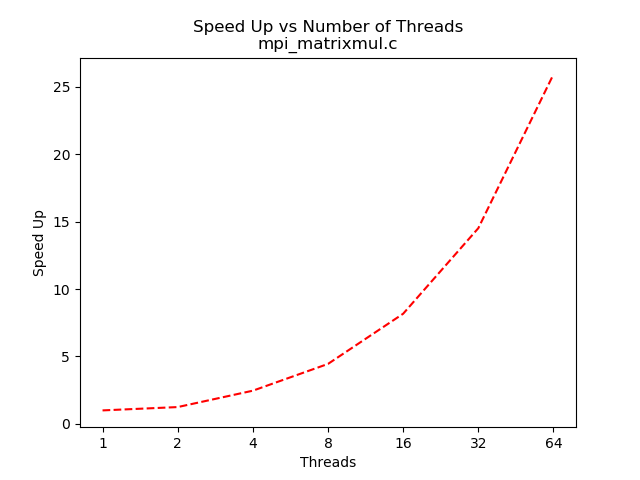
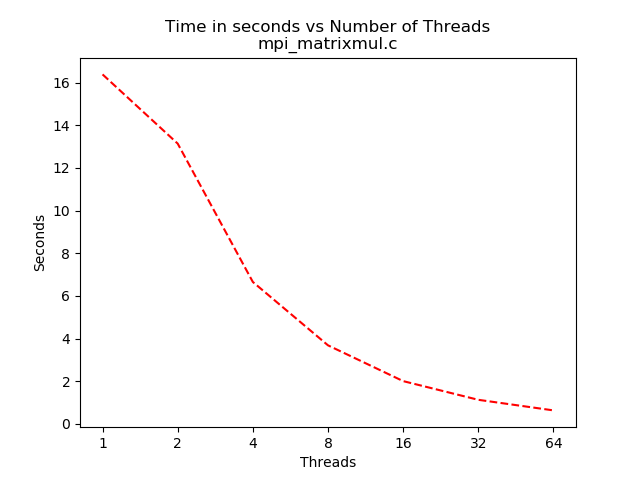
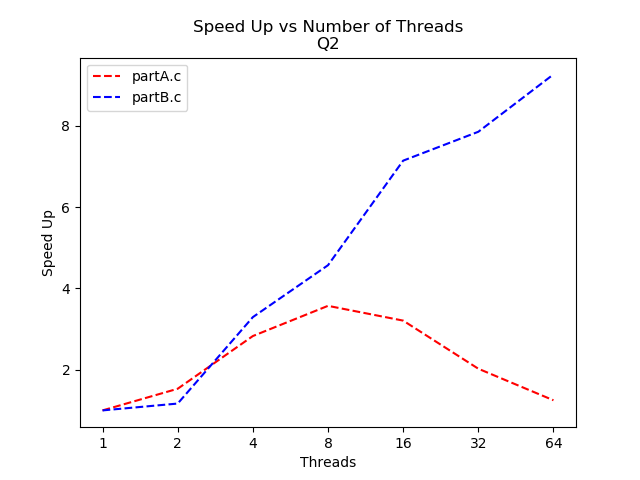
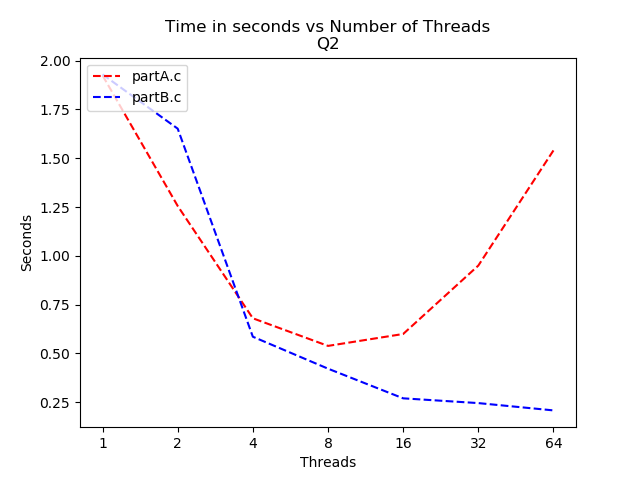
CS4379 Project 4 Brief Report  
Jacob Zahn  
Professor Yong Chen  
Computer Science, Texas Tech University

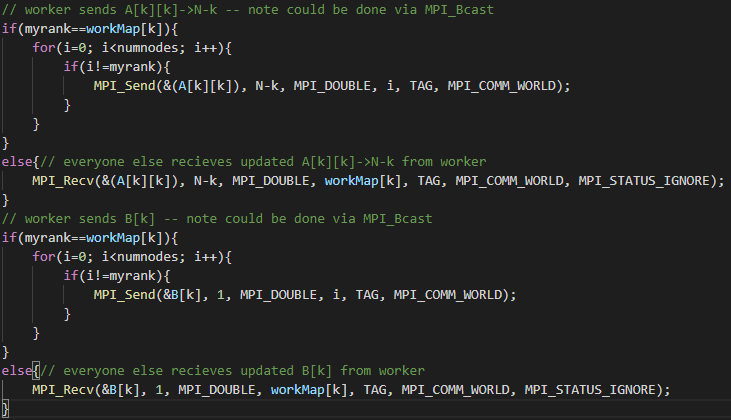


Above is the time and speed up charts for the mpi\_matrixmul.c program.

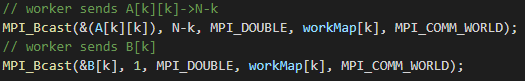


Above is the time and speed up graphs for partA.c and partB.c. The MPI point to point communication version of Gaussian elimination, partA.c did not scale well past 8 threads. Speed up is sublinear on both partA.c and partB.c, but partB.c manages to still reduce the execution time all the way to 64 threads. The major change between partA.c and partB.c are the communication calls within the work loop.

Code for partA.c



Code for partB.c



The for loops in partA.c are to implement the point to point sends, and as such are N many loops. This one by one communication severely limits the scalability of the algorithm and stops MPI from taking advantage of the network topology of the environment. Alternatively, partB.c’s MPI\_Bcast() calls can leverage the network topology of the environment and lower overall communication time and the amount of time every non worker spends blocked waiting for a message.

MPI\_Bcast() also allow the programmer to lower the complexity of the code used to communicate, swapping out two for loops and many if/else blocks for two API calls.