Parallel Computing with GPUs

CUDA Streams
Part 1 – Synchronous and Asynchronous Execution

The University Of Sheffield.

Dr Paul Richmond

http://paulrichmond.shef.ac.uk/teaching/COM4521/

This Lecture (learning objectives)

- Synchronous and Asynchronous Execution
  - Classify synchronous and asynchronous execution
  - Demonstrate examples of synchronous execution with CUDA
  - Demonstrate examples of asynchronous execution with CUDA

Blocking and Non-Blocking Functions

- Synchronous vs Asynchronous
  - Synchronous:
    - Blocking call
    - Executed sequentially
  - Asynchronous:
    - Non-Blocking call
    - Control returns to host thread
- Asynchronous Advantages
  - Overlap execution and data movement on different devices
    - Not just GPU and CPU
    - Also consider disk or network (low latency)

Asynchronous Behaviour so far…

- CPU pipeline
  - Programmer writes code considering it to be synchronous operations
  - Compiler generates overlapping instructions to maximise pipe utilisation
  - Same end result as non overlapping instructions (hopefully)
- CPU threading
  - Similar threads execute asynchronously on different multiprocessors
  - Requires careful consideration of race conditions
  - OpenMP gives us critical sections etc. to help with this
- CUDA Warp execution
  - Threads in the same warp execute instructions synchronously
  - Warps on a SMP are interleaved and executed asynchronously
  - Careful use of `__syncthreads()` to ensure no race conditions

CUDA Host and Device

- Most CUDA Host functions are synchronous (blocking)

- Exceptions (synchronous with the host)
  - Kernel calls
  - `cudaMemcpy` within a device (`cudaMemcpyDeviceToDevice`)
  - `cudaMemcpy` host to device of less than 64kB
  - *Asynchronous memory copies and streams… (this lecture)*

- Asynchronous functions will block when
  - `deviceSynchronize`() is called
  - A new kernel must be launched (implicit synchronisation)
  - Memory must be copied to or from the device (implicit synchronisation)

Asynchronous Execution

```
//copy data to device
cudaMemcpy(d_a, a, size * sizeof(int), cudaMemcpyHostToDevice);
cudaMemcpy(d_b, b, size * sizeof(int), cudaMemcpyHostToDevice);

//execute kernels on device
kernelA<<<blocks, threads>>>(d_a, d_b);
kernelB<<<blocks, threads>>>(d_b, d_c);

//copy back result data
cudaMemcpy(c, d_c, size * sizeof(int), cudaMemcpyDeviceToHost);
```

Is there any Asynchronous Execution?

Asynchronous Execution

```
//copy data to device
cudaMemcpy(d_a, a, size * sizeof(int), cudaMemcpyHostToDevice);
cudaMemcpy(d_b, b, size * sizeof(int), cudaMemcpyHostToDevice);

//execute kernels on device
kernelA<<<blocks, threads>>>(d_a, d_b);
kernelB<<<blocks, threads>>>(d_b, d_c);

//copy back result data
cudaMemcpy(c, d_c, size * sizeof(int), cudaMemcpyDeviceToHost);
```

Completely Synchronous

time

Asynchronous Execution

```
//copy data to device
cudaMemcpy(dev_a, a, size * sizeof(int), cudaMemcpyHostToDevice);
cudaMemcpy(dev_b, b, size * sizeof(int), cudaMemcpyHostToDevice);

//execute kernel on device
addKernel<<<blocks, threads>>>(dev_c, dev_a, dev_b);

//host execution
myCPUFunction();

//copy back result data
cudaMemcpy(c, dev_c, size * sizeof(int), cudaMemcpyDeviceToHost);
```

Is there any Asynchronous Execution?

Asynchronous Execution

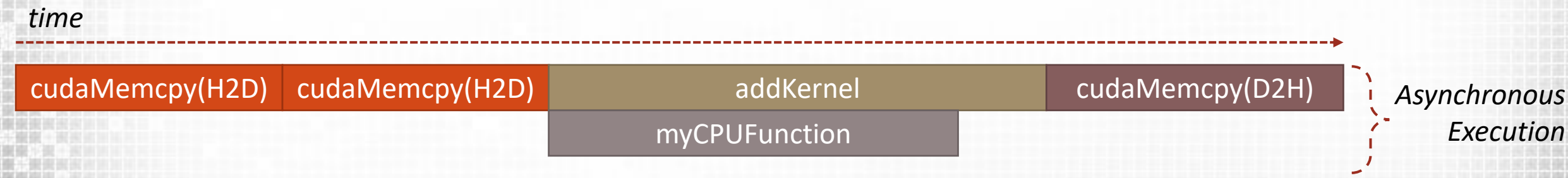
```
//copy data to device
cudaMemcpy(dev_a, a, size * sizeof(int), cudaMemcpyHostToDevice);
cudaMemcpy(dev_b, b, size * sizeof(int), cudaMemcpyHostToDevice);

//execute kernel on device
addKernel<<<blocks, threads>>>(dev_c, dev_a, dev_b);

//host execution
myCPUFunction();

//copy back result data
cudaMemcpy(c, dev_c, size * sizeof(int), cudaMemcpyDeviceToHost);
```

Asynchronous GPU and CPU Execution



Summary

- Synchronous and Asynchronous Execution
  - Classify synchronous and asynchronous execution
  - Demonstrate examples of synchronous execution with CUDA
  - Demonstrate examples of asynchronous execution with CUDA

- Next Lecture: CUDA Streams