Assignment2 Part 1: Due on Feb 28

In this assignment, you are supposed to calculate the product of two matrices A (of size N*32) and B (of size 32*N), which should be an N*N matrix. Specifically, you are supposed to

- Design a parallel scheme for computing matrix multiplication, including how to:
  - Separate the task into divisions and let each process finish one division
  - Transfer data between processes
  - Form the output matrix using the result of each process.
- Implement the parallel mechanism with 3 different types of communications in MPI, that is to say you have to write 3 programs that differ in communications between processes:
  - Blocking P2P (point-to-point) communication
  - Collective communication
  - Non-blocking P2P communication.
- Observe the running time of your programs; change some of the parameters to see how it is associated with N and communication type (and number of processes, if available).
- Try to prove that the theoretical time complexity is O(N^2)
- Finish a report that meets the requirements.

Programming tasks

Step 1: Install MPI libraries on your computer and make sure it works.

Step 2: Create a function, Multiply_serial() to perform multiplication without parallelism. Example code (assuming that matrix elements are row-major stored in an array):

```
void Matrix_Multiply(float *A, float *B, float *C, int m, int n, int p){
    int i, j, k;
    for (i = 0; i < m; i++i){
        for (j = 0; j < p; j++){
            C[i*p + j] = 0;
            for (k = 0; k < n; k++)
                C[i*p + j] += A[i*n + k] * B[k*p + j];
        }
    }
}
```

Create a function, IsEqual(), that examines if two matrices are exactly the same. The function should have the form:

```
int IsEqual(float *A, float *B, int m, int n)
```

In the function, we need to check if every pair of corresponding elements in A and B are the same. Once a discrepancy is found, the program can instantly return 0 (false). Return 1(true) if A and B are equal (no discrepancy is found). Later the function will be used to verify the results of your MPI programs. You can write a simple main() function to make sure IsEqual() works.

Step 3: Implement your parallel algorithm in main() function, using blocking P2P communication (MPI_Send/MPI_Recv) between processes. main() should at least:

- Initialize and finalize MPI environment
- Let Process #0 generate A (of size N*32) and B (of size 32*N) using random numbers in [0, 1].
- Implement communications between Process 0 and other processes
- Compute A*B using your parallel scheme (assuming the result is stored at C)
- Let Process #0 compute A*B by Multiply_serial() function (assuming the result is stored at C_serial)
- Let Process #0 check if C and C_serial are equal (using IsEqual() function) so as to verify your parallel algorithm
- Let Process #0 get the running time of the two computations above

Step 4: Compile and run your program. You can use 2 or 4 or 8 processes (according to how many cores your computer CPU has). Take down the running time for serial, blocking communication, collective communication and non-blocking communication.

Report

Your report should have the following:

- Elaboration of the parallel algorithm you have implemented in terms of design principle and implementation. A good algorithm description can make readers understand how your algorithm works even without reading/running the codes.
- Summary of the differences among your 3 versions of parallel algorithm in terms of principle and implementation.
- Discussion on restrictions of N in your program, if there is any (for example, the sample code assumes N to be a multiple of number of processes). Explain why there should be such restriction.
- Lists of running times (together with the time-N curves) of the four methods provided different values of N: serial (Assignment 1), MPI with blocking P2P communications, MPI with collective communications, MPI with non-blocking P2P communications.
- Your findings on the running times you have observed, like if they agree with theoretical time complexity, if being parallel can accelerate the computation and so on. Try to explain your findings.

Submission

Please submit a zip package including:

- 3 C code files, each corresponding to an MPI communication method. Please attach necessary comments to your code. Include a Makefile
- A report (.pdf) that meets the requirement above.