Yuan Huang(Yh638)

Preclass 10/08/2015

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

10/6 at noon, 2hours

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

The last slide of 2015-10-08-model, I don't quite understand what problem that slide is discussing about.

2. Now that we are now basically a third of the way into the semester, and are (mostly) settled into the steady pace of things, I would appreciate your feedback on what is working well or poorly about the class. Comments on things I can reasonably change are particularly useful -- venting about the cluster, for example, is understandable but doesn't help me that much in adjusting!

Could you put the link to pre-class questions on the schedule page (http://cornell-cs5220-f15.github.io/schedule.html)?

Because I think sometime people forget to check the git and just check the course page. (Especially at the beginning of the semester when students haven't got used to the things)

3. The ring demo implements the protocol described in the particle systems slide deck from 9/15:

http://cornell-cs5220-f15.github.io/slides/2015-09-15-particle.html#/11

a) In your own words, describe what ring.c is doing.

It first forms a ring and cut it into *size* parts, each part contains *nper* numbers. Each part initializes their data and start to change data in a loop. That is each time the i^{th} part send its current data to the $i+1^{th}$ part(in mod size sense) and receive data from the $i-1^{th}$ part of the ring and do the calculation. The calculation for a certain place is

$r_{jnew} = r_{jold} + \sum_{k=0}^{nper-1} | x_j - x_k |$,where x_j is the j^{th} number in current data.

So it is basically calculating $r_j = \sum_{k=0,k\neq j}^{all\ things} | x_j - x_k |$, x_j is the j^{th} element(particles) for all the particles. It takes p round and each round do n_{per}^2 calculation(n_{per} particles and for each particles we calculate n_{per} other aprticles).

b) How might you modify the code to have the same computational pattern, but using non-blocking communication rather than MPI_Sendrecv? Note that according to the MPI standard, one isn't supposed to read from a buffer that is being handled by a non-blocking send, so it is probably necessary to use three temporary buffers rather than the current two.

We add a buffer called send_buf, and set an variable request

At each loop:

1) we try to receive data and swap recv_buf and curr_buf
2) we copy the curr_buff into send buf
3) We initialize the send and receive
4) Then we do the computing on curr_buf
5) we wait until sending finishes

Basically the code should be

```
tmp = curr_buf;
curr_buf = recv_buf;
recv_buf = tmp;
memcpy(send_buf, curr_buf);

MPI_Isend(send_buf, …, requests[0]);
MPI_Irecv(recv_buf, …, requests[1]);

do computing on curr_buf

MPI_Wait(requests, status);
//end of the loop
```