

# alpaka Parallel Programming – Online Tutorial

## Lecture 30 – Portability with alpaka

### Lesson 35: The Platform Concept



**CASUS**

CENTER FOR ADVANCED  
SYSTEMS UNDERSTANDING

[www.casus.science](http://www.casus.science)



# Lesson 35: The Platform Concept

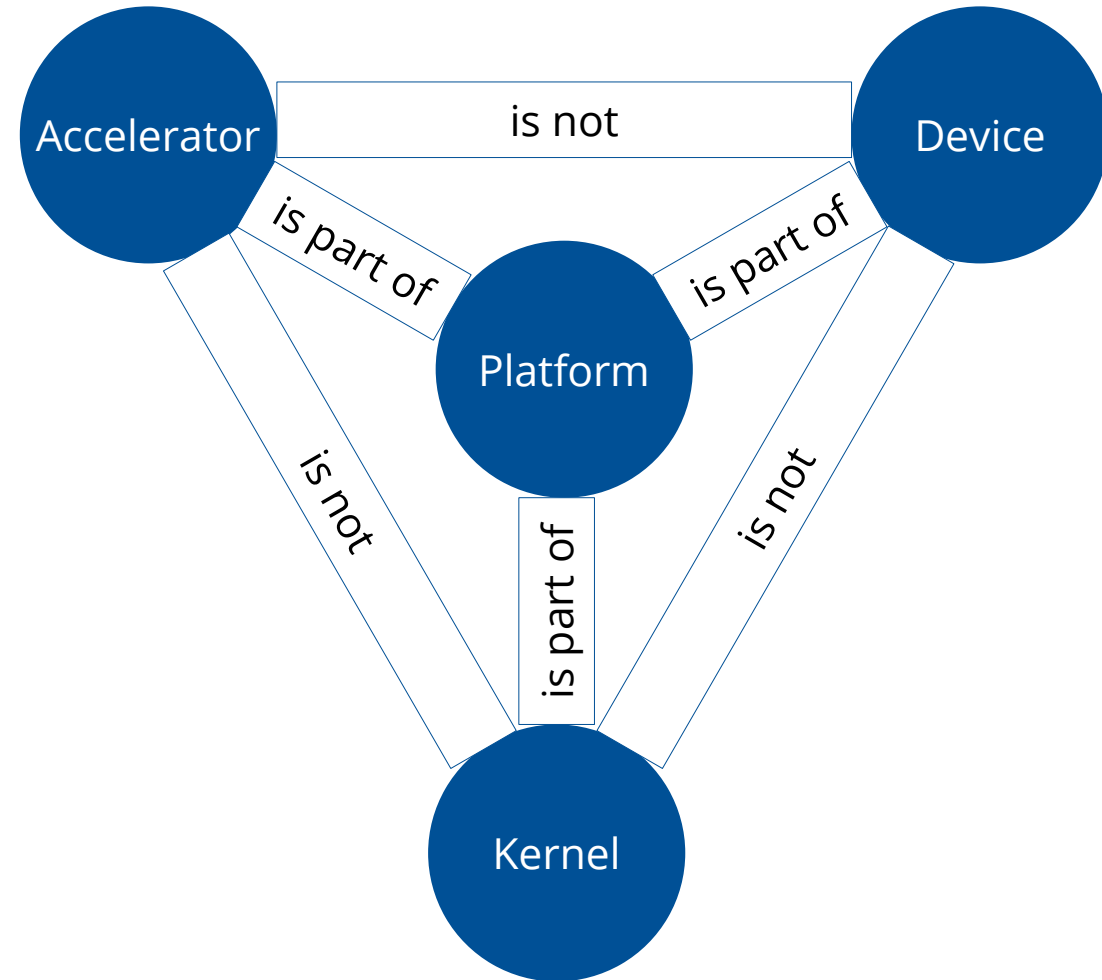
## Recap

- Accelerator provides abstract view of all capable physical devices
- Device represents a single physical device
- Queue enables communication between the host and a single Device
- Question: How is portability between back-ends achieved?

# Lesson 35: The Platform Concept

## alpaka Platform

- Platform is meta-concept in alpaka
- Union of Accelerator, Device and Kernel functionality
  - Accelerator gives structure to the host side and functionality to the device side
  - Device gives functionality to the host side
  - Kernels are agnostic of Device details  
→ Portable Kernels



# Lesson 35: The Platform Concept

## Changing the target platform

```
using namespace alpaka;

using Dim = dim::DimInt<1u>;
using Idx = std::size_t;

/** BEFORE */
using Acc = acc::AccCpuOmp2Blocks<Dim, Idx>;

/** AFTER */
using Acc = acc::AccGpuHipRt<Dim, Idx>;

/* No change required - dependent types and variables are automatically changed */
auto myDev = pltf::getDevByIdx<Acc>(0u);

using Queue = queue::Queue<Acc, queue::NonBlocking>;
auto myQueue = Queue{myDev};
```

# Lesson 35: The Platform Concept

## What alpaka does for you

- During configuration with CMake:
  - Default behaviour: Enables all suitable back-ends for your system
  - Behaviour is configurable with CMake variables
  - CMake handles back-end dependencies
- After changing the Accelerator:
  - Back-end switched automatically
  - All Queue operations will be executed on associated devices

# Lesson 35: The Platform Concept

## What you have to do for alpaka

- Device side: Make no assumptions about your hardware!
  - Program your Kernels as abstract and portably as possible
  - Use the Accelerator for device-side operations
  - Kernels are instantiated for a specific platform at compile-time
  - This is what the Accelerator template parameter is for!

```
template <typename Acc>  
ALPAKA_FN_ACC void operator()(Acc const & acc, /* ... */) const;
```

- Host side: Know your hardware!
  - Use Devices for management of physical devices
  - Adapt the work division (Blocks per Grid, Threads per Block, elements per Thread) to your hardware and problem size



# CASUS

CENTER FOR ADVANCED  
SYSTEMS UNDERSTANDING

[www.casus.science](http://www.casus.science)