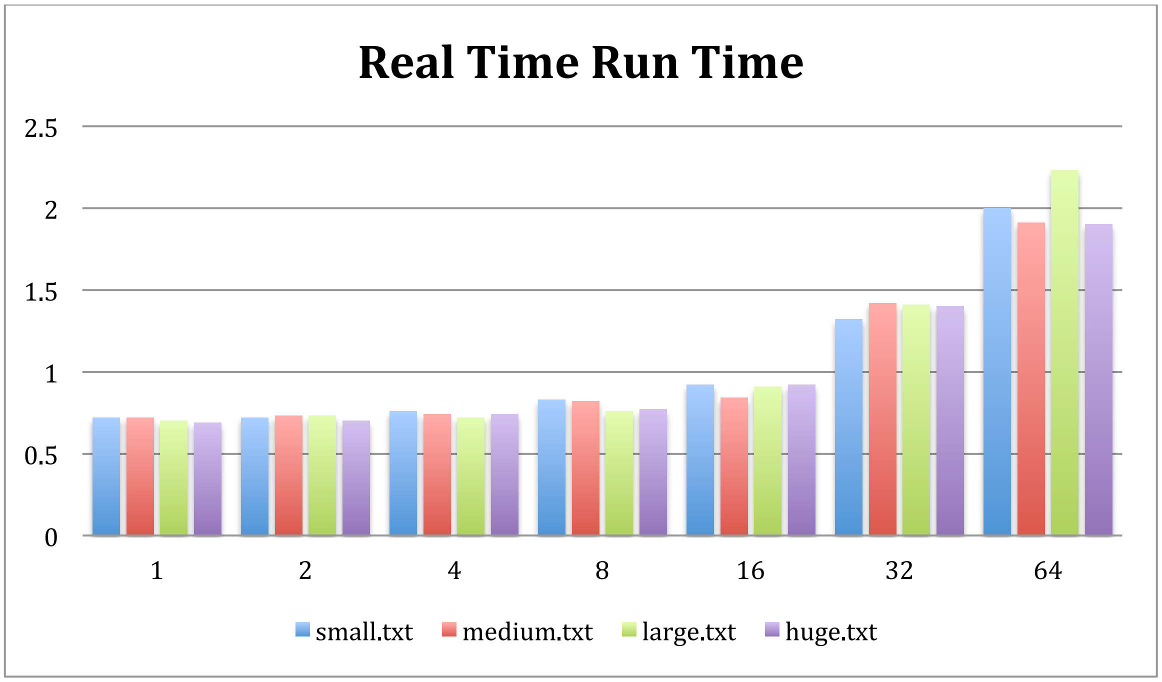
Connie Shi
Parallel Computing Lab 1

Part 1:

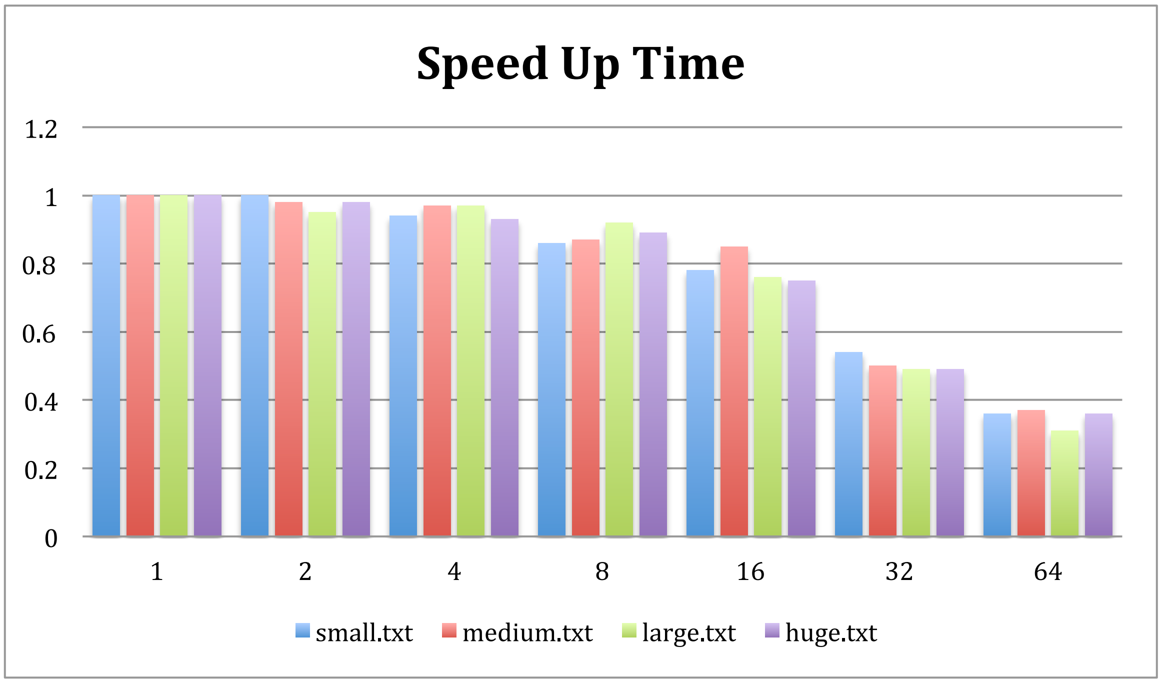
File	1	2	4	8	16	32	64
small.txt	0.72	0.72	0.76	0.83	0.92	1.32	2.00
medium.txt	0.72	0.73	0.74	0.82	0.84	1.42	1.91
large.txt	0.70	0.73	0.72	0.76	0.91	1.41	2.23
huge.txt	0.69	0.70	0.74	0.77	0.92	1.40	1.90



Part 2:
Speedup = T1/Tp = Tserial/Tparallelcores

File	1	2	4	8	16	32	64
small.txt	1	1	0.94	0.86	0.78	0.54	0.36
medium.txt	1	0.98	0.97	0.87	0.85	0.50	0.37
large.txt	1	0.95	0.97	0.92	0.76	0.49	0.31
huge.txt	1	0.98	0.93	0.89	0.75	0.49	0.36

Connie Shi
Parallel Computing Lab 1



Conclusions:

On all given files, 64 processes take the longest to run. This is because the files are rather small (huge.txt only has 50 equations) so in all test cases, there are many processes that do not do anything meaningful that contribute to the solving of the equations. In my program, I simply gave the leftover processes (that do not get equations to compute) 0's. Thus, the overhead of creating processes and inter-process communication (along with associated block time) is greater the iterative version. All test cases perform roughly the same with the same number of processes – I suspect this is because the test cases are very small so we cannot detect the difference in parallelizing. Of course, the lack of true perceptible scalability might be caused by the implementation of my code. In two places, I used broadcast instead of send/scatter because it was easier to implement. Because all collective calls block, my implementation may not be the best way to parallelize the program.

As for the reason the program decreases in speed up at every increased use of processes is obvious in the design of the lab. Every *a* value gets scattered, which is a collective call. Every ***b* and *aii*** value gets broadcasted – also a collective call. Allgatherv to retrieve the calculated values is also a collective call. Add in the overhead of creating processes and synchronizing their run (MPI_Barrier), we get even more added delays. In huge.txt, the most computation that occurs per equation is roughly 50*2 (multiply a by x and subtract, 50 times). Because modern computers can do roughly 1 million computation in a second, ~100 computations is negligible in timing. In other words, in this lab, we did not use parallel programming to its full functionality.