Instructions

In this exercise, we'll scale a vector (array) of single precision numbers by a scalar. You'll learn how to allocate memory on the GPU and transfer data to and from the GPU.

Data transfer with unified memory

Take a look at the file `scale_vector_um.cu`. It contains a number of todos. To compile and run this exercise you need to load CUDA: `module load cuda/`.

For this exercise, you'll use `cudaMallocManaged` to allocate memory:

```
cudaMallocManaged(T** devPtr, size_t size, unsigned int flags)
```

Like most CUDA functions, `cudaMallocManaged` returns `cudaError_t`.

See the source for more todos. For compilation, use

```
make
```

To run your code, call `bsub` with the correct parameters. A shortcut is given via

```
make run
```

Data transfer without unified memory

Unified memory requires a chip with capability 3.5 (Kepler and up). Older GPUs can't be used this way. See `scale_vector.cu` for things that need to be done and the slides for the API. In case you want to try out this version, compile it with:

```
make scale_vector
```

Submit to the batch system similarly as shown in above's `make run` invocation.