Brian Crafton
HW 4

4. Each worker determines its indexes by multiplying its index (minus 1) by the number of elements given to each process. Each process will get (total array size / number of processes) elements it needs to work on. The end index is just the index multiplied by the number of elements allocated to each process. This way each element works on the same number of elements.

6.

problem	problem size	run time
q2	10^3	2.24E-04
q2	10^5	0.0045
q2	10^7	5.0842
q3	10^3	0.9717
q3	10^5	0.7293
q3	10^7	3.9662
q4	10^3	1.3184
q4	10^5	0.8246
q4	10^7	5.2758
q5	10^3	7.1911
q5	10^5	6.0241
q5	10^7	10.1257

Looking at the results it seems as though the arrays were not large enough to see a speedup in the parallel code. Problems q3 and q4 started to see a speedup as the size of the problem approached 10^7.

7. Programming in the way would be most effective in situations where a systolic array would be effective. So whenever you want to do multiple computations on a single piece of data before returning it to memory. If this was pipe lined you could load whatever amount from memory is able to be loaded from memory at one time and then divide up the problem in chunks pipelining each chunk that is able to be pulled from memory. This way each unit does its computation on the data before having to return it to memory.

8. Unfortunately I could not get my scripts in the folder to actually execute for the number of nodes that were requested. They got hung up on the cluster and would not execute.

9. I also could not get a gpu script to run. I have still written the code, and I imagine the problem size does need to be a decent size before the GPU is able to execute faster than the CPU.