CS 420 - Intro to Parallel Programming

MP3: MPI and Stencil Computations - MP Notes

March 28th, 2023

Introduction

- **Brief overview of the MP**
  1. Functions to send and receive ghost cells. What are ghost cells? (Refer to lecture slides).
- **Purpose**
  1. Gain experience with MPI and stencil computations.
  2. Infer results from distributed jobs submitted to the cluster.
- **Note:**
  1. We have observed that sometimes, results are not collected across 8 nodes. If this is the case, please try re-submitting the job to the cluster a few times. **If the results for 8 nodes are still unavailable it is okay to ignore the data point for 8 nodes while plotting the required graphs.**

Five Point Stencil

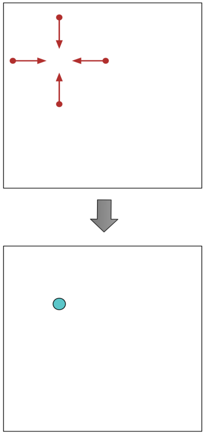
Figure: Five-Point Stencil

- **What is a stencil pattern ?**
  1. `https://en.wikipedia.org/wiki/Iterative_Stencil_Loops`
  2. For example: Figure 1 represents a stencil (Slide 32, Lecture 13).
- **Ghost Cells**
  1. In a distributed context, each process works on a smaller grid and there are some overlapping cells for each grid.

Visualisation of ghost cells

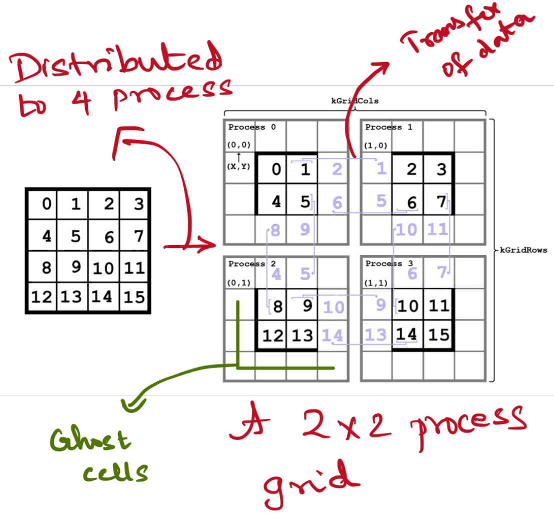

Figure: Illustration of how the 2-D workload is distributed to different process and an illustration of their ghost cells

Your task - An overview

Implement communication for exchanging ghost cells

1. **Your task is to only implement functions that exchange ghost cells.**
2. Implement send_recv_ghosts function in `stencil.c` using **non blocking MPI Communication**, `MPI_Isend` and `MPI_Irecv`.
3. In order to send and receive data, buffers should be used. We have given utility functions `copy_colbuf_in`, `copy_colbuf_out` to copy ghost columns and `copy_rowbuf_in`, `copy_rowbuf_out` to copy ghost rows into buffers. More information on these functions are present in the instructions pdf.
4. Framework handles the stencil computation.

Process grid visualisations

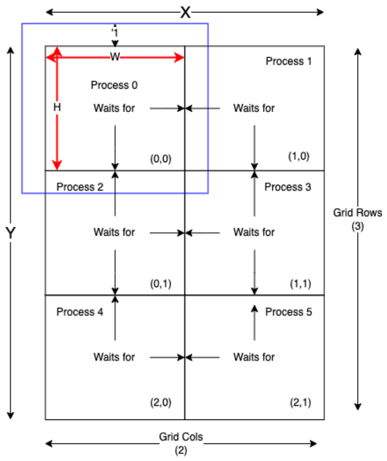

Figure: A 2 x 3 process grid

- **How to identify a process's neighbors ?**
  1. Each process's (rank's) position on the **process grid** is given by its kGridX and kGridY values. For example, Process 2 has kGridX has 0 and kGridY as 1. Can you determine its neighbors by knowing these values ?
- **Wait**
  1. Since we are performing a non-blocking communication, it is important that we wait for them to finish before we write the data to the receiving process tileś ghost row/column . Refer to lecture 13 slide 66 to understand how to wait for different processes. Can we use the same kGridX and kGridY values to determine which neighbors each process needs to wait for?

Understanding the Skeleton Program

- **The skeleton program (`main.c`).**
  1. The main program, initializes the MPI environment and creates the necessary tiles and buffers required by each process (rank).
  2. Then we use the `send_recv_ghosts` function that you write to exchange buffers between these processes.
  3. Post exchange of ghost cells, you need to copy the buffers to the ghost rows/columns of the tile in the processes.
  4. Once the communication function is returned, the framework calculates the local stencil at each process.
  5. Steps 1-4 run for `iter` number of times.

  We have added comments wherever necessary in the skeleton code. Please go through the same to understand the flow.
- **Adding debug logs and running**
  Please refer to section 2.8 of the instructions to know how to add additional logs for debugging purposes. We have also indicated how to run the program using the interactive `srun` command in section 2.6.
- **Benchmarking**
  Section 2.7 describes running the batch script to submit jobs to the cluster and also describes the output from the batch job.

Common Pitfalls

Here are some of the common mistakes / issues that can occur when implementing this MP.

- **Segmentation faults when copying in and out of buffers**
  Segmentation faults occur when there is an illegal memory access. Make sure you are using the right indices when copying to and from the given buffers (consider the fact that the tile has additional rows and columns for accommodating the ghost cells).
- **Wait before copying into tile's buffer in recv_ghost**
  Make sure you wait for `MPI_Irecv` to complete before copying the values into the tile's ghost rows/columns using the given helper functions. Not waiting can yield incorrect results and failed test cases.
- **Some issues with srun (due to change in processor type)**
  Sometimes it so happens that when using `srun` (for interactive jobs), the job can be scheduled on a node that does not have an INTEL processor. You would see errors related to compilation in this case. Compile it on the interactive job and then use the srun. Note that this is only for the interactive job for debugging. If you use our scripts, the compilation is taken care at the compute node and you will not see any errors.
- **Long queue wait times**
  For this MP, the batch jobs are configured to run across 8 nodes and based on the availability of the cluster, there could be longer wait times.