Lecture 10 – The alpaka Programming Model

Lesson 14: alpaka Kernels
Lesson 14: alpaka Kernels

What is a Kernel?

- Contains the algorithm
- Written on per-data-element basis
- alpaka Kernels are functors (function-like C++ structs / classes)
- \texttt{operator()} is annotated with \texttt{ALPAKA_FN_ACC} specifier
- \texttt{operator()} must return \texttt{void}
- \texttt{operator()} must be \texttt{const}

```cpp
struct HelloWorldKernel {
    template <typename Acc>
    ALPAKA_FN_ACC void operator()(Acc const & acc) const {
        using namespace alpaka;
        uint32_t threadIdx = idx::getIdx<Grid, Threads>(acc)[0];
        printf("Hello, World from alpaka thread %u\n", threadIdx);
    }
};
```
Lesson 14: alpaka Kernels

Threads and Kernels

- A Kernel is executed by a number of Threads
- Threads are executing the same algorithm for different data elements

- A Kernel **defines** an algorithm
- A Thread **applies** an algorithm

```c
struct myKernel {
    /* … */
};
```
Lesson 14: alpaka Kernels

Scheduling

- Threads are mapped to cores
- Many more Threads than cores → Thread scheduling required
- **Thread order is unspecified!**
  → Programmer cannot control the order of data element processing
- Hardware specifics need to be taken into account
Lesson 14: alpaka Kernels

Example: Thread mapping on CPUs

- CPU consists of multiple cores
  - Because of simultaneous multithreading there can be more logical than physical cores!
- alpaka Threads are executed by CPU cores
Lesson 14: alpaka Kernels

Example: Thread mapping on GPUs

- GPU consists of streaming multiprocessors (SMs)
- Each SM consists of multiple cores
- alpaka Threads are executed by individual SM cores