

HPC Homework 4

Austin McDowell

Greene network test

For the network test I ran the `pingpong` code on Greene with 2 tasks split between 2 nodes with one task per node. To make sure the tasks were actually split between two separate nodes I printed out the processor names. The latency and bandwidth are shown below

```
Rank 1/2 running on cs064.nyu.cluster.  
Rank 0/2 running on cs063.nyu.cluster.  
pingpong latency: 1.723910e-04 ms  
Rank 0/2 running on cs063.nyu.cluster.  
Rank 1/2 running on cs064.nyu.cluster.  
pingpong bandwidth: 1.283344e+01 GB/s  
slurm-5292078.out (END)
```

Figure 1: ping-pong timing on Greene

MPI ring communication

I started by making sure that my ring was working properly. With 4 processors and $N = 3$ the message should be 12 at the end.

```
The loop is on iteration 1/3 with 4 nodes, memo = 4  
The loop is on iteration 2/3 with 4 nodes, memo = 8  
The loop is on iteration 3/3 with 4 nodes, memo = 12  
run_output (END)
```

Figure 2: checking that message is correct

Next I check the latency of Greene using 2 nodes, 1 task per node, and passing the message back and forth. For this test I ran the loop $N = 100$ times. The latency timing is shown below.

```
The loop is on iteration 98/100 with 2 nodes, memo = 196  
The loop is on iteration 99/100 with 2 nodes, memo = 198  
The loop is on iteration 100/100 with 2 nodes, memo = 200  
integer ring time = 3.930481e-04  
(END)
```

Figure 3: checking Greene latency for integer message with two nodes, $N = 100$

Next I tried to measure the bandwidth of Greene using 4 nodes and 1 task per node. I passed an integer array of size 50,000 between the four nodes $N = 1000$ times and measured the bandwidth by calculating the size of the array and timing the full process.

```
The loop is on iteration 997/1000 with 4 nodes, memo = 3988
The loop is on iteration 998/1000 with 4 nodes, memo = 3992
The loop is on iteration 999/1000 with 4 nodes, memo = 3996
The loop is on iteration 1000/1000 with 4 nodes, memo = 4000
integer ring bandwidth = 2.579343e+00 GB/s
(END)
```

Figure 4: checking Greene bandwidth for integer message with four nodes, $N = 1000$

Pick one

b)

For this problem I attempted part b) : 2D Jacobi smoother with MPI. It's hard to say whether I succeeded or not. My code runs without any errors and I get a decreasing residual, however, the residual is quite large (larger than when I wrote my 2D Jacobi with OpenMP). I was able to run this on Greene with multiple processors. For a grid size of 128 ran for 1000 iterations on 4 nodes my residual begins at 8191.45 and decreases to 8143.36 in 35.29 seconds.

For a grid size of 64 ran for 1000 iterations on 4 nodes my residual decreased from 2047.45 to 1998.1 in 1.7 seconds.

I think I remember that this method had very slow convergence for larger N , maybe this is what I'm seeing here.

Final project pitch

For my final project I plan on implementing vector intrinsics into the C++ code `boxfit`. Found on github here: github.com/hveerten/boxfit. The `boxfit` code calculates the synchrotron radiation seen at different observer angles from the output files of a 2D hydrodynamics simulation. I hope to speed up the radiation calculations by using vector intrinsics in the most calculation-heavy portion of the code. I have also reached out to the developer of `boxfit` and asked if there are any areas of the code that they think could be improved upon.