Lab 4 - Matrix-Matrix Multiply and  
Third Party Libraries

NERS/ENGR 570 Fall 2020

September 25th, 2020

DUE BY: Oct. 5th, 2020

# **Exercise 1 (100 pts) - BLAS**

1. Clone this Lab’s git repository:
   1. From the terminal command line:

git clone https://github.com/bkochuna/Lab4-matmul.git

1. Read through the sample code we provided.
2. Read through the rest of this exercise and make a prediction about which implementation is fastest and how the run times will scale with the matrix size.
3. The Great Lakes cluster has a local installation of BLAS (/usr/lib/blas). Compile and link your program against this version of BLAS using one of the GNU compilers. Record the command used for compiling (don’t forget optimizations!):
4. On Great Lakes there is also a module for a different BLAS implementation called OpenBLAS. Load this module and any other required modules. Then recompile the program against this version of BLAS. Record the command used for compiling:
5. Run your executable for each BLAS implementation for the following matrix sizes and record the results. For consistency in the results, all results should be generated ***on the compute nodes.***

|  |  |  |
| --- | --- | --- |
| Matrix Size | BLAS dgemm time (s) | OpenBLAS dgemm time(s) |
| 20 |  |  |
| 100 |  |  |
| 500 |  |  |
| 1000 |  |  |
| 2000 |  |  |
| 4000 |  |  |

1. Provide your own dgemm implementation in either the Fortran or C++ program. Uncomment the code to compare the C matrix computed by your code to the C matrix computed by the BLAS library.  
   1. Again run your program for the various matrix sizes ***on the compute nodes***. Compare the run times.

|  |  |
| --- | --- |
| Matrix Size | My MatMult time (s) |
| 20 |  |
| 100 |  |
| 500 |  |
| 1000 |  |
| 2000 |  |
| 4000 |  |

1. Write a similar matrix-matrix multiply program in Python using the numpy/scipy libraries and generate the same set of results as the other cases.

|  |  |
| --- | --- |
| Matrix Size | Numpy dgemm time (s) |
| 20 |  |
| 100 |  |
| 500 |  |
| 1000 |  |
| 2000 |  |
| 4000 |  |

1. Questions
   1. Which BLAS implementation was fastest?
   2. How did each BLAS implementation scale with matrix size?
   3. Did the speed of each implementation agree with your prediction?
   4. If you observed differences from your prediction why?
   5. For your implementation, change the order of the loops that are executed and  
       rerun. Describe what you observe? Explain what might be going on.

*Extra Credit (****5 pts****)*

Install the ATLAS Library and compare this with your previously generated results.  
[Automatically Tuned Linear Algebra Software (ATLAS)](http://math-atlas.sourceforge.net/)

*Extra Credit (****5 pts****)*

Take the Fortran or C++ program and link it to the MKL library (using the GNU compilers). Generate results for this BLAS implementation, and compare to the other results

**Deliverables and how to submit:**

For the overall deliverable you may submit a single word doc or pdf based on these instructions. Name the file as follows:

<username>\_Lab4.doc (or docx or pdf).

For exercise 1, include the commands you executed for each part of the exercise for compiling. Also include one version of your Slurm script used to generate the results. You will run several cases and this may necessitate several Slurm files. Just provide one. For your naive implementation of matrix-matrix multiply include the code snippet for this routine. For your python code, copy the source into this word document as well (or use the listings environment in LaTeX if you’re so inclined).

Include answers for all questions.

Please upload your doc/docx/pdf file to canvas.

For the Extra-Credit include this in your submission with this document.