

alpaka Parallel Programming – Online Tutorial

Lecture 20 – Thread Parallelism in alpaka

Lesson 22: 2D Work Division



CASUS

CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

www.casus.science



Lesson 22: 2D Work Division

From 1D to 2D

- n -dimensional grids work in a similar way to 1D grids
 - `idx::getIdx<Grid, Threads>(acc)` returns a vector containing n indices
 - `idx::getIdx<Grid, Threads>(acc)[dim]` returns an integer
- **Beware:** In a 2D grid, y is dimension zero and x is dimension one
 - `idx::getIdx<Grid, Threads>(acc)` returns a vector containing 2 indices: the y -index at position `0` and the x -index at position `1`
 - `idx::getIdx<Grid, Threads>(acc)[0]` returns the y -index

Computing the 2D index

- 2D `gridThreadId` can be computed manually, too
- Can be done per vector:

```
using Vec = vec::Vec<dim::DimInt<2>, uint32_t>;  
Vec gridThreadId = gridBlockIdx * blockDimExtent + blockDimIdx;
```

- Or per index:

```
uint32_t gridThreadIdY = gridBlockIdxY * blockDimExtentY + blockDimIdxY;  
uint32_t gridThreadIdX = gridBlockIdxX * blockDimExtentX + blockDimIdxX;
```

Lesson 22: 2D Work Division

Preparing the Host for 2D

- Open the `helloWorld` example again
- Go the top of `main()` and enable 2D dimensionality on the Host:

```
using Dim = dim::DimInt<2>;
```

- Further down in `main()`, set up a 2D Thread hierarchy:

```
auto blocksPerGrid = vec::Vec<Dim, Idx>{2u, 4u};  
auto threadsPerBlock = vec::Vec<Dim, Idx>{1u, 1u};  
auto elementsPerThread = vec::Vec<Dim, Idx>{1u, 1u};
```

Obtaining the index

- Change the Kernel as shown on the right side
- Switch to your `build` directory and rebuild:

```
cmake --build . \  
--config Release
```
- Execute the example again

```
// Use these lines for obtaining the indices:  
uint32_t gridThreadIdY = idx::getIdx<Grid, Threads>(acc)[0];  
uint32_t gridThreadIdX = idx::getIdx<Grid, Threads>(acc)[1];  
  
printf("Hello, World from alpaka thread (%u, %u)!\n",  
      gridThreadIdY, gridThreadIdX);
```

Obtaining the index

- 2D blocks work the same way!
- Change the kernel again
- Switch to your `build` directory and rebuild:
`cmake --build . \`
`--config Release`
- Execute the example

```
// Use these lines for obtaining the indices:  
using Vec = vec::Vec<dim::DimInt<2>, uint32_t>;  
Vec gridBlockIdx = idx::getIdx<Grid, Blocks>(acc);  
Vec blockThreadIdx = idx::getIdx<Block, Threads>(acc);  
  
printf("Hello, World from thread (%u, %u) in block (%u, %u)!\n",  
       blockThreadIdx[0], blockThreadIdx[1],  
       gridBlockIdx[0], gridBlockIdx[1]);
```

Lesson 22: 2D Work Division

Summary

- n -dimensional grids are very similar to 1D grids
- Pitfall: Reversed index ordering
- n -dimensional indices and extents can be obtained through API calls or by computing them
- n -dimensional blocks work the same way



CASUS

CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

www.casus.science