October 8th, 2015 Pre-Class Questions

Elliot Cartee

Question 0

How much time did you spend on this pre-class exercise, and when?

I started these pre-class exercises the night of October 7th. I spent about an hour on them (including reading the slides and listening to the narration).

Question 1

What are one or two points that you found least clear in the 10/08 slide decks (including the narration)?

I still don't really understand the bullet point about wormhole and cut-through routing.

Question 2

I remember at the beginning of the course you said something about wanting to "flip the classroom" for this course. My understanding was that generally meant that watching/listening to lectures happened mostly outside of the class period, and other activities happened during the class period. But I feel like for a lot of this class, the class periods have kind of turned out mostly like normal lectures. This isn't necessarily a bad thing in itself, but I feel like this class hasn't really lived up to the whole "flipping the classroom" thing quite how I was expecting.

Question 3

(3a)

In `ring.c`, all the particles are divided into batches for each processor. Each processor has one main batch that it is responsible for. So each processor computes the interactions between its local data, and another set of particles at a time. Then each processor sends these other particles to the next processor through the `curr_buf` buffer, while receiving another set of particles from the previous processor in the `recv_buf` buffer. This is repeated

until each processor has computed all the interactions between the local data and all other particles.

(3b)

For nonblocking communication, I would make it so that there are now three buffers, `curr_buf`, `recv_buf` and `send_buf`.

Now instead of using the `MPI_Sendrecv` command, I would copy the data from `curr_buf` (which I just computed the interactions with) to `send_buf`. Then each processor would send the data in `send_buf` to the data in `recv_buf` in the next processor in the ring. And each processor would also receive the data via the `MPI_Isend` and `MPI_Irecv` commands. Then each processor would wait using the `MPI_Wait` command until the messages are done sending, and then move on to computing the next set of interactions.