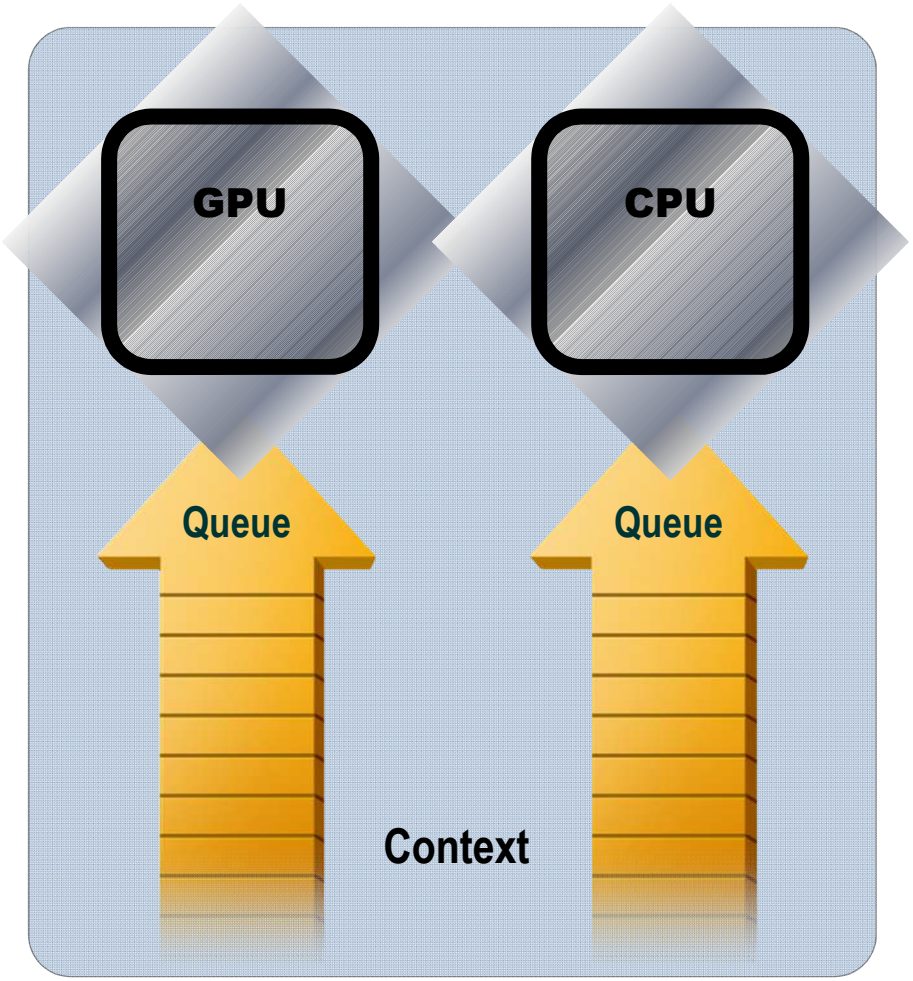
Lecture 8

HETEROGENEOUS COMPUTING WITH OPENCL

Running on the CPU and GPU

- Kernels can be run on multiple devices at the same time
- We can exploit many GPUs and the host CPU for computation
- Simply define a context with multiple platforms, devices and queues
- We can even synchronize between queues using Events (see appendix)
- Can have more than one context



Running on the CPU and GPU

1. Discover all your platforms and devices
   - Look at the API for finding out Platform and Device IDs

2. Set up the cl::Context with a vector of devices

```
cl::Context(const VECTOR_CLASS<Device> &devices,
            cl_context_properties *properties = NULL,
            void (CL_CALLBACK *pfn_notify)(
              const char *errorinfo,
              const void *private_info_size,
              ::size_t cb, void *user_data) = NULL,
            void *user_data = NULL, cl_int *err = NULL);
```

3. Create a Command Queue for each of these devices
   - C examples in the NVIDIA (oclSimpleMultiGPU) and AMD (SimpleMultiDevice) OpenCL SDKs

The steps are the same in C and Python, just the API calls differ as usual

Exercise 10: Heterogeneous Computing

- Goal:
  - To experiment with running kernels on multiple devices
- Procedure:
  - Take one of your OpenCL programs
  - Investigate the Context constructors to include more than one device
  - Modify the program to run a kernel on multiple devices, each with different input data
  - Split your problem across multiple devices if you have time
  - Use the examples from the SDKs to help you
- Expected output:
  - Output the results from both devices and see which runs faster