Where to teach CUDA?

Special topics course

- Varying levels possible
- Popular with students
- Lots of projects possible

Parallel programming course

- “Advanced topic” in parallel course
  - Commonly used in HPC
  - Good introduction to heterogeneity
- Re-emphasize concepts of data parallelism and locality

Computer Organization

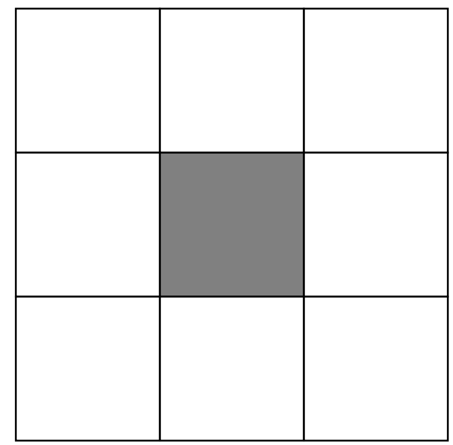
- Again "advanced topic"
- Tie to idea of general story of hardware/software interaction
  - data parallelism (vector instructions)
  - introduction to system heterogeneity
  - memory levels and access times

Two approaches

- Game of Life application
  - visually noticeable speedup
  - students get to play with significant code
- Concept-oriented lab
  - short code segments to illustrate specific features
  - possible with limited background

Game of Life

- Break simulation region into cells and time into steps



- Cells live or die based on neighbors
  - Living cells die unless 2 or 3 are alive
  - Dead cells become alive if 3 are alive

Conceptual unit goals

- Idea of parallelism
- Benefits and costs of system heterogeneity
- Data movement and NUMA
- Generally, the effect of architecture on program performance

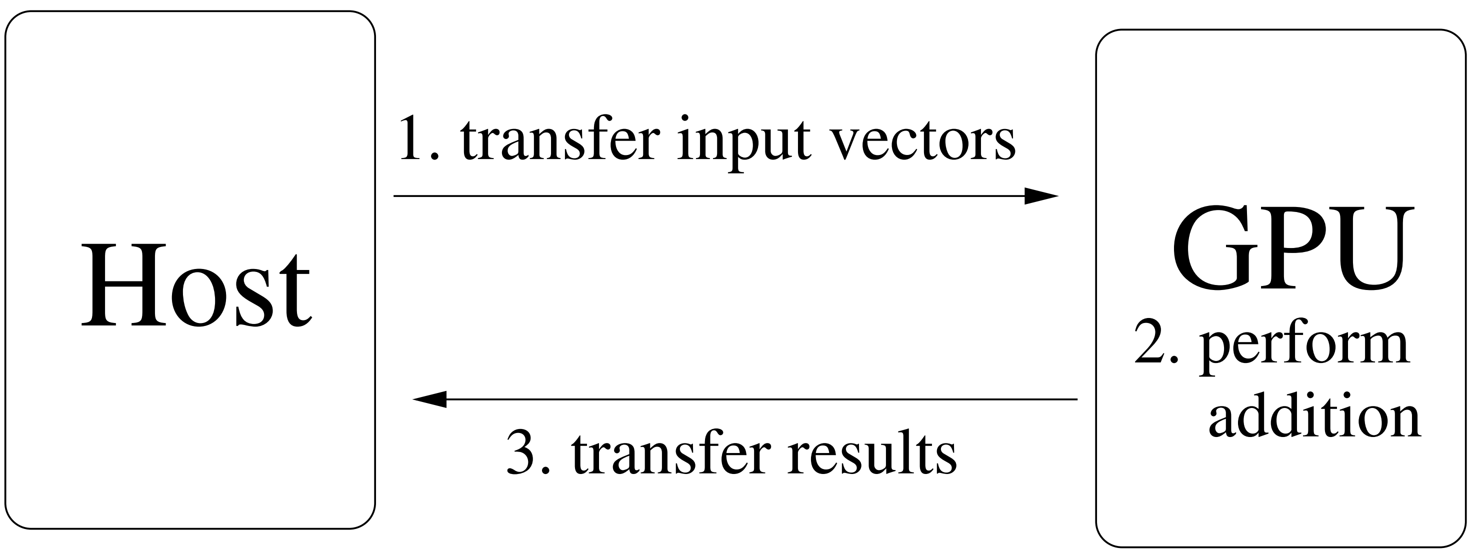
My constraints

- Brief time: Course has lots of other goals
  - One 70-minute lab and parts of 2 lectures
- Relatively inexperienced students
  - Some just out of CS 2
  - Many didn't know C or Unix programming

Conceptual exercises

- Data transfer time via vector addition
- Thread divergence via "bucketing"
- Constant memory via image generation

Data transfer time



Thread divergence

- Threads organized into warps of 32 threads
- All threads in a warp execute the same instruction

Thread divergence

- Threads organized into warps of 32 threads
- All threads in a warp execute the same instruction

```
if(condition) {
    statement1;
} else {
    statement2;
}
```

Thread divergence

- Threads organized into warps of 32 threads
- All threads in a warp execute the same instruction

all threads ➔

```
if(condition) {
    statement1;
} else {
    statement2;
}
```

Thread divergence

- Threads organized into warps of 32 threads
- All threads in a warp execute the same instruction

```
if(condition) {
    statement1;      <- some threads
} else {
    statement2;      <- some threads
}
```

```
void kernel_1(int* a) {
    int cell = threadIdx.x % 32;
    a[cell]++];
}
```

```
void kernel_2(int* a) {
    int cell = threadIndx.x %32;
    switch(cell) {
        case 0: a[0]++; break;
        case 1: a[1]++; break;
        case 2: a[2]++; break;
        case 3: a[3]++; break;
        default: a[cell]++;
    }
}
```

Constant memory

- Different but essentially the same calls
- Not allowed to change it

Constant memory

- Different but essentially the same calls
- Not allowed to change it

- Allows GPU to cache values
- Values are broadcast to half-warps

Constant memory

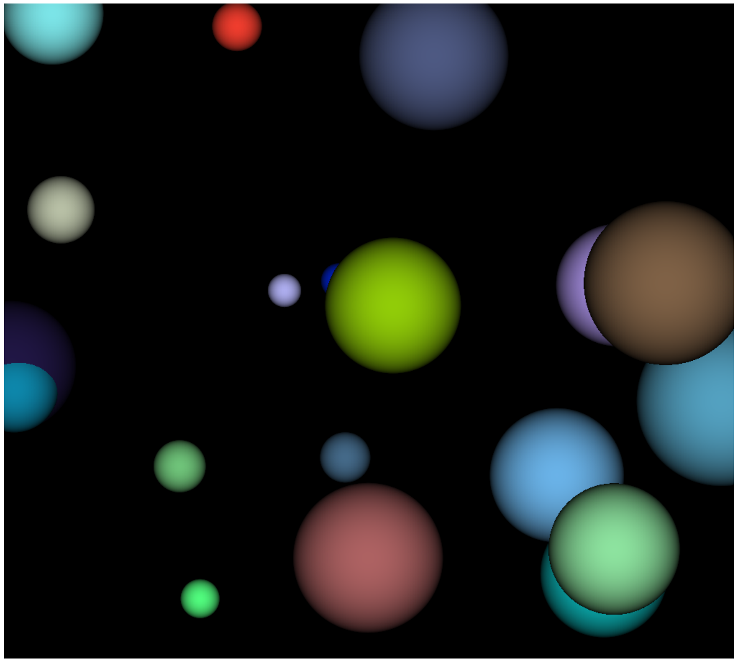
- Different but essentially the same calls
- Not allowed to change it

- Allows GPU to cache values
- Values are broadcast to half-warps
  - Serializes requests if different threads in a half-warp request different memory addresses

“Ray tracing” application

[“CUDA by Example” by Sanders and Kandrot]

- Each pixel traverses sphere array to find closest intersection
- Accesses to array all in same order



Key part of kernel

```
for(int i=0; i<SPHERES; i++) {

    float t = s[i].hit(x, y, &n);
    if (t > maxz) {
        //set color to sphere i
        maxz = t;
    }
}
```

Key part of kernel

```
for(int i=0; i<SPHERES; i++) {

    float t = s[i].hit(x, y, &n);
    if (t > maxz) {
        //set color to sphere i
        maxz = t;
    }
}
```

```
for(int j=0; j<SPHERES; j++) {
    i = (j+threadIdx.x) % SPHERES;
    float t = s[i].hit(x, y, &n);
    if (t > maxz) {
        //set color to sphere i
        maxz = t;
    }
}
```

Using conceptual exercises

- Introductory lecture
  - GPUs: massively parallel, outside CPU, kernels, SIMD
- Lab illustrating features of CUDA architecture
  - Data transfer time
  - Thread divergence
  - Memory types
- "Lessons learned" lecture
  - Reiterate architecture
  - Demonstrate speedup with Game of Life
  - Talk about use in Top 500 systems