alpaka Parallel Programming – Online Tutorial

Lecture 20 – Thread Parallelism in alpaka

Lesson 25: Computing π – Part III

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 25: Computing π – Part III

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

Recap

- Introduced parameter passing
- Introduced grid dimensionality
- Introduced memory access
- Introduced mathematical functions
- Now: Memory management

Lesson 25: Computing π – Part III

Kernel requirements

- alpaka kernels accept pointers to Device memory
- Challenge: Host and Device don’t always share memory
- Memory buffers need to be allocated on both the Host and the Device
- Memory needs to be transferred from the Host to the Device and vice versa
- In case of CPU Devices there is optimisation potential in avoiding unnecessary copies!

Lesson 25: Computing π – Part III

Allocating memory on the Host

- Memory can be allocated using `alpaka::mem::buf::alloc()`

```
using Host = /* … */;                                          // not important now
using BufHost = mem::buf::Buf<Host, float, Dim, Idx>;          // Host buffer type
using Vec = vec::Vec<Dim, Idx>;                                // Vector type

auto const devHost = alpaka::pltf::getDevByIdx<Host>(0u);      // create host device
Vec const extents(n);                                          // create extents
BufHost hostBuffer = mem::buf::alloc<float, Idx>(devHost, extents);
```

- Pre-allocated memory can be used with alpaka:

```
std::vector<float> plainBuffer(n);
using ViewHost = mem::view::ViewPlainPtr<Host, float, Dim, Idx>;
ViewHost hostViewPlainPtr(plainBuffer.data(), devHost, Vec(plainBuffer.size()));
```

Lesson 25: Computing π – Part III

Allocating memory on the Device

- Allocating memory on the Device works the same way!
- Memory can be allocated using `alpaka::mem::buf::alloc()`

```
using Acc = /* … */;                                          // not important now
using BufAcc = mem::buf::Buf<Acc, float, Dim, std::size_t>;   // Accelerator buffer type

auto const devAcc = pltf::getDevByIdx<Acc>(0u);               // create accelerator dev.

BufAcc accBuffer = mem::buf::alloc<float, std::size_t>(devAcc, extents);
```

Lesson 25: Computing π – Part III

Memory transfers

- After initializing the Host buffer (`for` loop, `<algorithm>`, `memset`, ...) memory can be transferred
- In alpaka all memory operations are explicit
- Use `alpaka::mem::view::copy()` to initiate transfers:

```
mem::view::copy(devQueue,           // queue (explained later)
                devBuffer,          // copy target
                hostBuffer,         // copy source
                extents);           // number of elements
```

```
mem::view::copy(devQueue,
                devBuffer,
                hostViewPlainPtr,   // for pre-allocated memory
                extents);
```

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