alpaka Parallel Programming – Online Tutorial

Lecture 30 – Portability with alpaka

Lesson 36: Programming Heterogeneous Systems

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

www.casus.science

HZDR HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF
TECHNISCHE UNIVERSITÄT DRESDEN
CBG Max Planck Institute of Molecular Cell Biology and Genetics
UFZ HELMHOLTZ Centre for Environmental Research
Uniwersytet Wrocławski

SPONSORED BY THE
Federal Ministry of Education and Research
STAATSMINISTERIUM FÜR WISSENSCHAFT KULTUR UND TOURISMUS
Freistaat SACHSEN

Lesson 36: Programming Heterogeneous Systems

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

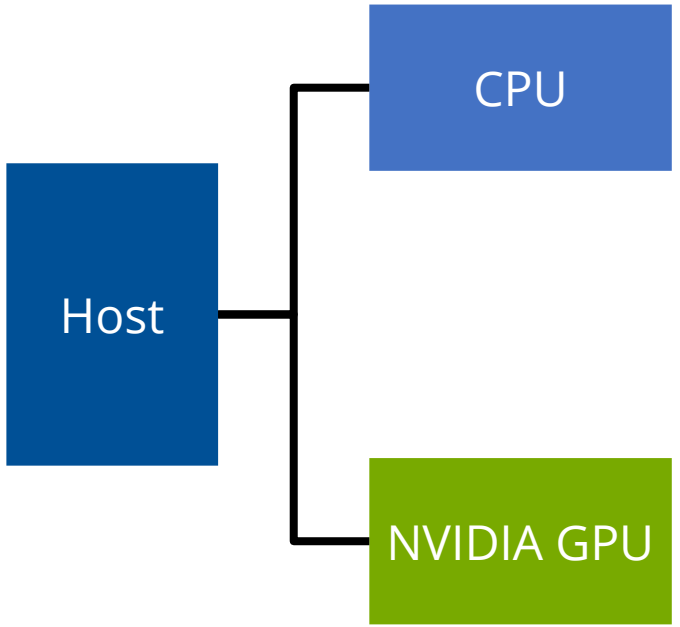
Recap

- Accelerator provides abstraction for a programming model and compatible physical devices
- Device represents a single physical device
- Queue enables communication between Host and a single Device
- Platform is a union of Accelerator, Device and Kernel
- Question: How to use multiple back-ends?

Lesson 36: Programming Heterogeneous Systems

Heterogeneous Systems

- Real-world scenario: Use all available compute power
- Also real-world scenario: Multiple different hardware types available
- Requirement: Usage of one back-end per hardware platform
- Requirement: Back-ends need to be interoperable



Lesson 36: Programming Heterogeneous Systems

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

Using multiple Platforms

- alpaka enables easy heterogeneous programming!
- Create one Accelerator per back-end
- Acquire at least one Device per Accelerator
- Create one Queue per Device

```
// Define Accelerators
using AccCpu = acc::AccCpuOmp2Blocks<Dim, Idx>;
using AccGpu = acc::AccGpuCudaRt<Dim, Idx>;

// Acquire Devices
auto devCpu = pltf::getDevByIdx<AccCpu>(0u);
auto devGpu = pltf::getDevByIdx<AccGpu>(0u);

// Create Queues
using QueueProperty = queue::NonBlocking;
using QueueCpu = queue::Queue<AccCpu, QueueProperty>;
using QueueGpu = queue::Queue<AccGpu, QueueProperty>;

auto queueCpu = QueueCpu{devCpu};
auto queueGpu = QueueGpu{devGpu};
```

Lesson 36: Programming Heterogeneous Systems

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

Communication

- Buffers are defined and created per Device
- Buffers can be copied between different Devices / Queues
- Not restricted to a single platform!
- **Restriction**: CPU to GPU copies (and vice versa) require GPU queue

```
// Allocate buffers
auto bufCpu = mem::buf::alloc<float, Idx>(devCpu, extent);
auto bufGpu = mem::buf::alloc<float, Idx>(devGpu, extent);

/* Initialization … */

// Copy buffer from CPU to GPU - destination comes first
mem::view::copy(gpuQueue, bufGpu, bufCpu, extent);

// Execute GPU kernel
queue::enqueue(gpuQueue, someKernelTask);

// Copy results back to CPU and wait for completion
mem::view::copy(gpuQueue, bufCpu, bufGpu, extent);

// Wait for GPU, then execute CPU kernel
wait::wait(cpuQueue, gpuQueue);
queue::enqueue(cpuQueue, anotherKernelTask);
```

Lesson 36: Programming Heterogeneous Systems

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

Heterogeneous programming with alpaka

- alpaka gives you access to all of your system's computation resources
- alpaka eases programming for different device types
- alpaka enables simple data transfers between different devices
- alpaka makes your code reusable
- alpaka makes your code portable

Run once, scale everywhere!

CASUS

CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

www.casus.science

HZDR HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF

TECHNISCHE UNIVERSITÄT DRESDEN

CBG Max Planck Institute of Molecular Cell Biology and Genetics

UFZ HELMHOLTZ Centre for Environmental Research

Uniwersytet Wrocławski

SPONSORED BY THE

Federal Ministry of Education and Research

STAATSMINISTERIUM FÜR WISSENSCHAFT KULTUR UND TOURISMUS

Freistaat SACHSEN