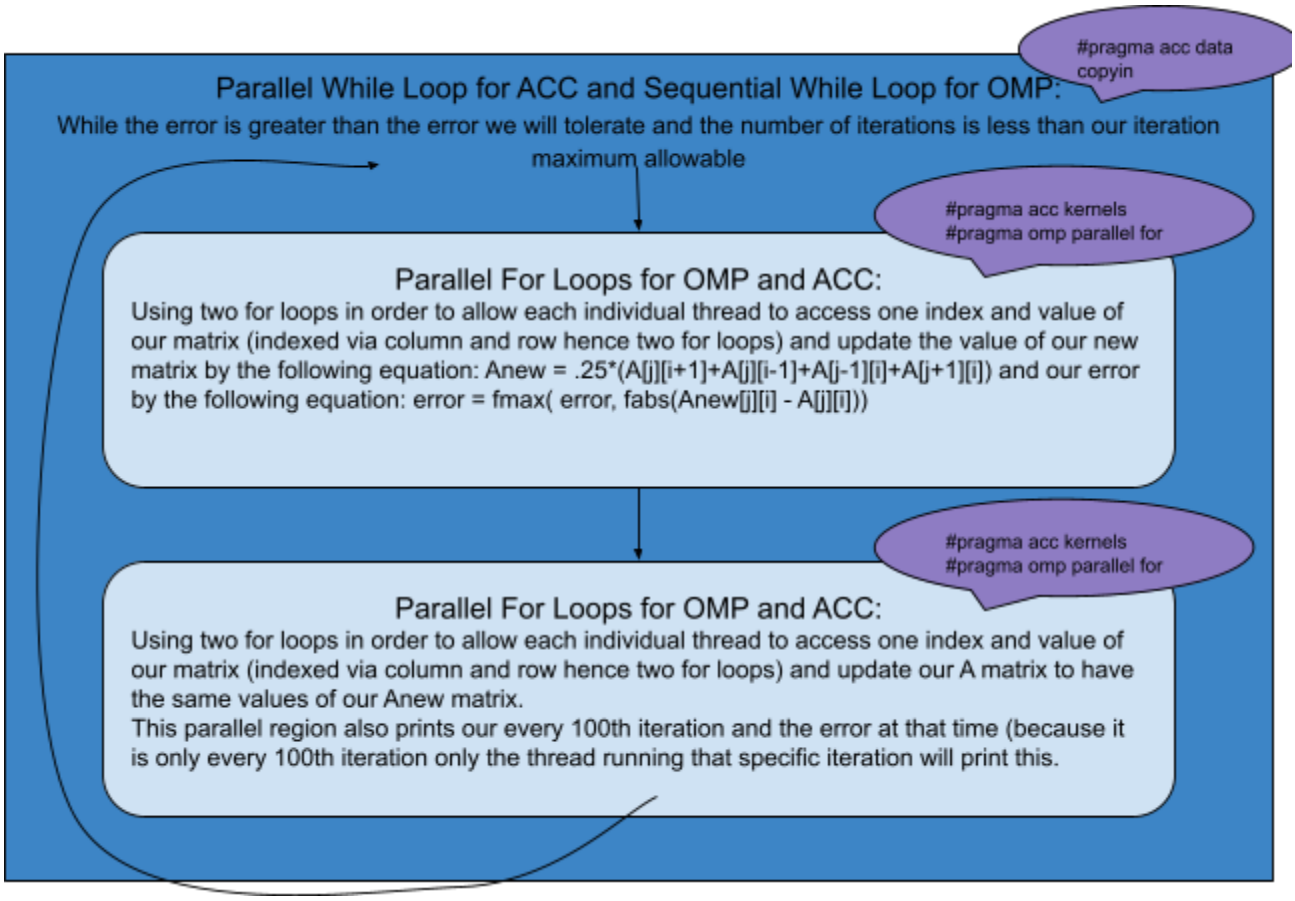
Kailee Madden
HW 3 Submission

I. Flowchart and Parallel Region Descriptions

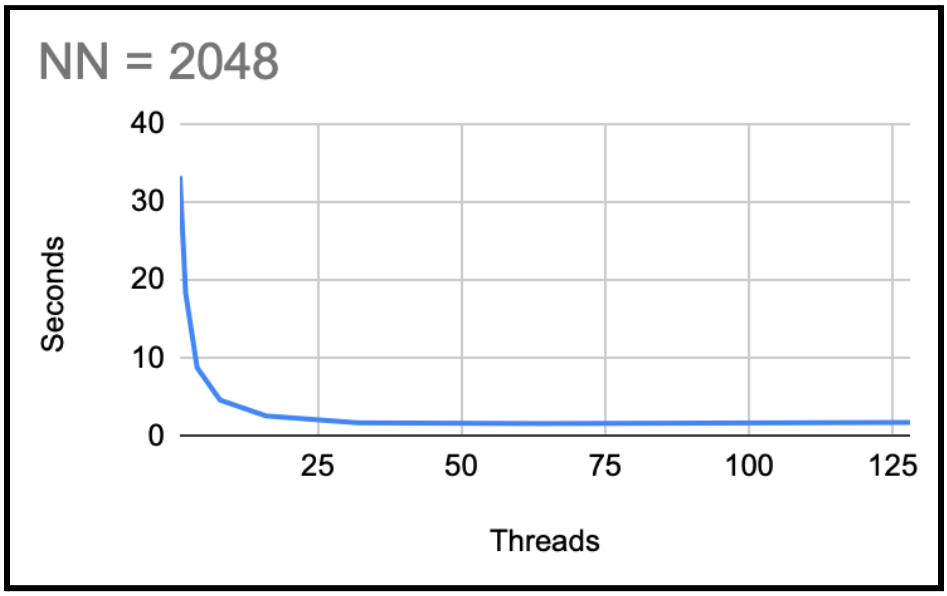
Note: outside the primary loops, there is a private parallel region that initializes our A matrix and prints the number of threads being used.
Flowchart for primary program:

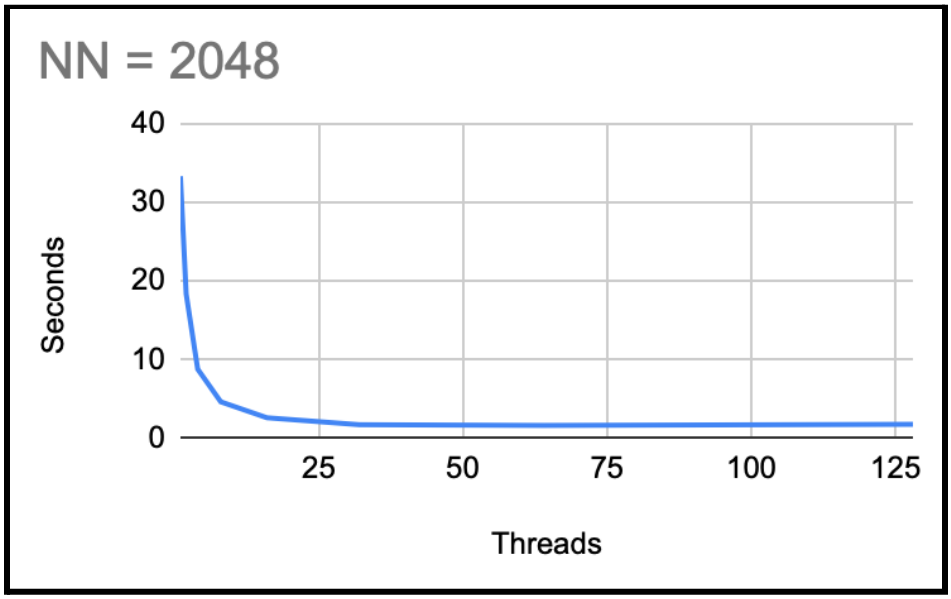


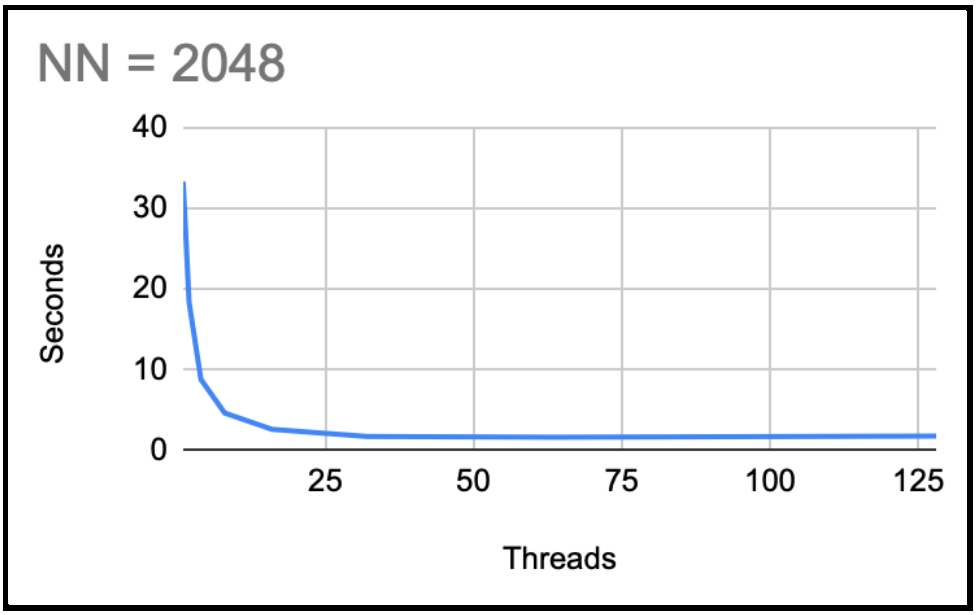
II. Runtime Plots for OpenMP and OpenACC

Save the batch script/interactive submission, tabulate and plot the runtime in the three series of tests and discuss what you have found.

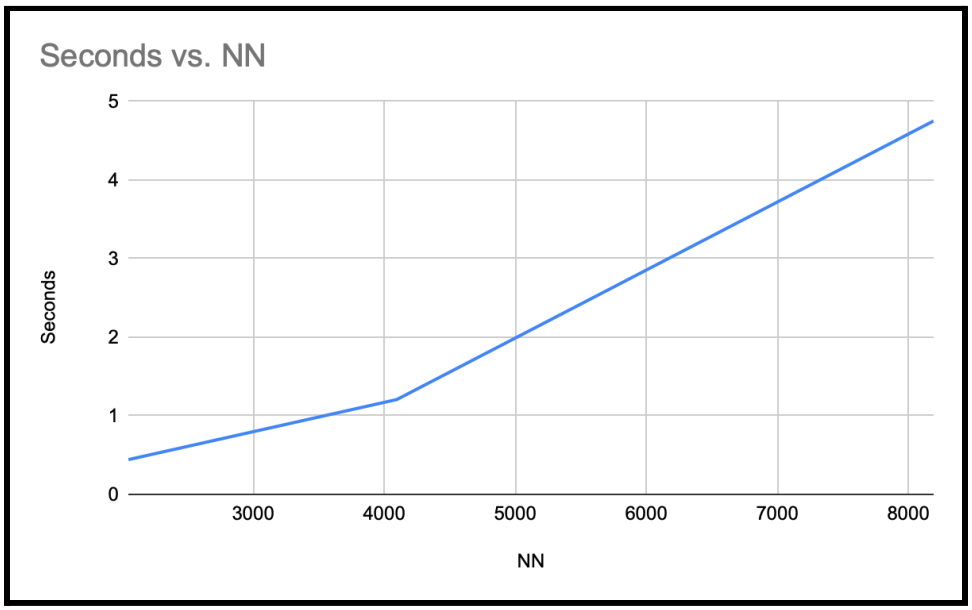
OpenMP


NN = 2048
40
30
20
10
0
Seconds
25
50
75
100
125
Threads


NN = 2048
40
30
20
10
0
Seconds
25
50
75
100
125
Threads



OpenACC



III. Implementation of CUDA

- Extensive comments are available in the actual code for the CUDA implementation to describe CUDA operations such as memory copies and kernel launches
- Output files are available for the three mesh sizes of 2048, 4096, and 8192 in the cuda_output directory

- Pointers were used for sharing memory (specifically for the matrices) across device and host
- Rather than using an array of arrays to represent the matrix, I used one array and collapsed the 2d into 1d (ex. A[row][col] = A[row*colmax +col])
- The main challenge was ensuring the correct communication between device and host, and as part of that using the pointers most effectively

IV. Runtime Plot for CUDA

The CUDA implementation is significantly faster than the OpenACC implementation. My understanding is that this is expected since the PGI compiler needs to translate OpenACC kernels into object code, whereas CUDA codes are already kernels and thus can be directly run.