How was the initial data distribution done?

For my data distribution, I used 1D decomposition and divided by rows. Each process has n/p elements with (Y_limit / p) number of rows. To help visualize the data and keep all of the neighbors near each other I chose to do it in a grid. Technically, it is faster to store all of the elements in a process sequentially (together rather than using pointers), however, I thought it would help me keep track of all of the rows and columns.

How did you ensure that the blocking version does not deadlock?

To ensure that the blocking version does not commit a deadlock I swapped the order of send and receive for the different processes. The only send and receive that happens between two processes is for the neighboring process so it was easy to split the processes by even and odds. The odd ones send first while the even ones receive first. This keeps the timing staggered. We can do this because MPI_Scatter divides the data by processes sequentially (for example process 2 only needs to communicate with processes 1 and 2). The order of send and receive will help prevent this deadlock.

What are the performance results, and are they what you expected

I couldn't get my code to work so I wasn't able to calculate my performance or speedup. I decided to submit my code for GFA reasons. I tried my best, but I keep getting seg fault errors. Everything compiles and this is what I think an implementation would look like. I will be studying C a lot tomorrow so I finish assignment 2 early, because wow I really need a refresher.