GPUs at UCD

Two servers with GPUs. Login with:

- `ssh username@lipschitz.ucdavis.edu`
- `ssh username@pearson.ucdavis.edu`

Background and History

- GPUs are specialized units for rendering high definition graphics quickly and smoothly.
- Designed to do many simple calculations in parallel.
- In recent years there has been a great deal of progress in using GPUs for calculations, not just graphics.
- CUDA (extension of C and C++) is the forefront language for programming GPUs. Other main language is OpenCL.

About CUDA

- CUDA is a bunch of C/C++ libraries. Low-level language.
- For NVIDIA GPUs only.
- Other higher level languages exist, PyCUDA, RCUDA.

Types of Parallelism

Two main types of parallelism:

1. Task Parallelism: Simultaneously running different tasks that do not depend on other completed tasks.
2. Data parallelism: Perform the same task on multiple pieces of data.

Some Definitions

- Kernel: GPU pogram that runs on a thread grid
- Thread Hierarchy:
  - Grid: a set of blocks
  - Block: a set of warps
  - Warp: a SIMD group of 32 threads
  - Grid size * block size = total # of threads

  Up to the programmer to determine the grid/block structure. Can have a large impact on efficiency.

Terminology

- Host: The CPU
- Device: The GPU
- Kernel: Function that runs on the device.
- Thread: Think of a series of calculations/operations
- Kernels are typically executed by lots of threads
- One kernel is executed at a time
- Threads are cheap to launch on GPUs
- Gains in efficiency come with using large number of threads to perform calculations in parallel.

Basics of CUDA

- Memory management: GPU memory must be allocated, initialized, and freed.
- Data transfer: Data required by GPU is copied from host to device.
- Kernel Launch: The kernel is launched with specified grid/block configuration.
- Result transfer: If needed, the results must be copied back from CUDA device to host.