GPGPU Computing

• Computation + Rendering on the Same Card

source: https://www.youtube.com/watch?v=53Z3VHGQjE8
GPU V.S CPU

Throughput

source: cuda programming guide
Memory Bandwidth

source: cuda programming guide
Applications accelerated by GPUs

- Bioinformatics
- Computational Finance
- Computational Fluid Dynamics
- Molecular Dynamics
- Imaging & Computer Vision
- Weather and Climate
GPU Competitors

- **NVIDIA**
  G80, Fermi, and Kepler Series

- **AMD**
  heterogeneous CPU/GPU = APU
  firestream, fusion, berlin, seattle chips

- **Intel**
  Larabee → intel phi (MIC)
GPU Programming Languages/libraries/models

- **Compute Unified Device Architecture (CUDA) - NVIDIA**
- Open Computing Language (OpenCL)
- C++ AMP - from Microsoft
- BrookGPU
- OpenGL / DirectX
- Stream (Brook+) - AMD
- ....
Programming with CUDA

An Example in C

Add vector A and vector B to vector C.

```c
void add_vector (float *A, float *B, float *C, in N){
  for ( index = 0; index < N; index++ )

void main ( ) { ...
  add_vector (A1,B1,C1,N1);
  ...
```
Programming with CUDA

An Example

Add vector A and vector B to vector C.

```c
__global__ void add_vector (float *A, float *B, float *C, in N){
    ...
}

void main ( ) {
    add_vector <<<dimGrid, dimBlock>>> (A1,B1,C1,N1);
    ...
}
```
Thread View

- A kernel is executed as a grid of thread blocks
- All threads share off-chip memory space
- A thread block is a batch of threads that can cooperate with each other by:
  - Synchronizing their execution
  - Efficiently sharing data through a low latency shared memory
  - Two threads from two different blocks cannot cooperate
• Threads in a block is organized into warps (32 threads/warp on Tesla C2075)

• A (half) warp is a SIMD thread group.
SIMD (Single Instruction Multiple Data)

\[ C[\text{index}] = A[\text{index}] + B[\text{index}] \]

- thread 0: \( C[0] = A[0] + B[0] \)
- ...  

Execute at the same time.
SIMD (Single Instruction Multiple Data)

\[ C[index] = A[index] + B[index]; \]

- thread 0: \[ C[0] = A[0] + B[0]; \]
- ....

Execute at the same time.

SIMT (Single Instruction Multiple Thread)

Simultaneous execution of many SIMD thread groups
Thread Life Cycle in HW

- Thread blocks are serially distributed to all the streaming multi-processors (SM)
  - potentially >1 thread block per SM
- SM schedules and executes warps that are ready
- As a thread block completes, resources are freed
  - more thread blocks will be distributed to SMs
GPU Architecture

Nvidia Tesla C2075

- 14 streaming multiprocessors (SM) -- 32 cores each @ 1.15 GHz

- Each SM has 48 KB shared memory, 32768 registers
Blocks on SM

- Threads are assigned to SMs in Block granularity
  - Limited number of blocks per SM as resource allows
    - SM in C2075 can host 1536 threads: 256 (threads/block) * 6 blocks Or 128 (threads/block) * 12 blocks, etc.

- Threads run concurrently

- SM assigns/maintains thread id #s

- SM manages/schedules thread execution
Thread Scheduling/Execution

- Warps are scheduling units in SM

- If 3 blocks are assigned to an SM and each block has 256 threads, how many warps are there in an SM?

- At any point in time, only one of those Warps will be selected for instruction fetch and execution on the SM.
SM Warp Scheduling

- SM hardware implements zero-overhead Warp scheduling
  - Warps whose next instruction has its operands ready for consumption are eligible for execution
  - Eligible Warps are selected for execution on a prioritized scheduling (round-robin/age) policy
  - All threads in a Warp execute the same instruction when selected
- 4 clock cycles needed to dispatch the same instruction for all threads in a Warp
- If one global memory access is needed for every 4 instructions
  - A minimal of $\text{?}$ Warps are needed to fully tolerate 200-cycle memory latency
CUDA Demo

The matrix/vector add example

/ilab/users/zz124/cs671_2013/samples/0_Simple/vectorAdd
21 GPU Machines at ilab (recently installed)

**GT 630**: 2GB memory, 192 CUDA cores

GPU Machines at my lab

**GTX 680** (1536 CUDA cores), **Tesla C2075** (448 CUDA cores), **Quadro 4000** (Fermi), **Geforce GT 640** (kepler)

Future deployment (hopefully)

**GPU cluster with NVIDIA Tesla K20c** (2496 CUDA cores, peak 4 Tflops) and **Quadro K5000**
Next Class

- Fundamental parallel algorithms on GPUs
  for seemingly sequential algorithms