Homework #6 Parallel Matrix-Matrix Multiply
Due: March 23, midnight.

Overview

You are going to write and evaluate Cannon's algorithm and Fox's algorithm for parallel matrix-matrix multiplication. For this assignment you must use a Cartesian topology, a custom communicator, and derived Datatype.

Software Development

Write a program called `multiply` that multiplies two matrices. Your program should accept five inputs:

- Algorithm: specified as `--fox or --cannon`
- First matrix input filename: specified as `-m1 <filename>`
- Second matrix input filename: specified as `-m2 <filename>`
- Output filename (optional): if specified, use `-o <filename>`
- Verbose (optional, see discussion below): if used, specify `--verbose`

Examples:

```
mpirun -np 4 ./multiply --fox -m1 m1000.hdf5 -m2 m1000.hdf5
mpirun -np 8 ./multiply --cannon -m1 m100.hdf5 -m2 m100.
mpirun -np 4 ./multiply --fox -m1 ma10.hdf5 -m2 mb10.hdf5 -o out
./multiply --fox -m1 m1000.hdf5 -m2 m1000.hdf5 --verbose
```

You may assume your input matrices will be square (N x N) and that the size will always be evenly divisible by the square-root of the number of processors (e.g. N mod sqrt(p) == 0). Your code must also run in serial.

You should use the example HDF5 code provided on the d2L site to assist you with reading and writing HDF5 files. Use the partial read from the previous homework where one rank reads the sub-matrix information from the specified files and distribute the partial matrices to the other ranks. Once a rank has received its information, the process continues until all ranks have their portion of the matrix.

You can use the code provided from the last homework to generate matrix files in HDF5. *Modify this code to make your performance better.* You should chunk the files for optimal performance based on a range of processors.

Only output your data if the -o <filename> flag is specified. Output is an HDF5 2D matrix (see helper code).

The verbose mode should collect timing information. Please output the number of processors used, the problem size, the total time, and the parallel processing time. Remember, you may want to warm-up the `MPI_Wtime`. Your output should be a single line with no other information. Fore example:

```
>mpirun -np 4 ./multiply --fox -m1 m.hdf5 -m2 m.hdf5 -verbose
4 <problem_size> <total_time> <parallel_time>
```

Analysis

You will be running a series of experiments. Consider writing bash scripts (or your favorite scripting language) for managing the execution of the different trials.

I. Chunking

You should experiment with different chunking sizes to determine which one is for multiplication on JANUS. Use this for your analysis.

- What size and shape of chunking works best for a given problem size? Hint: experiment with a large problem size and a range of sub-matrix sizes. Consider writing a separate executable for this experiment.
- Why do you think the chunking size and shape you discovered works best?

II. Serial analysis and Amdahl's Law (again)

Profile your serial implementation and estimate the percentage of time your program spends doing serial work (everything but the multiplication). Do this for problems of size 1024, 2048, and 4096. Based on this information and **Amdahl's Law**, how do you anticipate your algorithm will scale as you increase the number of processors for a give problem size? Remember, Amdahl's law doesn't require you to account for communication cost or additional MPI overhead. Show your data and explain your calculations.

III. Parallel analysis: variable number of processors

Run your experiments using 1, 4, 16, and 64 processors using problem sizes 1024, 2048, 4096, and 8192 (don't compute this in serial, it take too long. Estimate the serial time using the results of 4 processors). Do this experiment 10 times for each algorithm average the results (per algorithm and problem size).

1. Plot the **speedup** for each problem size and algorithm as a function of the number of processors.
2. How does the speedup change with problem size?
3. Do you notice **Amdahl's effect** for this problem? Why or why not?
4. Which algorithm has better speedup? Why do you think this is the case?

Assignment Submission

Your lab write up, including answers to questions and plots you generate should be clearly presented in a Portable Document Format(e.g. *.pdf file). If you use outside information, please make sure you properly reference them. Make sure all graphs are properly scaled, labeled, titled and referenced.

Create a file named <username>-hw6.tar.gz that contains a softcopy of your report, your source code and Makefiles. Please do not include object files, executables, or any *matrix files*. Submit the tar file to the d2L site by the due date. The assignment is due *before midnight* on Friday, March 23rd.

Grading

I. *Building (5 points)*
It is extremely important that we can untar your <username>-hw6.tar.gz file, change into the <username>-hw6 directory, and type make to get a successful build. Please test your tar file on JANUS to make sure you are linking to the correct libraries. We will be using the following dotkits:
- use .openmpi-1.4.3_gcc-4.5.2_torque-2.5.8_ib
- use .hdf5-1.8.6

You may use other dotkits if you'd like, but you should include them in your Makefile and mention them in a README or report if you do.

II. *Testing Correctness (12 points)*
We will run your code with double precision matrix files. Please make sure that you are reading your code with HDF5 using H5T_NATIVE_DOUBLE and writing your code using H5T_IEEE_F64LE (in the H5Dcreate() method). We will randomly generate 2 set of matrices and multiply them using 1, 4, and 16 processors. This is a total of 6 tests worth 2 points each.

III. *Testing scaling (10 points)*
We will run both algorithms on a contiguous matrix problem of size 8192 using 64 processors on JANUS scratch. You're total runtime should be no greater than 250 seconds for Cannon's algorithm. I'll post Fox's time later this week. Please check back.

IV. *Objective (15 points)*

The point of this assignment is to be familiar with MPI topologies, communicator, and custom Datatypes. You will receive 5 points for using each of the concepts in your code.

V. *Lab Report (29 points total)*
   a. *Chunking (5 points)*
      i. *First question, 3 points, second question 2 points.*
   b. *Serial Analysis (10 points)*
      i. *Data and calculations*
   c. *Parallel Analysis (14 points)*
      i. *Plot (5 points)*
      ii. *Questions 2, 3, and 4. (3 points each)*