Final Project Tutorial, Part 2: Writing with MPI I/O

Instructions : Complete the tutorial using C.

To use C, edit only the file `C/main.cpp`.

If you get confused with GitHub/CMake logistics, refer back to Homework 0 here.

1 Writing a Text File

Edit the `writeTranspose()` method. This method takes a 2-D array and its dimension, and writes it to file with a specified filename. You will need to open this file, paying special attention to the `MPI_MODE` passed to `MPI_File_open`. Remember that this method will need to create a file or overwrite it if it exists. To accomplish parallel writing across processes, we will assign each process a portion of the matrix to write. This partition will allow each process to hold a buffer of the elements it will write.
In C: Determine the elements in the matrix that each process is responsible for, and write them to a buffer.

```
for (int i = 0; i < rowsPerProc; i++) {
    int globalRow = startRow + i;
    for (int j = 0; j < n; ++j) {
        ptr += sprintf(ptr, "%8.2f ", matrix[globalRow * n + j]);
    }
    ptr += sprintf(ptr, "\n");
}
```

After each process has a buffer containing the elements it will write, you will need to determine the position in the file it will write the buffer to using an `MPI_Offset`. Then use that offset, the buffer, and the size of the buffer to write to the file with `MPI_File_write_at_all`. At the end of the method, make sure to free any allocated variables and close the file.

2 Update Main Method

After writing the function, incorporate it into your main method (`tutorial_main`). This function should be called after initialization and reading but before finalizing. The function requires a filename that will be used to create or overwrite the text file, as well as the read matrix and its dimension.

3 Compile Code

Compile your code with:

```
sh compile.sh
```

4 Run Your Code

I have provided an example for you at `examples/example.cpp`, which will call your `tutorial_main` method. To run this example, first:

```
cd build/examples
mpirun -n <np> ./example matrix.bin 16
```

where `<np>` is replaced by the number of processes on which you want to run. The project is set to generate a matrix of size 16 within the examples directory. Verify that the matrix printed and the matrix in the text file match.

```
mpirun -n 4 ./example matrix.bin 16
```

The transposed matrix should print to the screen and a text file should be created with the same elements. **Make sure to try multiple different, even process counts.** Depending on your MPI implementation, you may need to pass the oversubscribe flag if you surpass the number of available hardware threads on your system, e.g.

```
mpirun -n 16 --oversubscribe ./example matrix.bin 16
```

5 Check for Correctness

To check that your code is working, do the following:

1. Make sure all tests are passing locally

   ```
   sh compile.sh
   cd build
   make test ARGS="-L write"
   ```

2. Push all changes to GitHub
3. Check that your GitHub actions are passing

The End.