SOFT354: List of Revision Topics

What The Class Test Won't Require

- You don't need to learn the specifications for the various API functions that we've been using in SOFT354 (cudaMemcpy, MPI_Bcast, etc etc.) If a question needs you to use one of these functions, the function definition will be provided.
- You won't have to write any code longer than a single line. I won't deduce marks for trivial mistakes in code, like missing semi-colons.
- You don't need to remember how to do linear algebra options (e.g. matrix-vector multiplication, matrix-matrix multiplication). If these are on the test, the algorithm will be given.
- You definitely don't need to know the specifications for the different compute capability levels (e.g. how much shared memory a CC2.0 GPU has). Any questions that need you to use these values will include the relevant part of the table.

Lecture 1

- The importance of parallel programming.
- Basically what a GPU is, what a streaming multiprocessor is.

Lecture 2

- The difference between host and device memory.
- The difference between static and dynamic memory allocation (on host and device).
- The different types of memory in CUDA (Registers, Caches, Shared Memory (see also Lecture 3), Global Memory, etc.), including their advantages/disadvantages and how they're used.
- What pinned memory is and how it can be useful.

Lecture 3

- How threads are organised into grids, blocks and warps, and how these are assigned to streaming multiprocessors.
- How to calculate the maximum block occupancy of a kernel, for a GPU with a particular compute capability.
- The general principle of coalesced memory access, and how code can be written to take advantage of this.

Lecture 4

- How a GPU deals with branching in kernels, and why this can be a performance problem.
- The __syncthreads() function, and why it can be a problem with divergence.
- How threads can co-operate to load data into shared memory.
- What the CGMA ratio is, why it's important (see also lecture 2 slide 4), and how to calculate it for a given kernel.

Lecture 5

- What convolution is, and its applications.
- How shared memory can be used to optimize calculating a convolution.

Lecture 6

- The difference between task and data parallelism.
- What SISD, SIMD, MIMD, MISD, SPMD and MPMD are.
- The difference between a shared memory and distributed (or hybrid) memory system.

Lecture 7

- Sublinear, super-linear and linear speed-up.
- What speed-up and efficiency are, and how to calculate them given some values. You may also be required to rearrange the equations, for example to calculate the parallel run time given the serial runtime and speed-up.
- Amdahl's and Gustafson's laws: the equations and what the laws mean.
- The difference between threads and processes.
- MPI_Send and MPI_Recv: what they do, and what it means to say they're "blocking" and "nonovertaking".

Lecture 8

- The meaning of network diameter, bisection width, bisection bandwidth, valency and link count, and how these affect the performance / cost / practicality of a network.
- You may be given a diagram of a network and asked to find one of the measures above (e.g. bisection width). However, you don't need to know the general formula for each measure for each network type.
- What the different MPI collective communication functions (Bcast, Reduce, Allreduce, Gather, Allgather) do, although you don't need to memorise the actual function definitions (i.e. what parameters they take and in what order).
- Why collective communication functions can be more efficient than just sending all messages from / to a single process.

Lecture 9

- You should understand in general how a torus or fat-tree network is built. Again, you don't need to learn the formulae for calculating diameter, bisection width, etc. for a general network of size N.