CS 420, Parallel Programming, Fall 2019
MP6

Szaday, Justin   Shah, Ishan Pankaj

Due Date: November 22nd, 2019, at Midnight

1 Overview

The purpose of this assignment is for you to gain experience with MPI and stencil computations, iterative programs that update array elements based on a stencil (a fixed pattern). Due to the complexity of this MP's skeleton and its required runs, we recommend starting it early (especially if you have no prior experience with MPI).

2 A Five-Point Stencil in MPI

2.1 Five-Point Stencil

Five-point 2D stencils are made up from a point and its four neighbors. This pattern is typically used in iterative contexts, where the output of an iteration is used as the input for the next iteration. An example of a five-point 2D stencil is the Jacobi algorithm, which is discussed in Section 3.3 of the textbook.

In a distributed context, the computation is distributed across a grid with overlapping regions between processing elements (PE's), called *ghost* cells. These ghost cells must be large enough to contain all of the neighbors needed for an iteration, and must be updated with the values from other PE's between iterations. See Figure 1 for a visual representation of the grid layout used in this MP. Note, each tile of the grid is padded to accommodate the ghost cells and some of the padding goes unused.

2.2 Your Task

We have provided a skeleton program that handles much of the setup for this assignment, which can be pulled from the `_release` repository. Your task is to implement the communication for exchanging the ghost cells between PE's inside the functions `send_ghosts` and `recv_ghosts`. The skeleton distributes PE's across a `kGridRows` by `kGridCols` grid, with each PE taking a $\frac{kWidth}{kGridRows}$ by $\frac{kHeight}{kGridCols}$ tile of the `kWidth` by `kHeight` workload. The variables `kGridX` and `kGridY` represent a PE's position in the grid (noting that a PE's $rank = (x * numCols) + y$); use this information to determine whether a PE has north, south, east, and/or west neighbors that require ghost data (and their corresponding ranks).

We have provided helper functions to copy ghost cells between send/receive buffers and a PE's tile: `copy_rowbuf_out`, `copy_colbuf_out`, `copy_rowbuf_in`, and `copy_colbuf_in`. The "in" functions should be used for receives (copy data *into* the tile), and the "out" functions for sends (copy data *out of* the tile). The provided buffers, `colBuf` and `rowBuf`,

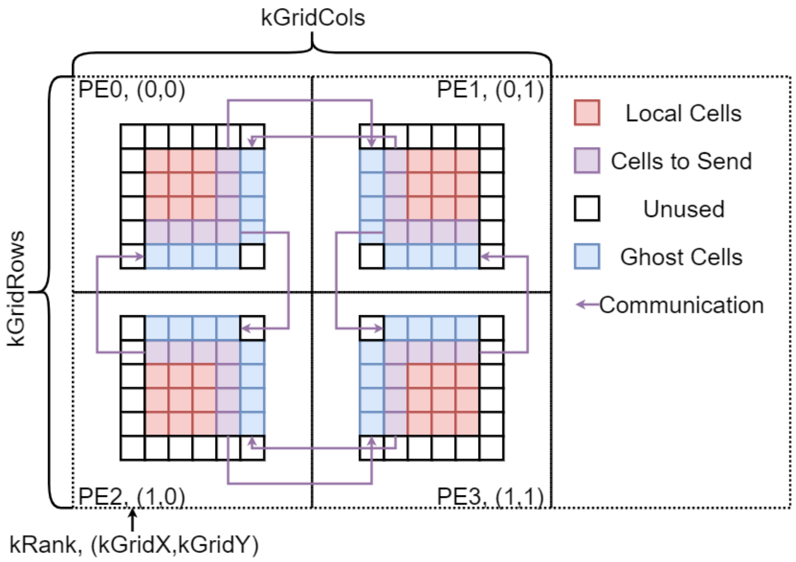

Figure 1: Grid layout used by this MP's skeleton.

are appropriately sized for columns and rows of the PE's tile respectively. Use blocking MPI calls, e.g. `MPI_Send` and `MPI_Recv`, for this assignment. For reference, we expect you to write somewhere between 20-30 lines of code for this assignment.

?

Strong scaling refers to how the execution time varies with the number of PE's for a fixed, overall problem size. For a single-node run and an 8192 by 8192 workload with 16 iterations, plot *Execution Time vs. Number of PE's* for PE counts: 1, 2, 4, 8, and 16. Briefly comment on your program's *strong scaling.*

?

Weak scaling refers to how the execution time varies with the number of PE's for a fixed problem size per PE. Plot *Execution Time vs. Number of PE's* for a fixed tile size of 1024 by 1024 per PE for PE counts: 1, 2, 4, 8, and 16; include data labels for each point indicating the overall workload size. Briefly comment on your program's *weak scaling.*

?

As you increase the amount of computation relative to the amount of communication, the cost of communication will be attenuated. For our stencil program, we can increase the amount of computation by running for more iterations. Run your program with one PE and four PE's for an 8192 by 8192 workload for iteration counts: 1, 2, 4, 8, and 16. Plot *Speedup vs. Number of Iterations* using these results, with each point's value being $(n_{its}, \frac{time_{np=1}}{time_{np=4}})$. Briefly comment on these results.

?

For an 8192 by 8192 workload with 16 iterations, how did running your program with multiple nodes affect its parallel performance?

2.3 Skeleton Notes

This MP's skeleton program is rather complex, so it may be worth taking some time to understand it before diving into the assignment. Unlike the other skeleton's we have provided – this MP's verboseness is determined at compile-time using a macro variable, `VERBOSE_LEVEL`. You can specify this to the compiler using `-DVERBOSE_LEVEL=X`, where: $X = 0$ produces no output (use this for performance analysis), $X = 1$ prints information for correctness checking, and $X = 2$ prints additional information about the program. Since no communication is necessary in single PE runs, these runs can be used as the basis for correctness checking. Note, your program's output should not vary with valid PE counts for a fixed workload size and number of iterations.

The expected usage of the program is `./stencil2d <width> <height> <numIts>`, specifying the overall width and height of the workload and the number of iterations to run for. This program assumes that the workload will always be evenly divisible by the number of PE's, dividing it up based on the largest prime factor of the number of PE's; an assertion will fail when this condition is not met.

2.4 Running MPI Programs on the Cluster

The following command is used to load MPI into the cluster's environment:

```
module load openmpi/3.1.1-intel-18.0
```

Among others, this will add the programs `mpicc` and `mpirun` to the `PATH`, used to compile and run MPI programs respectively. The number of PE's to be used is specified via `mpirun`; for example, the following command will run the program `stencil2d` (with appropriate arguments) on 32 PE's:

```
mpirun -np 32 ./stencil2d 2048 2048 16
```

You are expected to test multi-node runs for this assignment; to do so, change the value of `nodes` in your batch script. Especially for this assignment, we ask again that you do not execute any of your programs on the login nodes.

3 Submission Guidelines

3.1 Expectations For This Assignment

The graded portions of this assignment will be your stencil program and answers to the short answer questions.

3.1.1 Points Breakdown

- $4 * 13pts.$ — Each Short Answer Question
- $48pts.$ — Correctness of your Stencil Program

3.2 Pushing Your Submission

Follow the git commands to commit and push your changes to the remote repo. Ensure that your files are visible on the GitHub web interface, your work will not be graded otherwise. Only the most recent commit before the deadline will be graded.