GMU CSI 702 Supplemental Information for HW1

*Due Date Feb 17, 2020

1 Accessing CDS 249 machines

To connect to the lab machines from your terminal:

- *ssh user@cdsxx.mesa.gmu.edu* connects you to the cdsXX machine terminal from your host computer. Use your Mason user ID. The 'xx' is the machine number - cds01, cds02,....
- *exit* or *logout* will close the connection and return you to tour terminal.

To move files between your local machine and the CDS 249 machines;

- Use *scp ${local-filepath} user@cdsxx.mesa.gmu.edu:${remote-filepath}/.* to move from your local to remote machine
- Use *scp user@cdsxx.mesa.gmu.edu:${remote-filepath} ${local-filepath}/.* to move from your remote to local machine
- Add the *-r* option if you're moving a whole directory. If you have to move multiple files, it is better to zip/tar them into a single object first before moving then.

2 Accessing Bridges

The easiest way to access the machines is to login directly with your own ssh client to *login.xsede.org* and from there textitgsissh into the correct machine. You need to set up two-factor authentication with Duo in order to use the single sign-on hub. More information on is available here on the single login system.

An example of logging on to XSEDE would be to first connect to the single sign-on hub:

- ssh XSEDE-username@login.xsede.org
- Enter your password and complete the 2-factor authentication request. Then, run the following to hop over to Bridges:
- gsissh bridges

Another way to login to XSEDE resources is via the Accounts tab in XSEDE User Portal. To reach this page login to XSEDE User Portal, navigate to MY XSEDE tab and from there select the Accounts page. All machines you have access to will have a *login* option next to them that will launch OpenSSH via a Java Applet.

Please be aware that most XSEDE resources are not available via direct ssh and all requests must go through the XSEDE single login system.

3 File transfer on Bridges

When copying files to and from Bridges, you can use ‘scp‘ in conjunction with ‘data.bridges.psc.edu‘ to avoid having to copy your files to Single Site Login node first.This will work with the Two-Factor Authentication setup.

- Try running the following to copy files directly to Bridges:

  scp -P 2222 myfile XSEDE-username@data.bridges.psc.edu:/path/to/file
- To copy from Bridges:

  scp -P 2222 XSEDE-username@data.bridges.psc.edu:/path/to/file myfile

4 Some useful Unix/Linux Commands

This is but a small sampling of commands available on a terminal on a Unix/Linux computer. This include the terminal on Mac computers. These are most of what you need for this class. Feel free to search the internet for Unix quick guides if you are interested in learning more. There are very powerful things you can do at a command line.

- **ls** - (LiSt) - directory listing (Contents of current file folder)
- **cd dir** - (Change Directory) - change to directory dir
- **cd** - change to home directory
- **cd ..** - change to directory one level up
- **pwd** - (Print Working Directory) - displays your current directory
- **mkdir dir** - (Make Directory) - creates a folder in current directory named dir
- **rm file** - (ReMove) - deletes file file from current directory
- **rm -rf dir** - (ReMove -Recursive Force) - deletes folder dir and all of its contents and subdirectories. ***Use this one with caution.***
- **man command** - (MANual) - Displays a text manual of that command. Use up and down arrows to scroll through text. Hit 'q' to quit viewing and go back to terminal.
- **which command** - Displays path (folder) to the location where the command or program is located. If not found, gives message on where it looked.

5 Compiling and submitting codes

5.1 Compiling and testing

When testing the code on a local computer or within an interactive session on Bridges, you will use a simple command-line interface to select which benchmark you want to run.

To compile your code, just use the *make* command.

Using the HW1 as an example, There are three available benchmark modes: "naive", "block", and "blas". The "block" mode is the default.

To benchmark the "naive" mode, you would type:

./benchmark naive

To benchmark the "blas" mode, you would type:

./benchmark blas

The benchmark results will both print to stdout and be written to a CSV file named 'benchmark_MODE.csv' by default (replace 'MODE' with the benchmarking mode). If you want to change the name of the output file, use the '-o' flag, for example:

benchmark -o benchmark1.csv blocked

To read about other features you can set on the command line, such as changing the "max speed" variable (for example, if you want to benchmark on your own computer), run:

```
./benchmark --help
```

5.2 Submitting and running

The jobs queue on Bridges is managed via the SLURM scheduler.

To submit a job, use the 'sbatch' command like so:

sbatch job-blocked

To check the status of your running jobs you can use the following command:

squeue -u $USER

Append a '-l' flag will print additional information about the running jobs. If you want even more information, consider using the 'sacct' command, for example:

sacct -j $JOBID –format JobID,ReqMem,MaxRSS,TotalCPU,State where '$JOBID' is the ID number of the job.

If you want to cancel a job, run:

scancel $JOBID

If you would like to receive emails for job submissions add the following lines to the submission scripts.This sometimes helps tracking down issues.

#SBATCH –mail-type=ALL #SBATCH –mail-user=youremailaddress

For more details on SLURM commands please see Bridges documentation page.

Finally, there is an interactive mode that allows you to request computing nodes, but maintain a command line. This is ideal for prototyping and debugging purposes. To activate this mode, type

interact -N 1

For additional information, read the documentation on interactive sessions.

6 Report

In addition to your code, you are to submit an accompanying report for the homework assignment. The report is to be written in a style appropriate for an academic journal with any relevant citations provided in a bibliography. The report should only discuss the details of the final version of your submitted code. The report is to contain the following:

1. An introduction to the problem
2. A section detailing how you optimized the code. This requires both a conceptual discussion of what you implemented (figures are helpful!), and a code summary, where you explain each important step in your solution (this should be supported with code snippets).
3. A benchmark section where you show how your code performs relative to the naive and blas implementations. Figures are very helpful when reporting this data! Benchmark data should be taken from runs performed on Bridges, not your local computer or laptop. Make sure you discuss the reason for any odd behavior (e.g., dips) in the reported performance.

Your report should be converted to the PDF format prior to submission. The filename should follow this format: 'FirstNameLastName_hw1.pdf'.

7 Grading

Your final code grade will be determined by comparing your code's sustained performance relative to the Intel MKL implementation (i.e. the "blas" running mode). A quicker estimate of your code grade is outputted at the conclusion of each benchmarking run. It is computed by comparing your code's sustained performance on Bridges as a percentage of the theoretical peak performance. From that estimate, you will receive a score as follows:

- If your sustained performance is between 0% and 25% you will receive a score between 0 and 65 proportional to your sustained performance (Ex: 20% gives a score of 52)
- If your sustained performance is between 25% and 40% you will receive a score between 65 and 80 proportional to your sustained performance (Ex: 35% gives a score of 75)
- If your sustained performance is between 40% and 70% you will receive a score between 80 and 100 proportional

to your sustained performance (Ex: 60% gives a score of 93)

- If your sustained performance is above 70% you will receive 100

VERY IMPORTANT! Your submission must pass the error bound test and **cannot** call BLAS for dgemm; any submissions that fail these tests this will receive a grade of 0. For this reason, do not change anything in the 'benchmark.c' file, otherwise you might deactivate this check and not realize your code is not behaving properly.

8 Useful references

Here are some references that you can use to get started.

1. Goto, K., and van de Geijn, R. A. 2008. Anatomy of High-Performance Matrix Multiplication, ACM Transactions on Mathematical Software 34, 3, Article 12. *Note: explains the design decisions for the GotoBLAS 'dgemm' implementation, which also apply to your code.*

2. Chellappa, S., Franchetti, F., and Püschel, M. 2008. How To Write Fast Numerical Code: A Small Introduction, Lecture Notes in Computer Science 5235, 196-259. *Note: how to write C code for modern compilers and memory hierarchies, so that it runs fast. Recommended reading, especially for newcomers to code optimization.*

3. Bilmes, et. al.* The PHiPAC (Portable High Performance ANSI C) Page for BLAS3 Compatible Fast Matrix Matrix Multiply.*Note: PHiPAC is a code-generating autotuner for matmul that started as a submission for this HW in a previous semester of CS267.*

4. Lam, M. S., Rothberg, E. E, and Wolf, M. E. 1991. The Cache Performance and Optimization of Blocked Algorithms, ASPLOS'91, 63-74. *Note: clearly explains cache blocking, supported with performance models.*

5. A short video by Brian Van Straalen.