# Assignment #4 and #5 – Hybrid Programming with Parallel I/O: MPI + Threads Assignment on Blue Gene/Q

Christopher D. Carothers
Department of Computer Science
Rensselaer Polytechnic Institute
110 8th Street
Troy, New York U.S.A. 12180-3590
Email: chrisc@cs.rpi.edu

March 29, 2016

## DUE DATE: Noon, Tuesday, April 12th, 2014

## 1 Assignment Description

For this **GROUP** assignment (up to 4 students), you are to develop a MPI-based C/C++ program which also uses Pthreads to perform the 131,072x131,072 matrix sum of a randomly generated matrix and it's transpose (e.g., $C[i][j] = A[i][j] + A^T[i][j]$). This matrix is 128 GB in size (e.g., $2^{17} \times 2^{17} \times 8/2^{30}$).

First, each MPI rank will initial it's local slice of the matrix. The smallest number of BG/Q nodes that can be used is 32. Here, each node will have a a 4 GB slice of matrix. Also, the transpose will consume another 4 GB. Those slices could be subdivided further if there are multiple MPI ranks per node or not if pthreads are used and there is only 1 MPI rank per BG/Q node.

Also, recall, the transpose of a matrix $M$ is $M^T$ where $M^T[i][j] = M[j][i]$ for all $i$ and $j$. Also, notice that the above matrix is much larger than the available RAM on any one node. So, to compute the transpose of the matrix it will have to be done in a distributed fashion where each MPI rank only collects the columns it needs to compute it's local part of the transpose. For that, you should consider how to use the `MPI_Gather` operation or you may elect to use `MPI_Isend` and `MPI_Irecv`.

Now, within each MPI task, you can spawn/create compute Pthreads that enable you to parallelize the computation of each matrix slice on a row-to-column basis. That is each thread within an MPI rank will perform the dot for a whole row and column or multiple rows/ columns depending on the number of threads, etc.

Now, here is the key question: given a resource of 16 cores using the Blue Gene/Q, how best to allocate the right mix of MPI tasks and threads such that you get fastest execution time?

**NOTE: all experiments will be run on the Blue Gene/Q system at the CCI. No experiments are to be done on Kratos except for debugging**.

The possible compute configurations are:

- 64 MPI task with zero threads
- 16 MPI tasks each with 4 threads.
- 4 MPI tasks with 16 threads each.
- 1 MPI task and 64 threads per node.

You run those above configurations using 32, 64, and 128 nodes.

Note, the trade-off here is that you'll invoke fewer `MPI` operations however, as you have more threads and fewer MPI ranks, the amount of data gathered will be larger. So, what is the right mix of MPI tasks and Pthreads for the particular Blue Gene/Q node configuration that yields the best performance (e.g., lowest execution time).

Now, the possible parallel I/O configurations for writing out the entire C matrix are as follows:

- 1 single file for all MPI ranks, with 8MB block boundaries between rank using `MPI_File_write_at_all` collective IO operation.
- 1 single file for all MPI ranks, with compact boundaries between ranks (e.g., don't care about using 8MB boundary just write the size of each MPI ranks data chunk and don't leave space gaps) `MPI_File_write_at_all` collective IO operation.
- 4 ranks share the same file using `MPI_File_write_at` (non-collective write call) with 8MB block boundaries between ranks.
- 4 ranks share the same file using `MPI_File_write_at` (non-collective write call) with compact boundaries between ranks.
- 8 ranks share the same file using `MPI_File_write_at` (non-collective write call) with 8MB block boundaries between ranks.
- 8 ranks share the same file using `MPI_File_write_at` (non-collective write call) with compact boundaries between ranks.
- 32 ranks share the same file using `MPI_File_write_at` (non-collective write call) with 8MB block boundaries between ranks.
- 32 ranks share the same file using `MPI_File_write_at` (non-collective write call) with compact boundaries between ranks.

By default, using `MPI_Init()` will enable/allow only 1 thread to invoke MPI routines. This should be fine for your implementation where the default "thread 0" on each node handles all the MPI calls. However, if you wish to allow multiple threads to invoke MPI, then you must use the `MPI_Init_thread` with the `MPI_THREAD_MULTIPLE` as the "required" argument. See:

http://www.mcs.anl.gov/research/projects/mpi/mpi-standard/mpi-report-2.0/node165.htm

**NOTE, you need to time using the BG/Q cycle counter, the parallel I/O time separately from the time spent in compute and comm. This time you don't need to time the amount of time spent in MPI send and recv. operations. Include that time as part of the overall time in compute.**

Make sure you include:

```
#include<hwi/include/bqc/A2_inlines.h>
....
unsigned long long start_cycle_time=0;
unsigned long long end_cycle_time=0;
unsigned long long total_cycle_time=0;
....
start_cycle_time = GetTimeBase();
Do stuff!!
end_cycle_time = GetTimeBase();

total_cycle_time = end_cycle_time - start_cycle_time;
```

**The clock rate on the BG/Q is 1.6 GHz.!!!!** You need to use that that cover the cycle times to seconds.

## BIG NOTE: You must write ALL data to your `scratch` directory. It is the only location that will hold it!!

Overall you'll have 3 different node scenarios, 4 different thread/MPI rank configurations per node and 8 different parallel I/O scenarios per MPI rank. Thus, you will have 96 data points for this assignment. You will want to pay special attention to cases where an MPI rank will write more than a single 8 MB block.

For these 96 data-points, generate the following performance graphs.

- For each node count, plot **COMPUTE** execution time only (y-axis) as a function of threads (not MPI ranks) per node (x-axis). This excludes the time to perform parallel I/O. This graph should have 3 plot lines - 32 nodes, 64 nodes and 128 nodes.
- For each node count, plot the data bandwidth written to the parallel I/O file system (y-axis) as function of the level of file sharing, all ranks per single file, 4 ranks per file, etc. Observe that the C matrix data your are writing is the same size. So, the bandwidth calculation is straight forward. Also, there will be two lines in each node count graph - 8MB boundary vs. compact. Thus, here you will have 3 graphs with 2 plot lines in each graph.

Write your report to explain the performance trends you observe based on the topics we have covered in lecture.

## 2 HAND-IN INSTRUCTIONS and GRADING CRITERIA

Leave your code and write-up in your **assignment4** sub-directory on your account on `kratos.cs.rpi.edu`. Note, we'll be using the following check list and point deductions when grading your assignments.

- **CORRECT SUBMISSION: 10 points.** Source code is inside your **assignment4** directory. The code compiles without errors.
- **CORRECT RESULT: 40 points.** The code produces the intended result.
- **ANALYSIS: 50 points**. Performance graph(s) are displayed with labels. An explanation of what the graphs represent is presented.
- **Other Point deductions:**
  - -5pts: no labels and no mention of labels.
  - -10pts: if only one graph presented.
  - -20pts: If no graphs exist.
  - -30pts: If no explanation (this is the main point of the assignment).
  - -3pts: If the scales on all of the graphs are the same.
  - -2pts: If the source code is not in a file, but in the pdf.