alpaka Parallel Programming – Online Tutorial

Lecture 30 – Portability with alpaka

Lesson 31: Changing the Accelerator

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

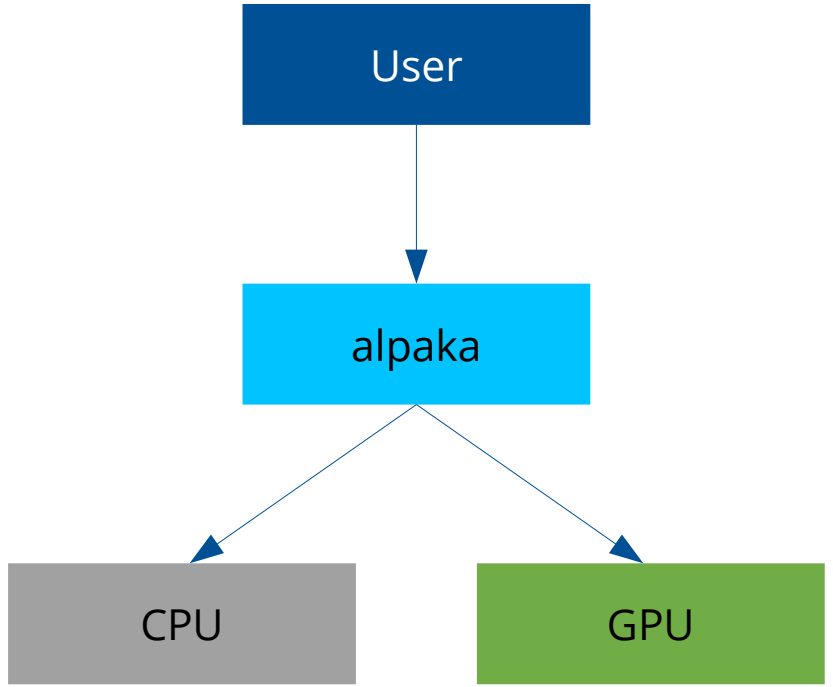
SPONSORED BY THE
Federal Ministry of Education and Research
STAATSMINISTERIUM FÜR WISSENSCHAFT KULTUR UND TOURISMUS
Freistaat SACHSEN

Lesson 31: Changing the Accelerator

Moving from CPU to GPU

alpaka allows for easy ...

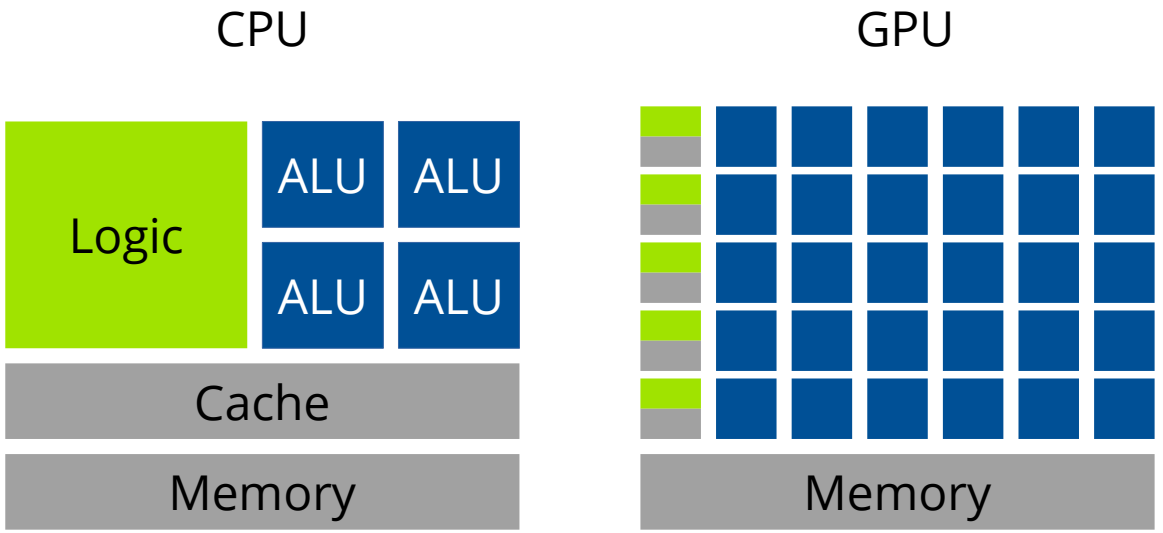
- ... exchange of the accelerator
- ... porting of programs across accelerators
- ... experimentation with different devices
- ... mixing of accelerator types



Lesson 31: Changing the Accelerator

Architectural differences

- Rule of thumb: Offload computationally intensive parts to GPUs
- GPUs are designed for high throughput
  - Many lightweight threads
  - High memory latency
- CPUs are designed for low latency
  - Few heavyweight threads
  - Low memory latency



Source: Pradeep Gupta, *CUDA Refresher: Reviewing the Origins of GPU Computing*. https://developer.nvidia.com/blog/cuda-refresher-reviewing-the-origins-of-gpu-computing/. Access date: 25 June 2020

Lesson 31: Changing the Accelerator

Switching the Accelerator

- alpaka provides a number of pre-defined Accelerators in the `acc` namespace.
- For GPUs:
  - `AccGpuCudaRt` for NVIDIA GPUs
  - `AccGpuHipRt` for AMD and NVIDIA GPUs
- For CPUs
  - `AccCpuFibers` based on Boost.fiber
  - `AccCpuOmp2Blocks` based on OpenMP 2.x
  - `AccCpuOmp4` based on OpenMP 4.x
  - `AccCpuTbbBlocks` based on TBB
  - `AccCpuThreads` based on `std::thread`

```
// Example: CPU accelerator
using Acc = acc::AccCpuOmp2Blocks<Dim, Idx>;

// Example: CUDA GPU accelerator
using Acc = acc::AccGpuCudaRt<Dim, Idx>;

// Example: HIP GPU accelerator
using Acc = acc::AccGpuHipRt<Dim, Idx>;
```

Lesson 31: Changing the Accelerator

Changing the work division

- GPUs have many more cores than CPUs
  → More parallel threads possible
- GPUs have several multiprocessors
- Each multiprocessor can execute multiple threads
- Threads are grouped into blocks
- Blocks are scheduled to run on multiprocessors

```
// CPU work division (example)
Idx blocksPerGrid     = 8;
Idx threadsPerBlock   = 1;
Idx elementsPerThread = 1;

// GPU work division (example)
Idx blocksPerGrid     = 64;
Idx threadsPerBlock   = 512;
Idx elementsPerThread = 1;
```

Lesson 31: Changing the Accelerator

GPU performance hints

- Avoid divergent `if`-`else`-blocks
  - GPU threads are organized into groups (NVIDIA: *warp*, AMD: *wavefront*)
  - Groups are executed in lock step
    → If there is divergence, all threads execute the `if` block first and the `else` block next
- GPU threads are much more lightweight than CPU threads
  - Context switch is much cheaper on GPUs
  - Spawn many more threads than you have GPU cores
    → Hide memory latency behind computation

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