# Compiling and Running the Programs

The submission folder contains 5 sub-folders, 1 pdf file, and 1 env.output file. The table below explains the usage of each sub-folder.

|  |  |
| --- | --- |
| Folders/Files | Comments |
| BlockingMPI | This folder contains all the source code, test results, Makefile, and bash scripts for the Blocking MPI program. |
| NonBlockMPI | This folder contains all the source code, test results, Makefile, and bash scripts for the Non-Blocking MPI program. |
| BalanceLoad | This folder contains all the source code, test results, Makefile, and bash scripts for the Non-Blocking MPI program with balanced load. |
| Serial | This folder contains all raw source code provided by Dr. Sherman from directory, ~ahs3/cpsc424/assignment2 |
| testEntry | This is the top level wrapper folder where user can change the block size and processor number, and submit qsub request. |
| report.pdf | The Write-up of this assignment |
| env.output | System environment output file for my develop system. |

Table : Top-tier directory

As stated above, each folder is named after its task, and more details in each task and folder will be provided in the upcoming sections of this report. In this secton, I will first look into the *testEntry* folder, its files, and the usage of its files for compiling and running the entire assignment.

In the *testEntry,* you will simply see two files. The table below shows the usage of each file.

|  |  |
| --- | --- |
| Files | Comments |
| build\_all.sh | This is the script to build all the program, and run the program on qsub by issuing qsub command. |
| variable.h | This is simply an input file for all the programs. It specifies the matrix size and number of processors for the programs. |

Table : testEntry folder

As presented in table 2, build\_all.sh is the qsub script for compiling and running the program. By default, the number of processors is set to 8 on one node, calculating matrix size of 8. Without changing anything, to submit the qub job for the assignment, one can do:

*Bash$ qsub build\_all.sh*

Once the qsub finishes running, an output file named *result.o123456* will be placed in this folder, and this the *STDOUT* of all three MPI programs required in this assignment. The result file will contain some progress prints for *make* and *qsub*. Further down the result file, for each MPI program, The *STDOUT* will contain very last element of each product matrix for verification purpose along with the calculation time and real time on each process. If the specified matrix size is less than 18, the *STDOUT* also contains the entire product matrix. Details on how to interrupt the *STDOUT* for each program will be presented in the next sections.

The *variable.h* is the file that all three MPI programs reference to get the matrix size and process numbers (MPI\_WORLD\_SIZE). I have placed it in this folder for easy access for Ronghui. The contents of this file is shown in the screen-shot below.



Figure : screen shot of variable.h

By default, the MAT\_SIZE is set to 8, and NUM\_PROCESSORS is set to 8. To change the matrix size, one can simply replace the MAT\_SIZE value in figure 1 to any number as one wish. Similarly, one can replace the 8 with any number that is the factor of the matrix size. However, since the number of processors gets changed, there is a couple changes need to be made in the *build\_all.sh,* referring to the screenshot below.









Figure : NUM\_PROCESSOR constnat dependent lines in build\_all.sh

As figure 2 shown, user need to change 8s in the lines specified in the screenshot above to same value as NUM\_PROCESSORS. Once done changing the constants, one should save the file and submit the qsub job (*Bash$ qsub build\_all.sh*). The newly generated results will be based on the new MAT\_SIZE and NUM\_PROCESSORS.

Though all three programs can run on any matrix sizes, **the number of processors has to be the factor of the matrix size.** The programs does not have error checking schemes for this requirement, so user needs to be aware of this problem.

# Task 1: Serial Program

The files in the *Serial* folder are copied directly from ~ahs3/cpsc424/assignment2. There is no major changes in the given source files. I only added printf() for the multiplication results in *serial.c*. However, the results may not be the same in my MPI programs since the values of the triangular matrices (A and B) are different between the serial program and my MPI programs due to different ways of initializing the matrices. The serial program initializes matrices, A and B, together to produce different elements in each matrix. My MPI programs initialize A and B separately to produce the same random elements for each matrix. One advantage of my method is that if I print out the entire product matrix, I can verify the product pattern is of the result of two triangular matrices (lower and upper).

# Task 2: Blocking MPI

# Task 3: Non-Blocking MPI

# Task 4: Load Balance

# Extra Credit