Michael Laucella
Parallel Processing
Lab 2

Number of Threads	Run Time (s)	Size
1	.000060	10,000
2	.000141	10,000
5	.000267	10,000
10	.002131	10,000
20	.006513	10,000
40	.003452	10,000
80	.002754	10,000
100	.003540	10,000
1	.005578	1,000,000
2	.005448	1,000,000
5	.004888	1,000,000
10	.006257	1,000,000
20	.006037	1,000,000
40	.009175	1,000,000
80	.007614	1,000,000
100	.009133	1,000,000

The run times observed show that with a size of 10,000 using a cyclic 4 schedule that no speedup has been gained from the increasing of threads in fact the opposite has occurred. There are a few areas of influence here, one the size is not very high as computation with one thread took barely any time at all, two the overhead for thread creation for a size that did not scale with it, three the scheduling algorithm used. I primarily choose cyclic because the problem gets less intensive as I increases with I=3 being the most intensive part. This should have relieved some load imbalance better than dynamic which would have given the worst half to one thread and then kept handing out the rest. The chunk size of 4 was to not spend too much time

handling the load balance by distributing 4 at a time. And the last factor was caching, by sharing the entire array any updates can lead to cache coherence protocols when modifying the same blocks, also false sharing when certain updates might not affect the computation of the other threads but can cause cache coherence protocols that update the block.

With the size of 1,000,000 we still face all the same issues listed above, however there is a speed up from 2-5 threads this is likely the problem size scaling better for the number of threads being used.