alpaka Parallel Programming – Online Tutorial

Lecture 10 – The alpaka Programming Model

Lesson 15: The Problem Size

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 15: The Problem Size

Problem size and hardware capabilities

- The programmer's questions:
  - How large is the problem? (= How many data elements need processing?)
  - Which capabilities are offered by the hardware? (= How many cores are available?)
- The programmer's challenge:
  - Problem size and number of cores completely disjoint
  - How to distribute the former amongst the latter?

Lesson 15: The Problem Size

How to choose the number of alpaka Threads

- The two important factors:
  - Problem size → number of data elements
  - Hardware capabilities → number of cores
- Rule of thumb: One Thread per data element
  - Not always ideal (depending on algorithm)
  - Chance for optimisation

Lesson 15: The Problem Size

Choosing the number of Threads

- (Usually) you have more Threads than cores
- In alpaka, the overall number of Threads is `blocksPerGrid * threadsPerBlock`
  - We will introduce Thread Blocks in a later lecture!

```
using Idx = uint32_t;

Idx blocksPerGrid = 8;
Idx threadsPerBlock = 1;
```

Lesson 15: The Problem Size

Beware!

- Don’t run too many Threads in parallel!
  - An exact definition of “too many” depends on your hardware.
- Some hardware resources are always shared between Threads
- Having too many Threads accessing shared resources results in bottlenecks
  - Can seriously impact your program’s performance
  - Chance for optimisation

Lesson 15: The Problem Size

Example: I/O buffer

- All Threads call `printf`
- The access to the output buffer needs to be serialized
- More Threads
  → more serialization
  → worse performance

```
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);
}
```

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