Laboratory # 2

Goals

The goals of this lab are:

1. To explain the design for parallelizing a physics code that computes on a spatial discretization (mesh).
2. To implement that design.
3. To learn debugging techniques for parallel programming.

During Lab: Team Development

You will be given `fd_mpi`, a 2D steady-state diffusion code. You must complete the parts of the code labeled "TO-DO in Lab", which appears in multiple places in the various source code files, including `fd.cpp` and `linear_solver.cpp`. The place where it appears most prominently is in `mpiInfo.cpp`, which is the capability (a C++ class) that organizes which part of the global 2D mesh each PE owns and also provides communication between PE. That part of the code will likely take you the most effort to complete for two reasons. First, it is the most complex and, second, you will be required to write about it. Specifically, the `mpiInfo.cpp` class implements what one would call "the parallel design" for the finite difference code. It embodies the strategy for how the mesh is decomposed and how parallel communication is accomplished. The code in `linear_solver.cpp` is also part of the design, for it is the place where the parallel communication is required.

For this lab, you are to do the following:

1. Review the "TO-DO in Lab" parts of the `*.h` and `*.cpp` files. Use `grep -i ''TO-DO''` to ensure you have found all such locations.
2. Complete those parts of the code with your teammate. Note that the `mpiInfo.cpp` code will likely require a nontrivial amount of thought. While completing that part of the code, as well as the other "TO-DO in Lab" code, you should be cementing your understanding of what we would call "the parallel design" for `fd_mpi`.
3. There are two test cases for the parallelized code. One is to run it on one PE and plot the results. The second is to run it on four PEs (2 in the $x-$ direction and 2 in the $y-$ direction), and plot those results as well. The `run` file, included in the `./codes/fd_mpi/src` directory shows how to run the code.[1] Those plots, which can be created by the lightweight package "gnuplot", must be included in your report. If you use gnuplot, note that `fd` automatically writes a short, human-readable file called "`pc`", short for "plot commands". To use that plot commands file, do the following:
   (a) Run gnuplot. You can install it on your own computer for free. It is already installed on Alpine.
   (b) At the gnuplot prompt enter, `load 'pc'`, which will display the plot to your screen.
   (c) Also, take a moment to read the `pc` file itself. It contains instructions on how to make a graphics file for inclusion in your report.

[1] It also provides some handy processing of the screen output that can parse output from different PE to different files.

4. Ensure that the plot for the 4-PE case is continuous, with no gaps in the temperature field where the different PE's solutions touch. You may need to rotate the plot on the screen and zoom in to be sure.
5. If you see gaps, or the solution is not nice and smooth, you will have to debug the code. Expect that. Here are some tips for debugging:
    - **Asymmetric Reduced Spatial Decomposition:** Run a 2x1 case, i.e., 2 PE in the $x-$ direction and 1 PE in the $y-$ direction. Looking at that resulting plot might help you find bugs. You can then swap the above strategy, i.e., 1 PE in the $x-$ direction with 2 PE in the $y-$ direction.
    - **Coarse Mesh Testing:** Use far fewer cells. Modify `./run` to use, say, 3 cells in each direction, or maybe 2 cells in the $y-$ direction with 3 cells in the $x-$ direction. It is amazing what you can learn with a much coarser mesh. Note that you must use at least 2 cells in any given direction.
    - **Printing Values:** In combination with the above techniques, print `b`'s and `Solution`'s values inside the linear solver, preceding the outputs with the `myPE: <PE number>` so that the lines can be parsed into different files (see, again, `./run`). Ensure the solution from one PE is correctly being communicated and stored on the neighboring PE. Consider, also, printing the `i,j` values, then the natural point number, along with each of the `b` or `Solution` values so that you can more easily track where, in the mesh, each value lies.

    If you are unsuccessful getting the parallel version to work, for full credit you must exercise the above techniques, or similar techniques. Use the names listed in bold, above, in your report, with truncated output that demonstrates you have systematically worked through the debugging process. Even if you were ultimately not successful in finding your bug(s), you will receive full credit if you carry out and report on the debugging process professionally.

After Lab: Individual Lab Report

Lab reports will be jointly written but submitted individually. In other words, both team members will submit the same document to Canvas. In your report, include the following:

1. In the main body:
    (a) A concise but illustrated description of the parallelization design. It should include and reference an image of meshes with ghost cells and refer to the Jacobi solver and when parallel communication occurs.
    (b) The two test case plot results or, if none, reference to the results of the debugging process, in Appendix C.
    (c) Self-evaluation: What worked out well, what did not work out well. This is the only part of the report that can include time-lines, and a "story". The rest of the report should strike a professional, concise tone.
2. In the Appendices:
    - Appendix A: A listing of the `GridDecomposition` routine, only, not the entire class.
    - Appendix B: A listing of the `ExchangeBoundaryInfo` routine, only, not the entire class.
    - Appendix C: Required only if the code did not produce correct results. Include a description of the debugging process.

Resources

- Helpful aids on MPI and C MPI and C at the CU Boulder Research Computing website.
- `fd_mpi` in the hpsc-2024 repo.

Grading Rubric

Component	Expectations	Weight
Main Body - Design	About one-page description with illustration	35%
Main Body - Results	About one-page description with plot	35% or 0%
Self-evaluation	What worked what did not work	10%
Appendix A	Code listing with in-line comments	10%
Appendix B	Code listing with in-line comments	10%
Appendix C	Only if code not working: Debugging process	0 % or 35%