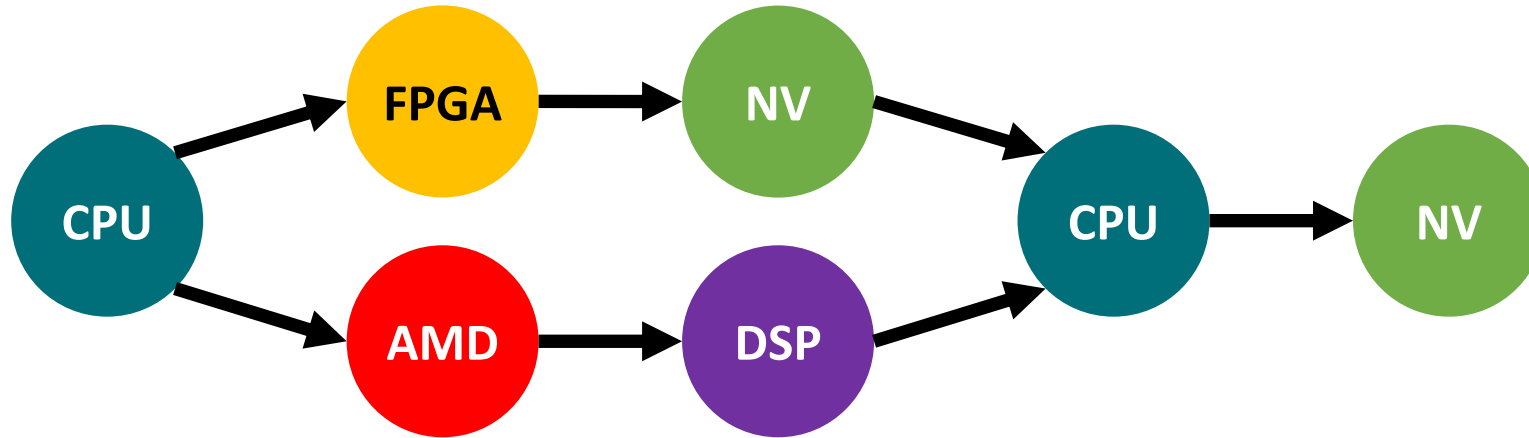
- IRIS simultaneously exploits all available devices in a single application through task-parallelism.

Tasks

- A task consists of one or more commands.
 - Memory copy command (H2D, D2H)
 - Kernel launch command
 - In-order command execution
- Commands define code and data in the task
 - Code: Kernel in the kernel launch command
 - Data: Accessed memories in the commands
- A task runs on a single device.
- The device execute the kernel with data stored on its device memory.



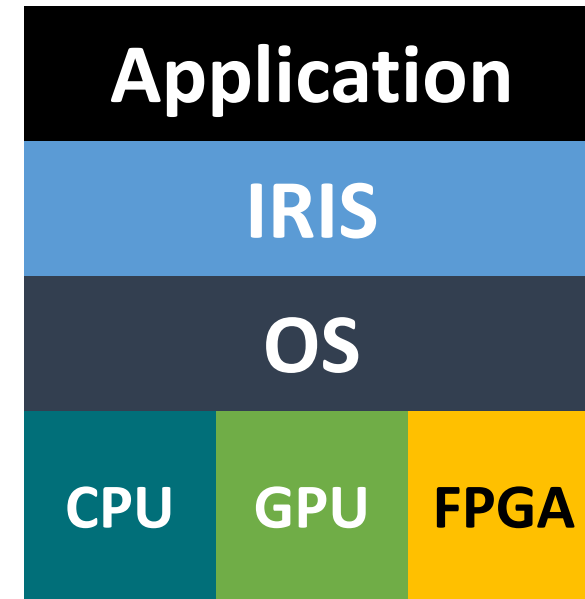
Task Graphs (DAGs)



- *[Revisited]* Accelerator programming model: host + kernel
 - Host: orchestration of kernel executions (H2D → Kernel → D2H)
- IRIS host orchestrates kernel executions by building task graphs.
- Task graphs
 - A node → a task
 - An edge → control dependency between two tasks
- Multiple independent tasks can simultaneously run on multiple devices.

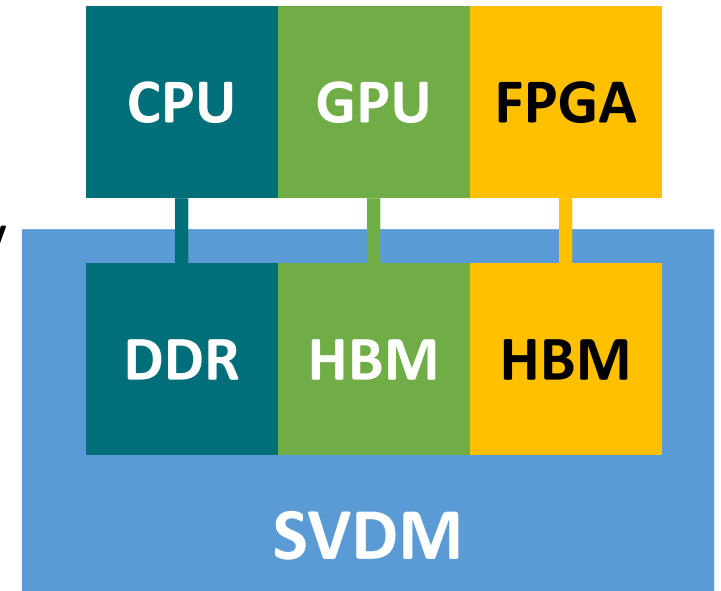
An Intelligent *Task-Based Runtime System* for Extremely Heterogeneous Architectures

- IRIS is a runtime system linked with the applications as a user-level shared library.
- IRIS mainly consists of
 - Shared Virtual Device Memory (SVDM)
 - Task Scheduler (TS)
 - Dynamic Platform Loader (DPL)



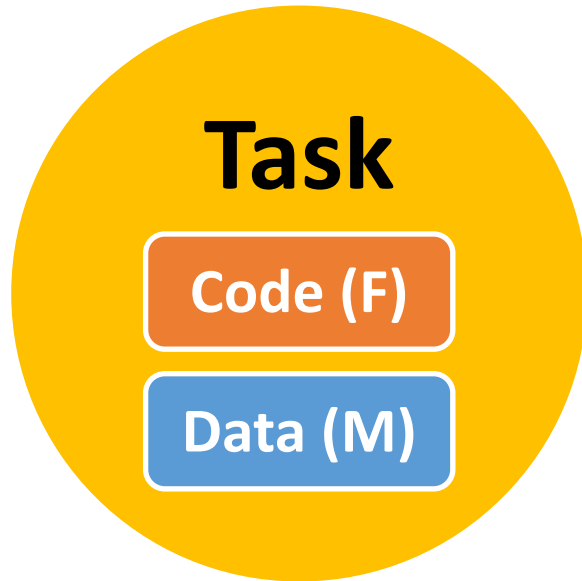
Shared Virtual Device Memory (SVDM)

- IRIS provides SVDM to programmers
 - An illusion of single logical device memory across all physical device memories
 - Physical device memories are hidden from the programmers.
- Data in tasks
 - A logical handle to a region of SVDM
 - Physically allocated on the running device memory
 - Lazy local copy creation
 - Shared across multiple devices
 - Multiple copies on multiple device memories
 - Relaxed memory consistency across devices
 - Synchronization points

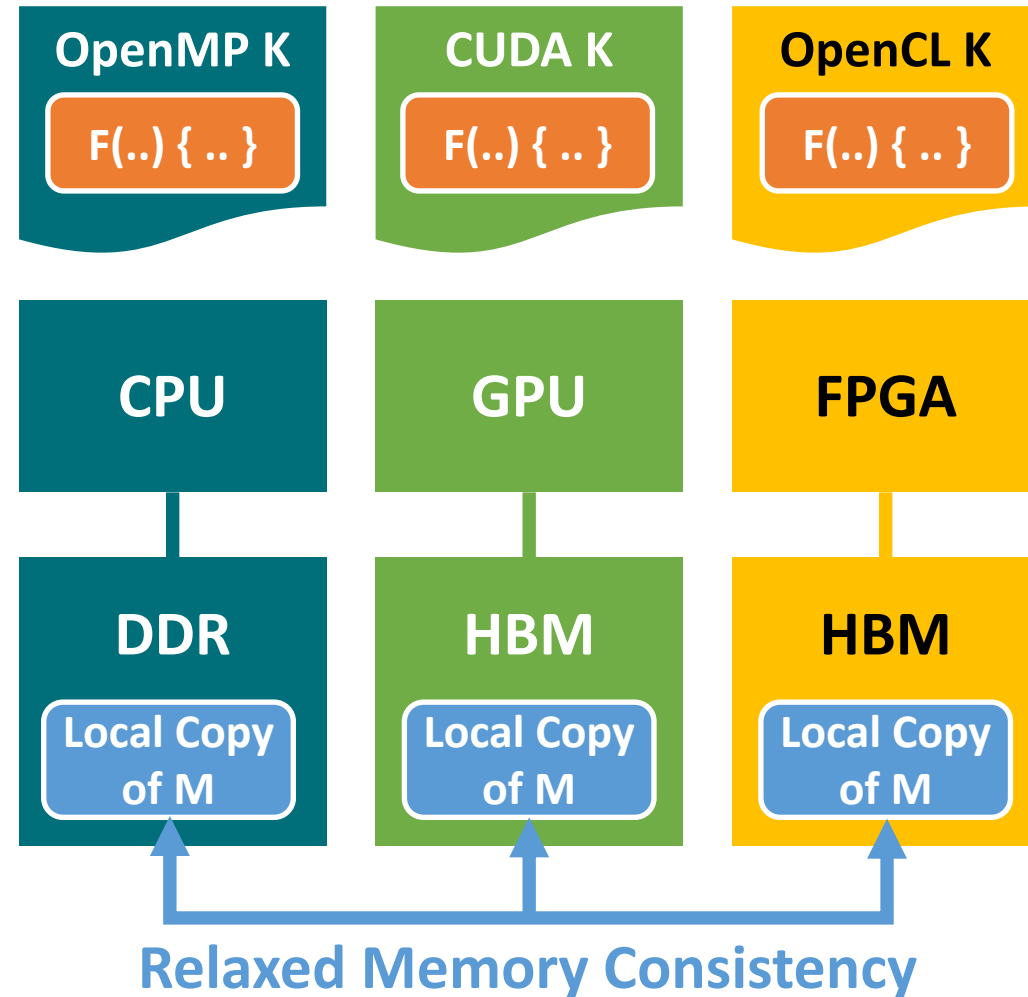


Programmers' view:
3 devices and 1 shared device memory


Multiple Native Kernels + SVDM = Portable Tasks

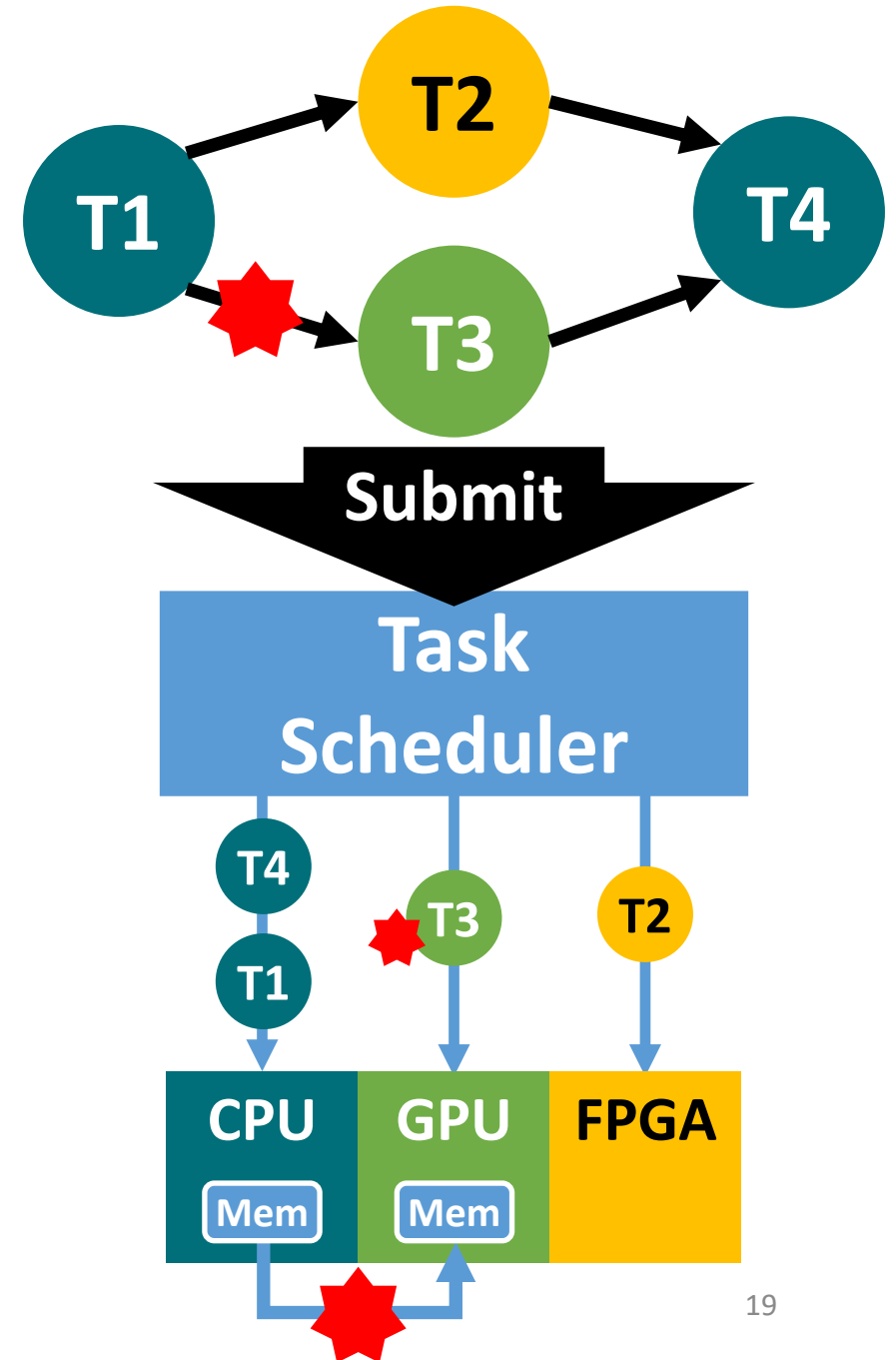


- A task is portable across all devices.

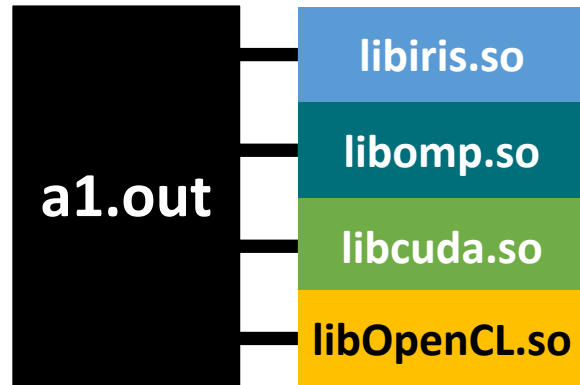


Task Scheduler

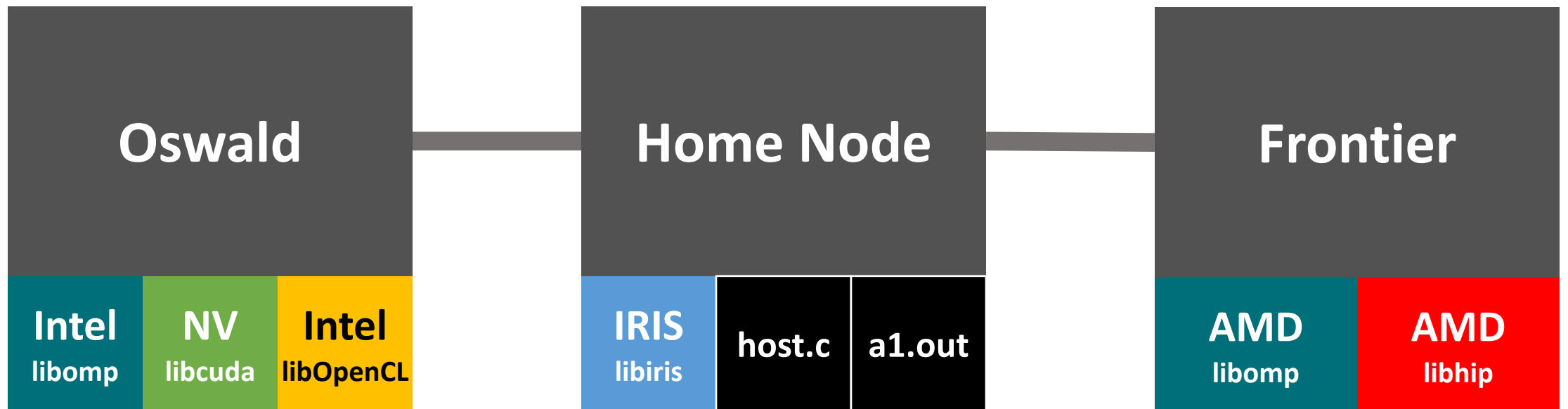
- The host builds task graphs and specifies the target device of each task.
- The host submits the graphs into the task scheduler.
- The task scheduler schedules the tasks in the graphs based on their dependencies.
- Memory (local copies) consistency b/w tasks
 - Edges → synchronization points
 - Inter-device memory copy 



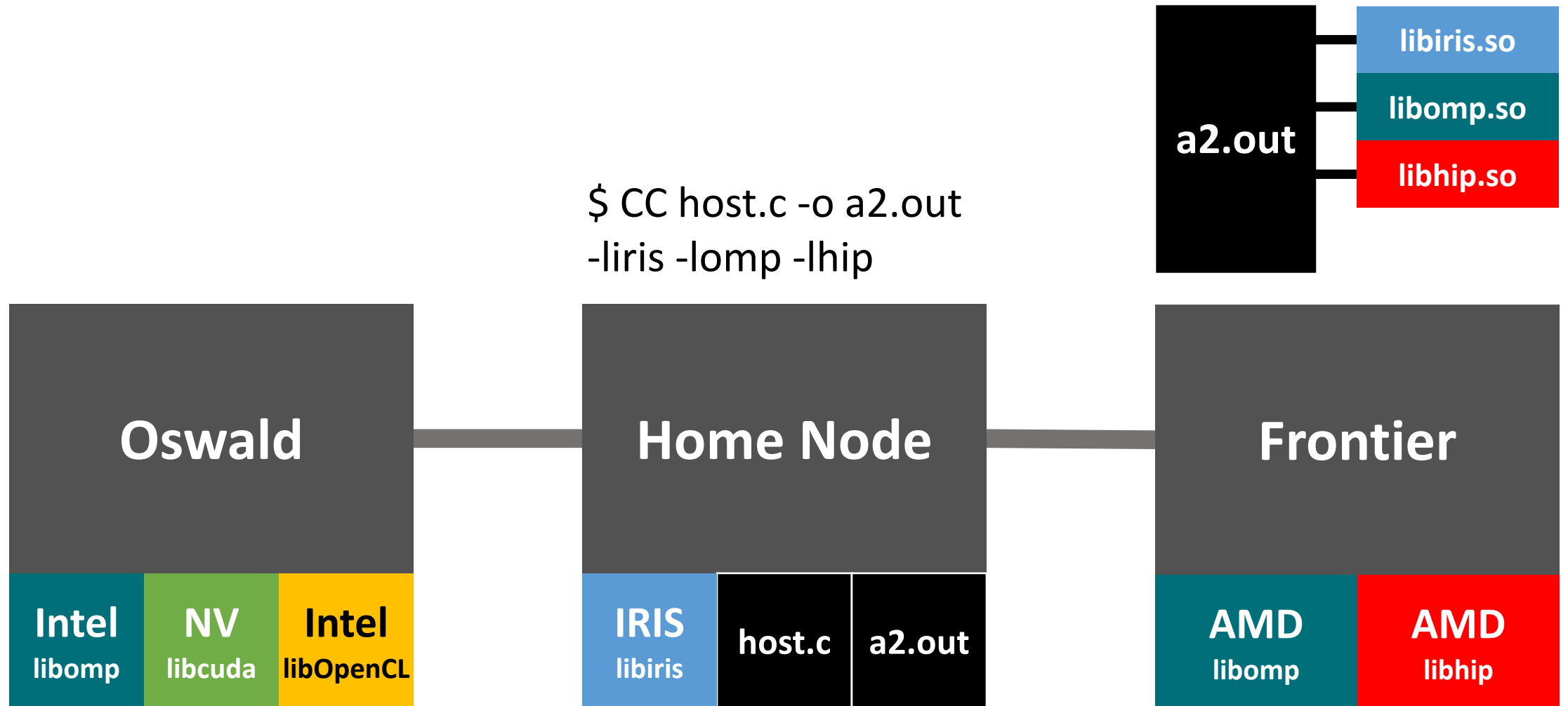
Building host.c on Oswald



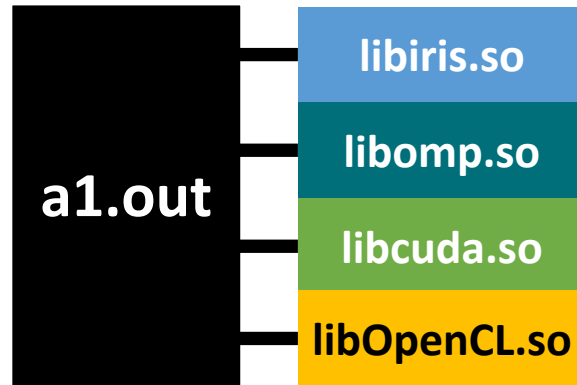
```
$ CC host.c -o a1.out  
-liris -lomp -lcuda -lOpenCL
```



Building host.c on Frontier

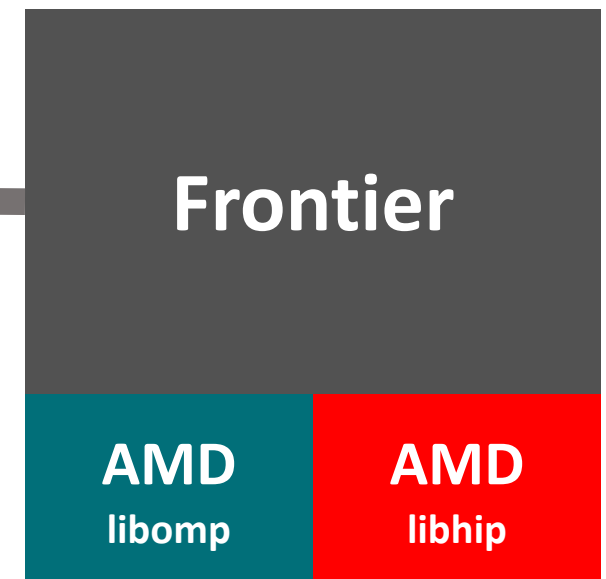
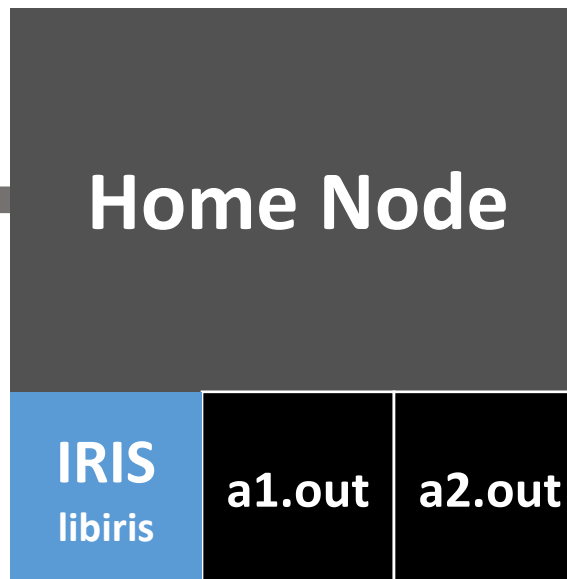
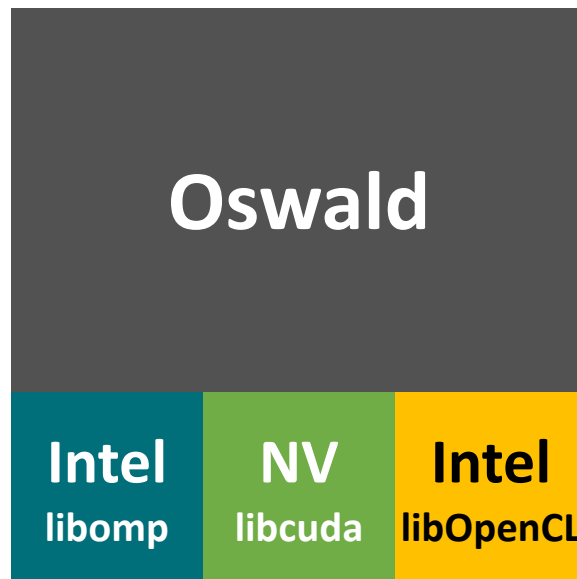
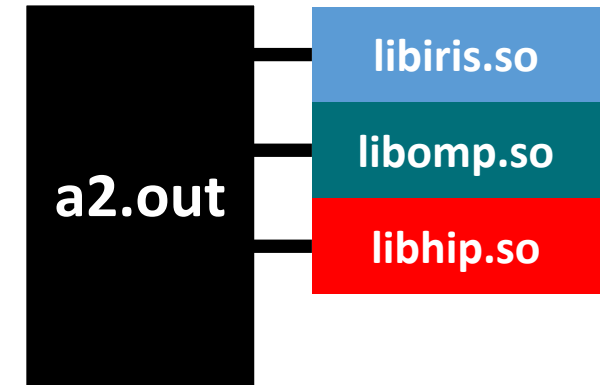


Not Portable Executables and Build Scripts

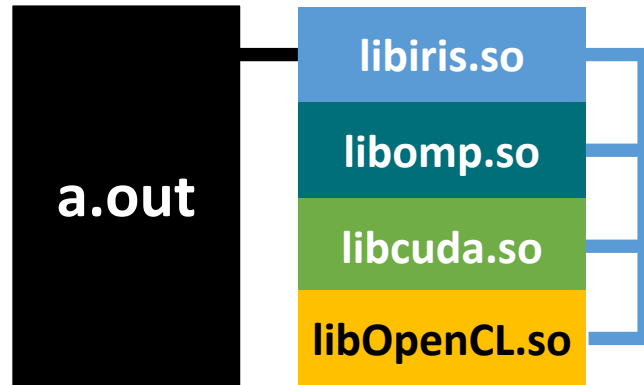


```
$ CC host.c -o a1.out  
-liris -lomp -lcuda -lOpenCL
```

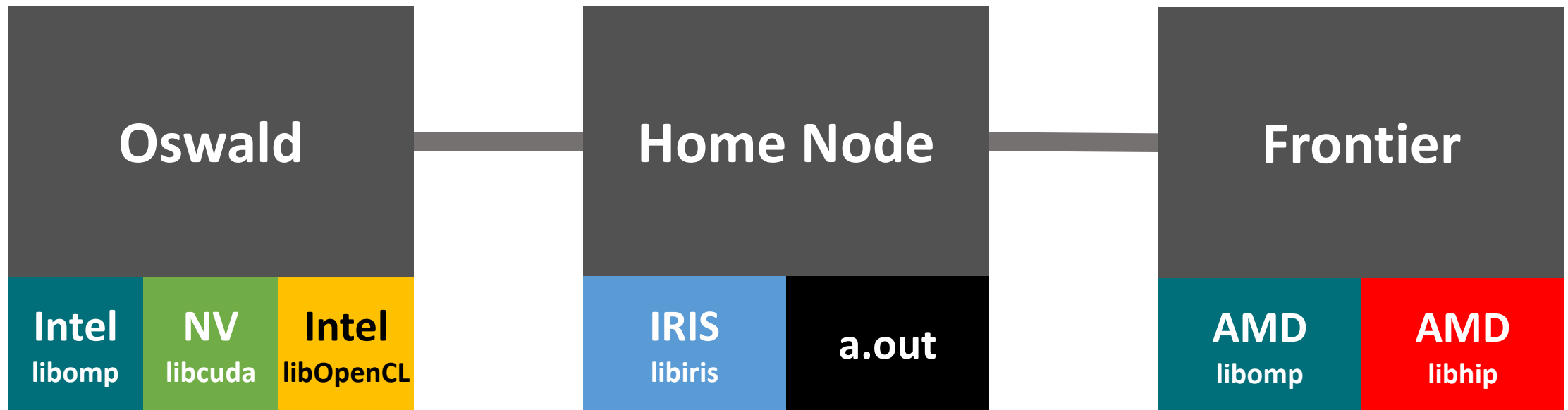
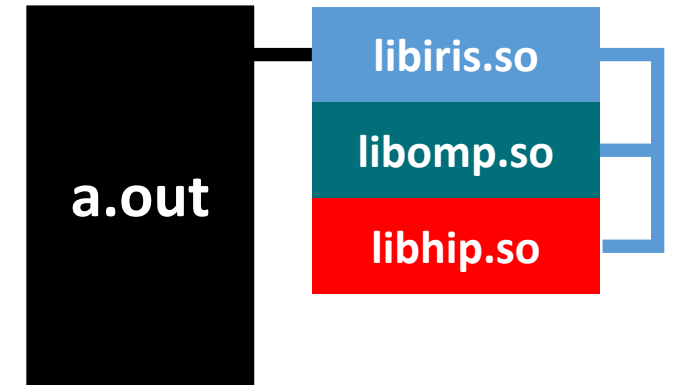
```
$ CC host.c -o a2.out  
-liris -lomp -lhip
```



Dynamic Platform Loader (DPL)



- ```
$ CC host.c -o a.out -liris
```
- DPL automatically loads all available platforms in the system on run time.
  - **Private linkchain**



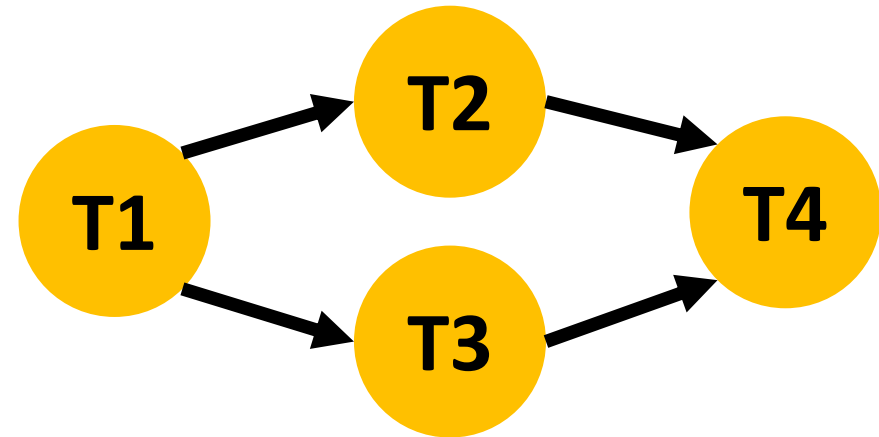
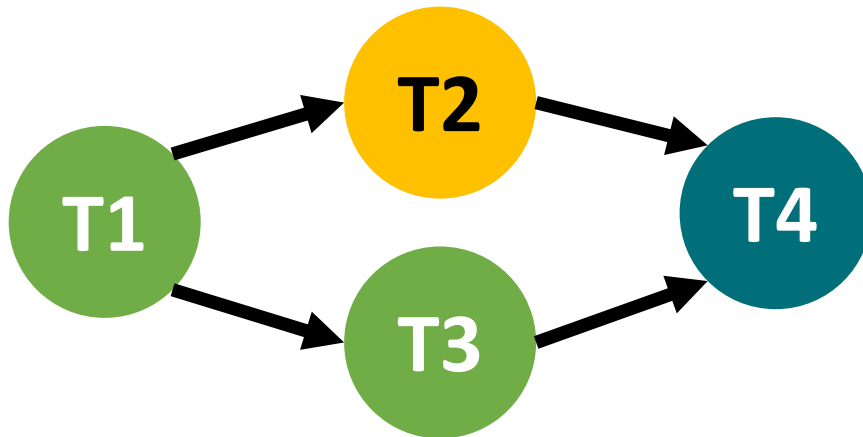
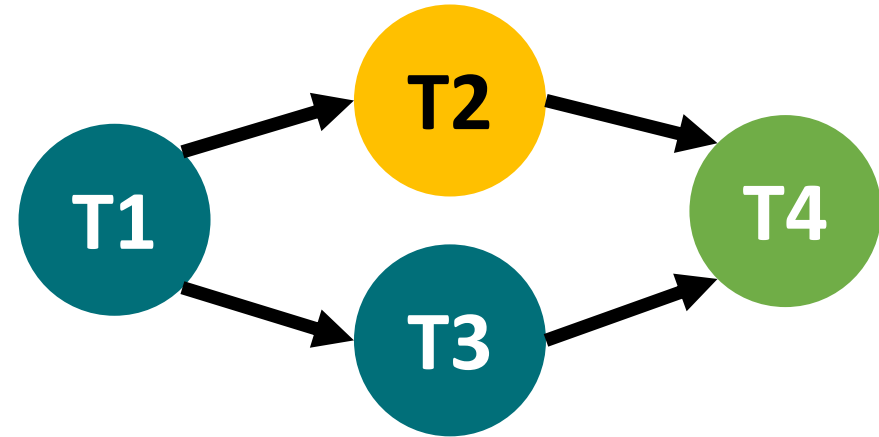
# An *Intelligent* Task-Based Runtime System for Extremely Heterogeneous Architectures

- Intelligent task scheduling makes IRIS intelligent.



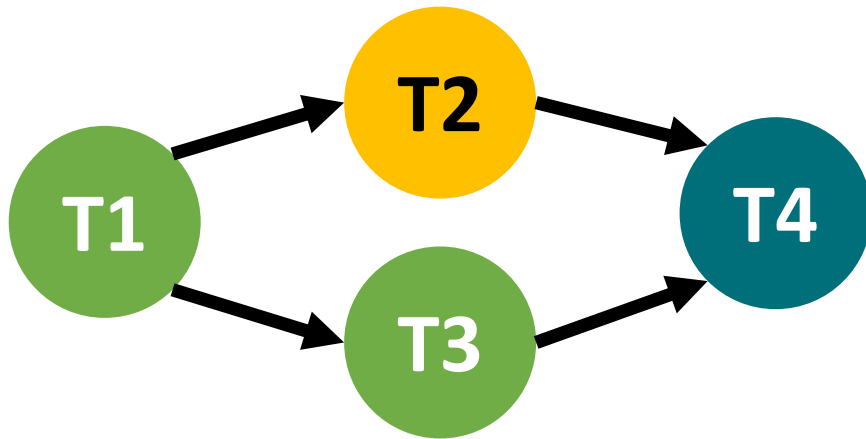
# Scheduling: Mapping b/w Tasks and Devices

- 4 tasks with 3 devices
  - $3^4$  (81) scheduling scenarios
  - Which one is the optimal?
    - High performance (HPC)
    - Low energy (edge computing)
    - Can users or IRIS automatically select the optimal one without 81 times of execution?
      - IRIS needs intelligent performance prediction & scheduling techniques.



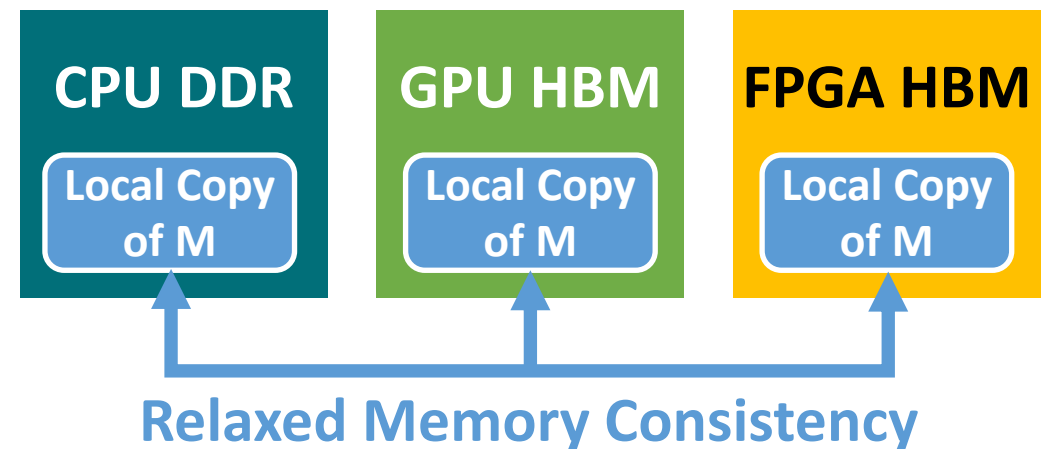
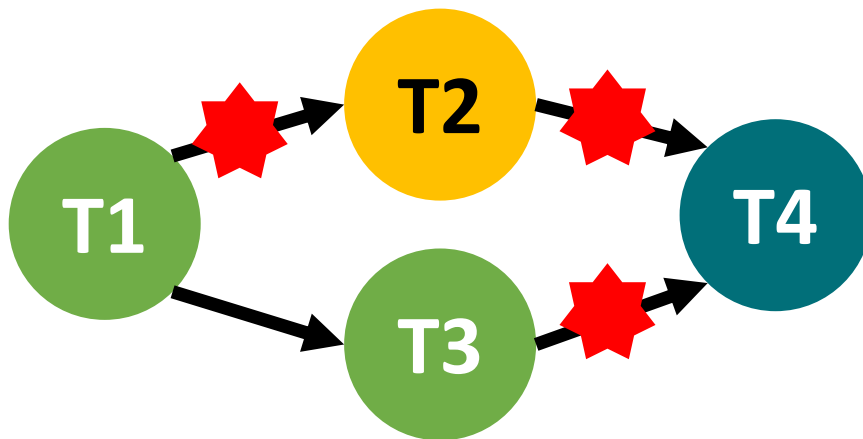
# Predicting Computation Time

- Nodes consume computation time (kernel execution commands).
- Predicting computation time is challenging.
  - Different device-specific optimized kernels & different HW architectures
  - Profiling-based scheduling policy
  - AI-based scheduling policy



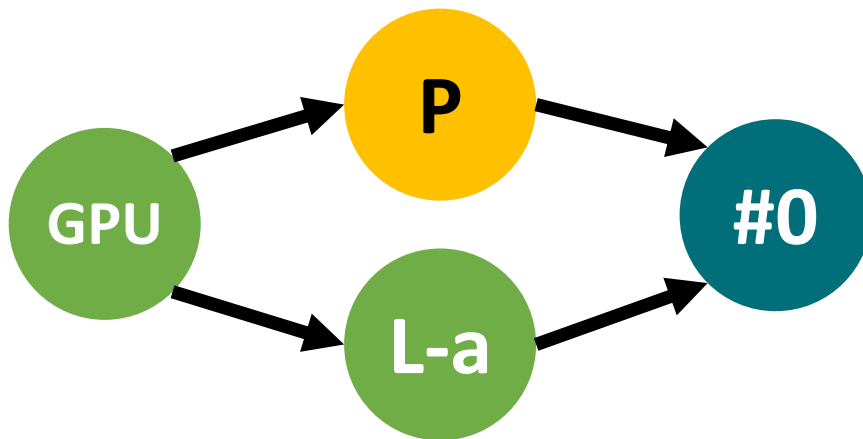
# Predicting Communication Time

- Nodes consume communication time (H2D & D2H memory copy commands)
- ★ Edges consume communication time (inter-device memory copy for memory consistency)
- Predicting communication time is easy ( $\text{data\_size} / \text{bandwidth}$ ).
  - Locality-aware scheduling policy



# Builtin Scheduling Policies

- IRIS provides seven builtin scheduling policies.
  - Device number or type
  - Locality-aware
  - Profiling-based
  - All, Any, Random



## Task Scheduler

### Scheduling Policies

*Device #/Type*

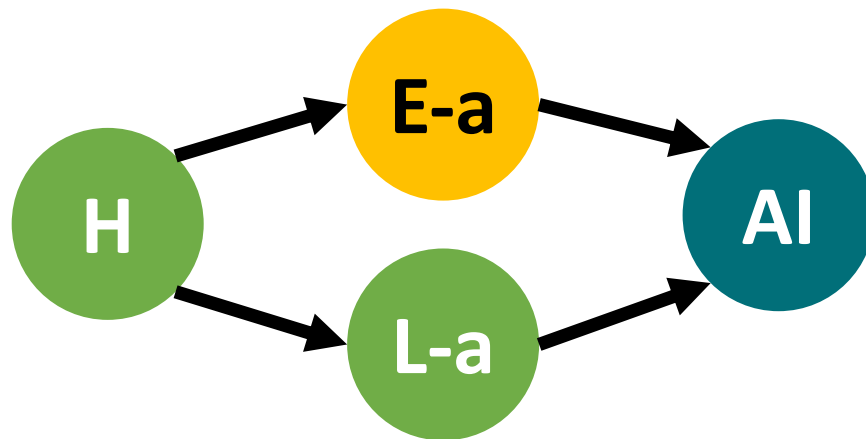
*Locality-aware*

*Profiling*

*All, Any, Rand*

# Custom Scheduling Policies Plugin

- Users can write their own scheduling policies and plug them in IRIS.
  - Out of the IRIS source tree
  - Specialized for their target applications and/or systems
  - Energy-aware, AI-based, many heuristics.



## Task Scheduler

### Scheduling Policies

*Device #/Type*

*Locality-aware*

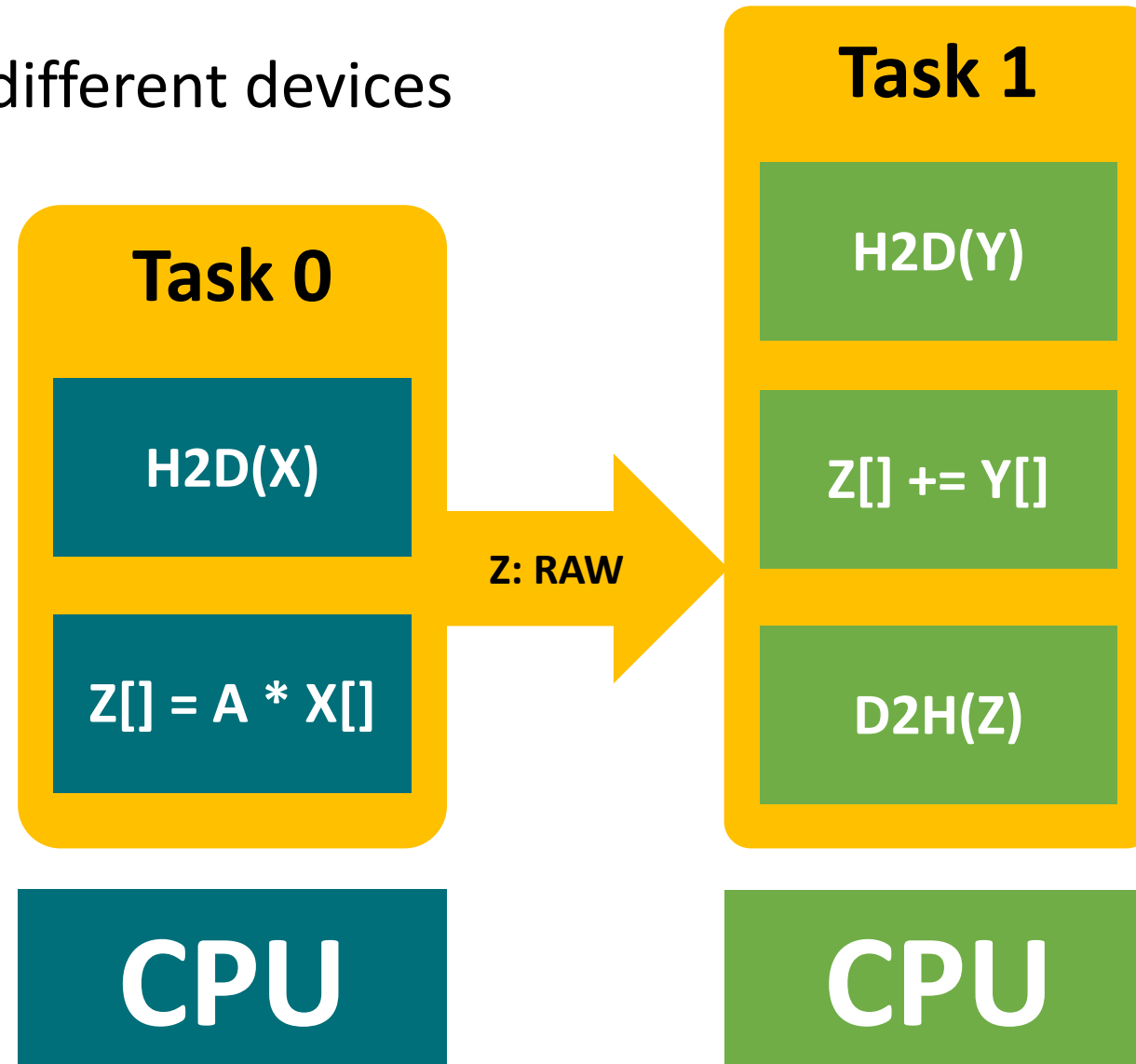
*Profiling*

*All, Any, Rand*

*User Customs*

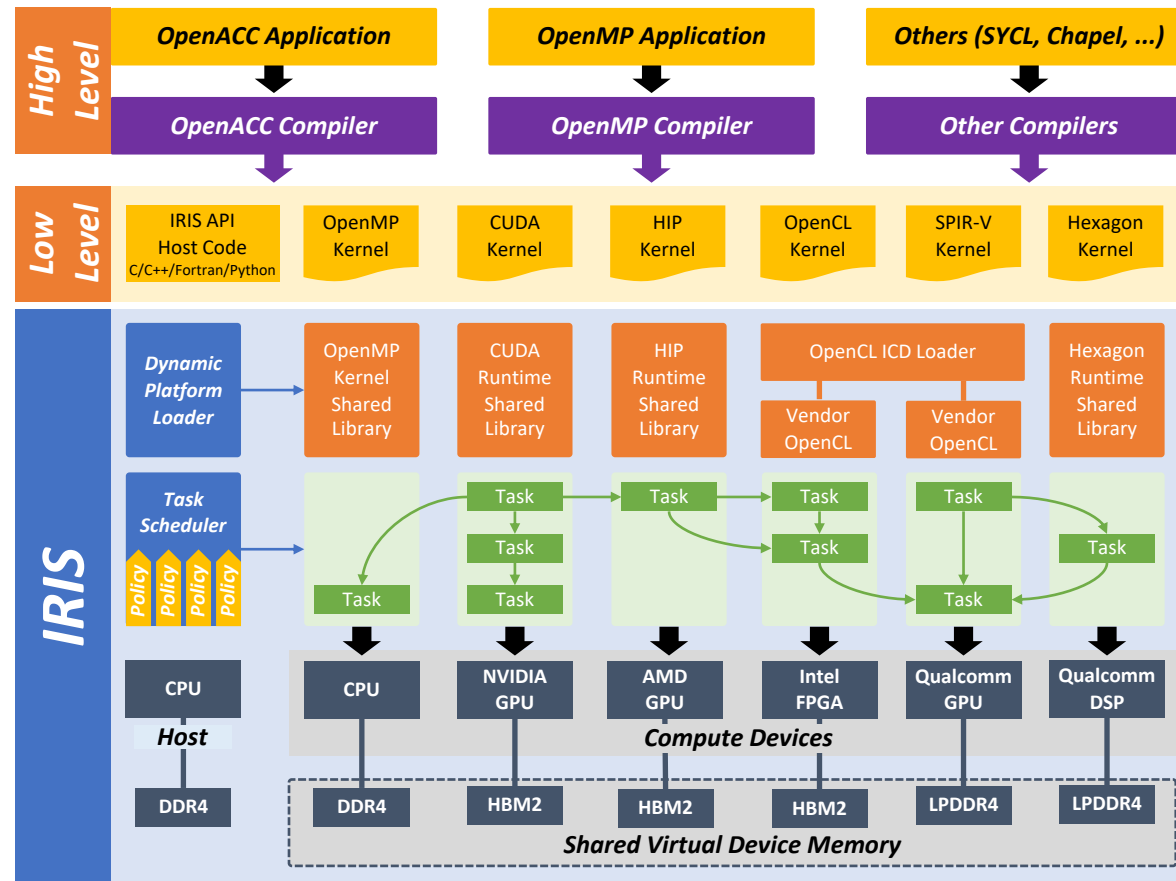
# Demo: SAXPY ( $Z[] = A * X[] + Y[]$ )

- Two tasks on two different devices



# Summary

IRIS is an Intelligent Task-Based Runtime System for Extremely Heterogeneous Architectures.



SYCL, Chapel, C++17

JIT Compilation, AI Workload: PyTorch, TensorFlow

AI-based Scheduling

TPU, DPU, Remote, Quantum

IRIS Overview

Future Work