CS 5220 – 2015-09-15 Preclass Questions

Michael Whittaker (mjw297)

September 21, 2015

0. I spent roughly an hour on the preclass assignment partly before class and partly after class.

1. As a computer science major, I have very little exposure to differential equations of any variety. Since the lumped parameter and distributed parameter simulations are motivated by differential equations, it's very hard to understand what they are. Also, in the section on passing particles, we discuss that we can optimize code by having processors pass around memory. This is clear, but implementing such a thing seems unclear.

2. Each Intel Xeon E5-2620 processor has 15MB of cache. The naive implementation of the game of life requires two $n \times n$ boards where each entry of a board is a single byte. This means we can fit a board of size $n = \sqrt{\frac{15000000}{2}} \approx 2738$.

   After running and timing a simulation with a 100×100 board for 1000 generation and a simulation with a 4000×4000 board for 100 generations, I found the former took 0.357 seconds and the latter took 55.577 seconds. This is a cell per second rate of 28011204 and 2878887 respectively.

3. Assume we have an $n \times n$ board and want to partition the board into 4 equal sized quadrants. Naively, each processor would get exactly $\frac{1}{4}$ of the board. However, in order to advance a single generation, the processor need to exchange the values of the cells on the boundary of the quadrants. Instead, we can assign overlapping quadrants where each processor gets slightly more than $\frac{1}{4}$ of the board. This would allow each processor to advance multiple generations without communication.

4. In order to parallelize the code, each processor would get a partition of the $n \times n$ grid. They would locally update their grid and synchronize every once in a while. I think this would lead to considerable speedup on the totient cluster because the game of life is rather pleasingly parallel.

5. Yes, for each particle in the processor's local memory, it must access every single particle. This requires it to fetch memory from all processors. It repeats this process for every particle which is very inefficient. It would be better to do a circular passing strategy, as discussed in the slides.