

alpaka Parallel Programming – Online Tutorial

Lecture 30 – Portability with alpaka

Lesson 34: The Queue Concept



CASUS

CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

www.casus.science



Lesson 34: The Queue Concept

Recap

- An alpaka Accelerator provides an abstract view of all physical devices
- An alpaka Device is a representation of exactly one concrete physical device
- Question: How are Kernels and memory operations offloaded to individual devices?

Lesson 34: The Queue Concept

Connecting Host and Device

- alpaka Queues enable communication between Host and Device
- Two queue types: blocking and non-blocking
- Blocking queues block the Host until Device-side command returns
- Non-blocking queues return control to Host immediately, Device-side command runs asynchronously

```
// Choose queue behaviour - Blocking or NonBlocking
using QueueProperty = queue::NonBlocking;

// Define queue type
using Queue = queue::Queue<Acc, QueueProperty>;

// Create queue for communication with myDev
auto myQueue = Queue{myDev};
```

Lesson 34: The Queue Concept

Queue operations

- Queues execute Tasks (see next slide):

```
queue::enqueue(myQueue, taskRunKernel);
```

- Check for completion:

```
bool done = queue::empty(myQueue);
```

- Wait for completion, Events (see next slide), or other Queues:

```
wait::wait(myQueue); // blocks caller until all operations have completed
```

```
wait::wait(myQueue, myEvent); // blocks myQueue until myEvent has been reached
```

```
wait::wait(myQueue, otherQueue); // blocks myQueue until otherQueue's ops have completed
```

Lesson 34: The Queue Concept

Tasks and Events

- Device-side operations (kernels, memory operations) are called Tasks
- Tasks on the same queue are executed in order (FIFO principle)

```
queue::enqueue(queueA, task1);  
queue::enqueue(queueA, task2); // task2 starts after task1 has finished
```

- Order of tasks in different queues is unspecified

```
queue::enqueue(queueA, task1);  
queue::enqueue(queueB, task2); // task2 starts before, after or in parallel to task1
```

- For easier synchronization, alpaka Events can be inserted before, after or between Tasks:

```
auto myEvent = event::Event<Queue>(myDev);  
queue::enqueue(queueA, myEvent);  
wait::wait(queueB, myEvent); // queueB will only resume after queueA reached myEvent
```

Lesson 34: The Queue Concept

Setting up Accelerator, Device and Queue

```
// Choose types for dimensionality and indices
using Dim = dim::DimInt<1>;
using Idx = std::size_t;

// Choose the back-end
using Acc = acc::AccGpuHipRt<Dim, Idx>;

// Obtain first device in the HIP GPU list
auto myDev = pltf::getDevByIdx<Acc>(0u);

// Create non-blocking queue for chosen device
using Queue = queue::Queue<Acc, queue::NonBlocking>;
auto myQueue = Queue{myDev};

// Done! We can now enqueue device-side operations.
```



CASUS

CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

www.casus.science