Agenda for Today

1. Bug of the Day
2. Lab Questions
3. New Assignment
4. Distributed Systems
5. Networks Exercise
Bug of the Day
What's wrong here?

... 
rc = cudaMemcpy(gpu_x, cpu_x, sizeof(float)*N, cudaMemcpyHostToDevice);
if (rc != cudaSuccess) { ... }

kernel<<<blocks, threads_per_block>>>(gpu_x, gpu_y);
rc = cudaDeviceSynchronize();
if (rc != cudaSuccess) { ... }

rc = cudaMemcpy(cpu_y, gpu_y, sizeof(float)*N, cudaMemcpyHostToDevice);
if (rc != cudaSuccess) { ... }

... 
this should be device \rightarrow host!
Q&A
How are we supposed to use __synthreads_count in task #3?

You’ve seen __synthreads before. This function causes all the threads in a block to wait until the entire block has reached that point. The __syncthreads_count function adds functionality to that basic model. Each thread passes a parameter to __syncthreads_count when it blocks. When all the threads unblock, __syncthreads_count returns number of threads that passed a non-zero parameter.

At a higher level, if each thread passes a predicate to __syncthreads_count, then the __syncthreads_count function returns to all threads the number of true predicates. By “predicate” I just mean that you evaluate a boolean expression and pass the result to __syncthreads_count.
Can we use `cudaMallocManaged` instead of manual copying?
Yes, you can. You’ll end up replacing some provided code, but not much. Ignore the comment that says not to modify other things.

How does the `dim3` type work with kernel invocation?
When you launch a kernel like this: `kernel<<<4, 9>>>(...)` you get four blocks of nine threads each. Within a block, you’ll have threads with values of `threadIdx.x` from 0 through 8.

If you launch a kernel like this: `kernel<<<4, DIM2(3, 3)>>>(...)` you also get four blocks of nine threads each. Within each block, you’ll have threads with indices `threadIdx.x` from 0 through 2, and for each of those x values, a thread will have `threadIdx.y` from 0 through 2.
How do we convert between 1D and 2D indices? *(row-major form)*

If we have a 1D coordinate \(i\) into the board (value is 0 to 80, inclusive), we can convert it to 2D coordinates like this:

\[
\begin{align*}
\text{int} \ row &= \frac{i}{9}; & \text{// } i \text{ must be an integer for this to work} \\
\text{int} \ col &= i \mod 9;
\end{align*}
\]

You can convert back from 2D coordinates to 1D coordinates:

\[
\text{int} \ i = \text{row} \times 9 + \text{col};
\]
Sudoku Lab Q&A
New Assignment
(released later today)
Distributed Systems
Distributed Systems Basics

What is a distributed system?
A group of computers that share resources. Resources can be storage or state, computing power, etc.

Why do we use distributed systems?
If we want computers to work together to solve a problem, we have a distributed system. Why do we want computers to work together? Using a distributed system gives us access to more resources, either more varied, or just more capacity. Some problems require more computing power, storage, memory, etc. than you can possibly fit into one computer, even a very expensive or large one.
Distributed Systems Basics

How do machines in a distributed system communicate?

Participants in a distributed system communicate via messages, and sometimes acknowledgements of those messages.
In a distributed system, messages are delivered using a network.

How is that mechanism different from the way threads communicate?

Threads communicate using shared memory and synchronization, whereas distributed systems send messages.
Synchronization tools like condition variables are a little like messages, but they’re just an indication that another thread should view some shared state now (or later).
Messages in a distributed system are different because no state is shared automatically.
Distributed Systems Basics

What makes building distributed systems difficult?

Sharing information is important, but some information shouldn’t be shared widely.

**Issues:** information flow policies, security

Unreliability! Networks could fail to deliver messages. Computers that participate in a distributed system can fail too. Distributed systems have to plan for failure.

**Issues:** redundancy, coping with failure

We have to limit how much our system trusts others. Distributed systems are usually accessible on the internet.

**Issues:** security, trust

Data movement becomes a significant part of the work in a distributed system.
Network Programming Exercise