Laboratory # 1

Summary

Students will convert a serial code to a parallel MPI code.

Goals

The goals of this first lab are:

1. To ensure you have taken the steps to access Alpine, part of CU Boulder's Research Computing resources,
2. That you know how to run jobs in parallel on multiple cores,
3. That you are familiar with queues, slurm, and batch jobs,
4. That you can write a rudimentary MPI code.

During Lab: Team Development

In this lab, you will work on a team of two or maybe three (if there is a spare that needs a team). You will also submit a joint report. Specifically, you will each submit a file on Canvas independently, but it will be the same file that you submit.

1. From the `hpsc-2024` repository on GitHub.com, obtain the `lookup` software, located in `./codes/lookup`.
2. Together, expand the tabular lookup software such that it can perform lookups (linear interpolations) for an array of 20 values.
   (a) Convert `xVal` and `yVal` to arrays of length `m`. And set `m = 20`.
   (b) Populate `xVal[]` with 20 values, as follows: `for ( int i = 0 ; i <= m ; ++i) xval[i] = 2.*i;`
3. Using MPI, parallelize the software to run on `nPE` processors such that each processor performs a lookup on a sub-set of the `xVal[]` array. While each processor should have the entire data table, each processor should only be required to populate part of the results, i.e., the `yVal[]` array. Make it be that each of the processors are doing roughly the same amount of work.
4. Have processor 0 collect the lookup results from the other processors and print the results for all 20 values to the screen.
5. Write a slurm script that runs the software as a batch job on 4 processors.
6. Run the software as a batch job on 4 processors, redirecting the screen output (called "std out" in Linux) to a file using the ">" operator.

After Lab: Team Lab Report

1. Include in your report the following:
   (a) The team's parallel code in Appendix A.
   (b) The team's slurm batch script in Appendix B.
   (c) The output your team obtained in Appendix C.
2. In the main body of the report:
   (a) Parallel design: The idea behind how you parallelized the code
   (b) Self-evaluation: What worked out well, what did not work out well

Resources

- A very important link for this assignment is the CURC Documentation. From this link, you can find many resources that will help you throughout the semester.
- The above link will lead you here, for requesting a Resource Computing account: Request an Account.
- You might find it to be of great advantage to mount your home directory on Alpine from your local computer. See our Canvas page for instructions. With this approach, you can edit and plot your Alpine files using your own computer's resources.
- You can use `-X -C` when performing an `ssh` to Alpine, both to the login nodes *and* to the compile nodes as well. Doing so allows Alpine to "tunnel" X-Windows graphics through your `ssh` session. The `-C` performs compression and makes it faster.
- Remember to load the intel and impi modules, i.e., `module load intel` and `module load impi` before you compile and run your codes. Remember, you cannot run in parallel on a compile node.

Grading Rubric

In all of your lab reports, please note these expectations:

- A professional tone, with no use of slang or anything informal
- Grammatically perfect, spell-checked
- A concise report, with no "stories", i.e., accounts of hurdles you encountered and overcame - Please save those for the "Self-evaluation" section.
- When in doubt, adopt the tone that you would adopt if you were being paid by a client to do the work in this lab and you are reporting on what you have done, in expectation they will pay the invoice you will send for the work.

Component	Expectations	Weight
Appendices A	Code listing with in-line comments	30%
Appendices B	Slurm script listing with in-line comments	20%
Appendices C	Results	20%
Parallel design	Less than one-page description	20%
Self-evaluation	Results, if obtained, in Appendix C, discussion of outcome	10%