

alpaka Parallel Programming – Online Tutorial

Lecture 00 – Getting Started with alpaka

Lesson 02: Portable Heterogeneous Parallel Programming



CASUS

CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

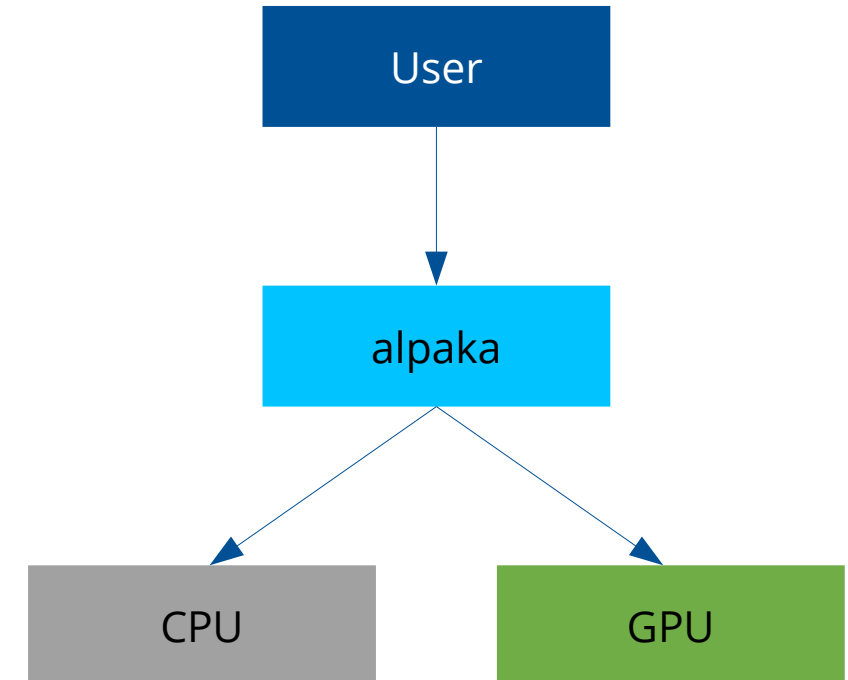
www.casus.science



Lesson 02: Portable Heterogeneous Parallel Programming

alpaka enables portability!

- Idea: Write algorithms once...
 - ... independently of target architecture
 - ... independently of available programming models
- Decision on target platform made during compilation
 - Choosing another platform just requires another compilation pass
- alpaka defines an abstract programming model
- alpaka utilizes C++14 to support many architectures
 - CUDA, HIP, OpenMP, TBB, ...



Lesson 02: Portable Heterogeneous Parallel Programming

alpaka enables full utilization of heterogeneous systems!

- Algorithms are generally independent of chosen target architecture

```
auto const taskCpu = alpaka::kernel::createTaskKernel<AccCpu>(workDivCpu, kernel, ...);  
auto const taskGpu = alpaka::kernel::createTaskKernel<AccGpu>(workDivGpu, kernel, ...);
```

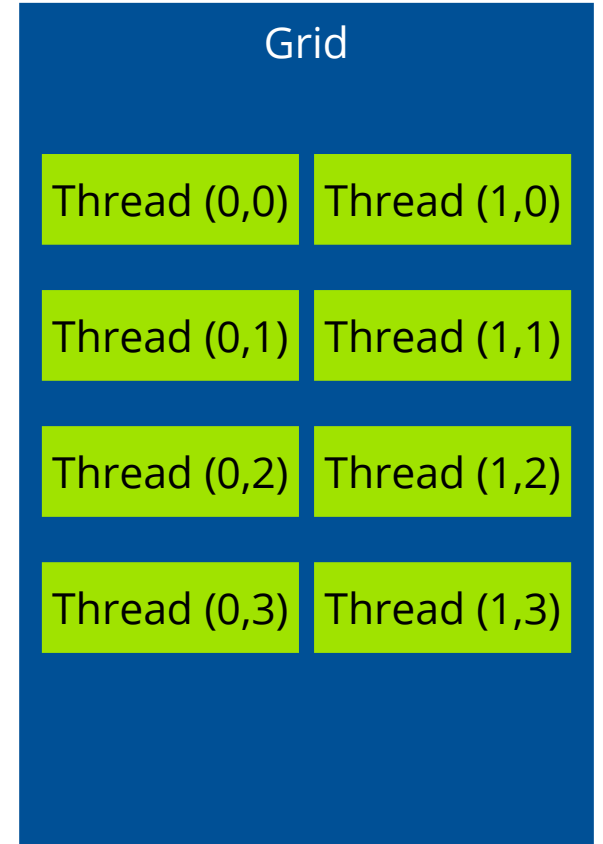
- Optimization for specific architecture is still possible

```
template <typename TAcc> // general case  
void someComputationIntensiveFunction(TAcc const & acc) { ... };  
  
template <> // specialization for AccGpu  
void someComputationIntensiveFunction<AccGpu>(AccGpu const & acc) { ... };
```

Lesson 02: Portable Heterogeneous Parallel Programming

How parallelism is achieved, Part I: The grid, a digital frontier

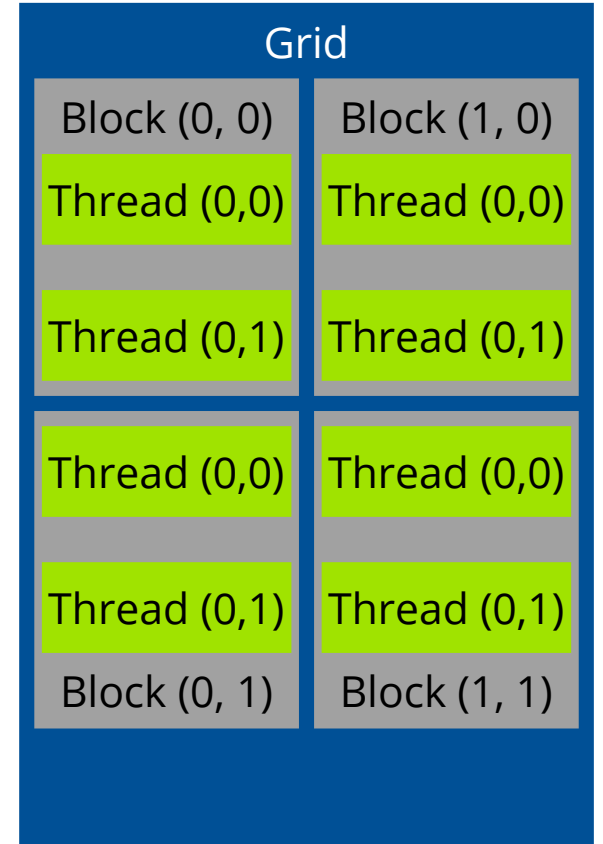
- alpaka is ideal for data-parallel algorithms
 - execute the same algorithm on different data elements
- alpaka **kernel**: sequence of commands forming the algorithm on a per-element level
- alpaka **thread**: execution of a kernel for a single data element
- threads are executed in parallel and are independent of each other
- alpaka **grid**: n-dimensional grid of all **threads** executing a specific kernel
 - each thread is assigned a unique index on the grid
 - threads on the grid are able to communicate through high-latency **global memory**



Lesson 02: Portable Heterogeneous Parallel Programming

How parallelism is achieved, Part II: Blocks on the grid

- Grids are divided into independent **blocks** of equal size
- Each thread is assigned to exactly one block
- Each thread is assigned an unique index on the block
- All threads inside a block are executed in parallel
- All threads inside a single block can be **synchronized**
→ no synchronization on the grid level!
- All threads inside a block can communicate through low-latency **shared memory**



Lesson 02: Portable Heterogeneous Parallel Programming



Summary

- alpaka is ideal for data-parallel algorithms
- Algorithms are written per data element (**kernel**)
- data parallelism achieved through a hierarchy of independent **threads** and **blocks** on a **grid**
- All threads can communicate through high-latency **global memory**
- Threads inside a block can be **synchronized**
- Threads inside a block can communicate through low-latency **shared memory**



CASUS

CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

www.casus.science