Lab 8

Math 9830 Timo Heister, heister@clemson.edu

Note: Unless specifically asked to submit a solution, just work on the exercises and keep track of your progress in your journal.

1. Determine the number of (virtual) cores in your machine (hint: see the file `/proc/cpuinfo`). Figure out if your machine uses hyperthreading. Also report what `std::thread::hardware_concurrenc` returns. **Report in your journal.**

2. Write a program using MPI that sends a token around in a ring pattern from one processor to the next:

   - The token is of type integer starting with the value 0 on rank 0.
   - The token is send from the process with rank i to the process with rank $1+i$ (wrapping around to 0). Before every send, the value of the token is incremented by one.
   - The process is repeated for three rounds and process 0 announces the current round.
   - Each process prints to the screen when a message is received (include the rank and the token value). Finally, process 0 outputs the final value before exiting.
   - Use MPI_Barrier to ensure the output appears in order.
   - **Submit your .cc file**

   Example output (try to match this):

```
$ mpirun -n 3 ./main
We are having a party with 3 processes!
ROUND 0
process 1 got token = 1 from rank 0
process 2 got token = 2 from rank 1
process 0 got token = 3 from rank 2
ROUND 1
process 1 got token = 4 from rank 0
process 2 got token = 5 from rank 1
process 0 got token = 6 from rank 2
ROUND 2
process 1 got token = 7 from rank 0
process 2 got token = 8 from rank 1
process 0 got token = 9 from rank 2
final answer: 9
```

3. Watch Wolfgang's video lecture about parallel debugging `https://www.math.colostate.edu/~bangerth/videos.676.41.25.html`

4. MPI Collectives ("let's play dice"):

   - In a loop, let each process roll a dice (draw a random number between 1 and 6 using `1+rand()%5` (initialize the random number generator at the start with `srand(time(NULL)+rank);`). Exit the loop on each process if a) nobody rolls a 1 or a 2 (hint: minimum of rolls is?), and b) the average value is exactly 4 (hint: or the sum is?). Otherwise, all processes try again.
   - `Gather` the rolls from the last successful round on rank 0 and print them.

- Only use MPI collectives and only print things on rank 0. Your final output should be something like:

```
$ mpirun -n 6 ./main
Running with 6 ranks. Trying to roll a sum of 24 ...
It took 213 rolls to get: 6 3 6 3 3 3
```