HPC Hw5 (Zhe Chen)

Machine configuration

Results are obtained by running on CIMS's machines Crackle{1-5}.cims.nyu.edu. Configurations of these servers can be found on https://cims.nyu.edu/webapps/content/systems/resources/computeservers.

Modules environment is gcc-8.1, mpi/openmpi-x86_64.

Notice that one must use cuda 9 or higher verision to support __syncwarp in P1.

P1: MPI ring communication

To model ring communicaiton along a ring, with length=N, circulantly, we don't actually need N processors since there are only two modes active in communication at each repeat. If we do it smartly, we only need two processors indeed. In even repeat, info is sent from proc0 to proc1, while in odd repeat, info is sent from proc1 to proc0. Structure of codes is kind of similar with pingpong.cpp in lecture 10, but modified a lot.

There's makefile that compile the codes automatically as

```
mpic++ -std=c++11 -O3 -march=native -fopenmp
```

Module environment is gcc-8.1, mpi/openmpi-x86_64

One should run this programming with command like

```
mpirun -np 2 ./int_ring <NRepeats>
```

There several keys in the code that need to be emphasized. WITHOUT DOING THESE CORRECTLY, THIS PROGRAM IS NOT WELL-FUNCTIONAL

- Here we should use type int instead of char. Especially when length of the ring is big, which means the sum of all "rank" is big, char type is OVERFLOW. This means that we should use MPIINT instead of MPICHAR in MPI communication. And We should be careful when using MPIReiv and MPISend for the size of bytes we send. Thus, in the second part of this problem, we are passing 2MB/sizeof(int) long int array along the ring.

- We need to be careful with DEADLOCK of communication along this ring and smartly arrange MPIReiv and MPISend to make sure infomation goes smoothly along the ring. At lease for the original pingpong.cpp code, that logics would not apply here.

Results

I runs this codes with 1000 repeats on CIMS's machine crackle{1-4}.cims.nyu.edu and results are as following.

Machine: Crackle	**1**	**2**	**3**	**4**
Latency/ms	8.404679e-04	8.922210e-04	4.423610e-04	8.518831e-04
Bandwidth(GB/s)	1.091566e+00	1.392411e+00	1.343077e+00	1.426949e+00

P2 Details of final project plan

Date	**Members**	**Tasks**
04/29 - 05/03	(Zhe & Guanchun)	Literature search, discuss algorithms and write pseudocode and think about implementation with CUDA
05/03 - 05/10	(Zhe & Guanchun)	Write code in CUDA, run the test program & implement system code for Ewalds sum problem
05/10 - 05/16	(Zhe & Guanchun)	Finish the coding, find proper Stokes flow problem to run scalability test, prepare the numerical results & work on the report
05/16 - 05/19	(Zhe & Guanchun)	Finish the report and the presentation slides.