Spring 2023: Advanced Topics in Numerical Analysis: High Performance Computing Assignment 4, due Apr 23, 2023

Handing in your homework: Hand in your homework as for the previous homework assignments (git repo with Makefile), answering the questions by adding a text or a LaTeX file to your repo. This assignment requires running some code on the NYU Greene supercomputer.

1. **Greene network test.** Use the `pingpong` example from class to test the latency and bandwidth on Greene. To do that, you must send messages between CPU cores that are on *different* nodes, such that they go through the network.[1]

2. **MPI ring communication.** Write a distributed memory program that sends an integer in a ring starting from process 0 to 1 to 2 (and so on). The last process sends the message back to process 0. Perform this loop N times, where N is set in the program or on the command line.

   - Start with sending the integer 0 and let every process add its rank to the integer it received before it sends it to the next processor. Use the result after N loops to check if all processors have properly added their contribution each time they received and sent the message.
   - Time your program for a larger N and estimate the latency on your system (i.e., the time used for each communication). Use either Greene or several machines on the the NYU CIMS network for this test.[2] If you use MPI on a single processor with multiple cores, the available memory is logically distributed, but messages are not actually sent through a network.[3] Specify the timings you find and which machines you ran on.
   - Hand in your solution using the filename `int_ring.c`.
   - Modify your code such that instead of a single integer being sent in a ring, you communicate a large array of about 2MByte in a ring. Time the communication and use these timings to estimate the bandwidth of your system (i.e., the amount of data that can be communicated per second).

3. **Pick one out of the following two:**

   (a) Implement an MPI version of the scan function we implemented in OpenMP. To make things easier, you can do something you should not (or cannot) do in practice, namely fill a long vector of random numbers on processor number 0 and split up the vector and communicate it to the other processors (you can use the `MPI_Scatter` function for this communication). After scattering the data, each processor performs a local scan. Then, each processor shares their offset with everybody else. The simplest way to do this is to use an `MPI_Allgather`. After that, each processor knows everybody else's offset and can figure out (by adding the offsets of processors with smaller rank) how to modify its vector entries.

[1] To remind you how to do this, see *Step 2* from here https://sites.google.com/nyu.edu/nyu-hpc/hpc-systems/greene/getting-started. When you hand in timings, include the names of the nodes on Greene your code ran on.

[2] See the computing CIMS information sheet posted in the first week of the class on how to use multiple hosts when using `mpirun`. Note that on each host, the same compiled file must be available, i.e., they need to have a shared directory such as the home directories in CIMS.

[3] It depends on the implementation of MPI how it handles sending/receiving of data that is stored in the same physical memory.

(b) Extend the MPI-parallel implementation of the 1D Jacobi smoothing from class to 2D. You can take the number of points in each dimension as a power of 2 such that you can easily split it into pieces for 2×2, 4×4, 8×8 etc processors. Instead of communicating one point to the right and one to the left as we have seen in 1D, now you communicate n points to left, right, top and bottom. Most other things should be similar.

4. **Pitch your final project.** Summarize your current plan for the final project. Detail *what* you are planning to do, and with *whom* you will be cooperating. The preferred size of final project teams is two, but if this makes sense in terms of the size of the project, three team member or doing a project by yourself is fine as well. Each team is expected to give a 10 minute presentation about their problem and experience in the final's week and hand in a short report (these final presentations will likely be on May 8 and May 15, and additionally either on May 9 or 16). A short summary of what you intend to do is sufficient; if you need feedback on your plans or would like to discuss, please message me on Slack. Please take a look at the list of example projects, but you are encouraged to also work on a project that is motivated by your own research. Note that I will request regular updates on the progress you are making over the next weeks and will help if you get stuck.