Homework 3
Matrix-Matrix Multiplication on GPU
CUDA and OpenACC

In assignment 1, we looked at improving Matrix-Matrix Multiplication using SIMD via AVX on a single core. We noted how difficult of a problem this was. In assignment 2, we seen how fast we could get improvements on a complex problem using OpenMP, and how much work had to go into using MPI. In this assignment, we want to see how GPU computing changes the level of difficulty to get improvements. Matrix-Matrix Multiplication is one of the key algorithms they demonstrate on GPU computing because it is computational bound and not memory bound. We can think of a GPU a processor with huge SIMD lanes (really a vector processor). However, because the fact that it is a device attached away from the processor and the unique structure it can be very hard to code on GPUs. You will explore two language extensions: CUDA and OpenACC. CUDA is lower level, and you have to so most of the system work. OpenACC is compiler based and is similar to OpenMP. Below are some key ideal to get you started.

0. You must keep note on all your design decisions and the impact on execution time these decisions had.
1. Timing on a GPU is hard! In this case we will also be using a shared GPU. Make sure you experiment to make sure your timings make sense, and run the same experiment multiple times.
2. You will need: module load cuda for the cuda compiler nvcc
3. You will need: module load pgi for the openacc compiler by the Portland Group
4. You will need to use blocking to get good performance. (Remember only threads within a block can sync/talk to each other) Blocks can be executed in any order, and there is a fix size to the shared memory.
5. You DO NOT NEED to use grids with cuda. You can do it in 1D layout, put you might find it easier to use grids.
6. There are two references on Canvas. One that talks about implementing matrix-matrix multiplication on CUDA and other a reference to OpenACC commands.
7. Remember threads should be a multiple of 32. (64/128/256)
8. General outline of how to work
    a. Get a crude block version working with CUDA with blocks
    b. Get a crude block version working with OpenAcc with blocks
    c. Make sure the time seem to make sense
    d. Tune your CUDA code (this may be easier then OpenAcc tuning)
    e. Tune your OpenAcc code