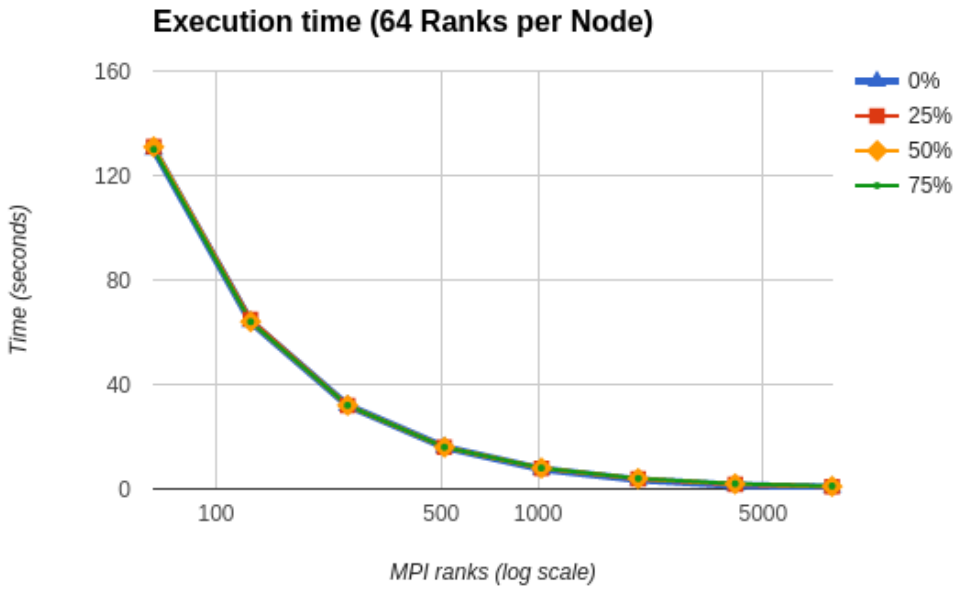
Benjamin Mizera
Parallel Computing
March 11th, 2016

Homework 3:

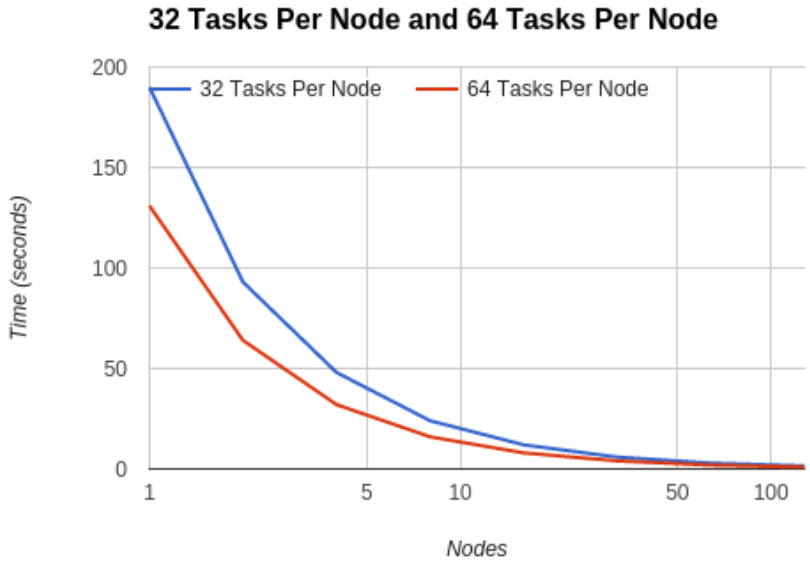
For Homework 3, we were asked to execute our “Game of Life” code on RPI’s BlueGene Q system, AMOS. There we would simulate a 67108864 cell universe for 100 ticks at varying degrees of randomness. Below you can see my plotted results from the first experiment, along with the stated runtimes in a table.



64 proc per node				
Rank #	0%	25%	50%	75%
64	130	131	131	130
128	64	65	64	64
256	32	32	32	32
512	16	16	16	16
1024	7.6	7.9	8	8
2048	3.5	3.8	4	4
4096	1.4	1.7	1.9	2
8192	0.66	0.8	0.95	1.1

As you can see, this programming is extremely strong scaling, and provided the ample resources available on AMOS, can be run insanely fast even at the 6 million data point mark. I was extremely surprised that as the RNG threshold increased that the runtime went up. My only guess is that the calls to the RNG library were rather expensive, considering it was slower than the time it took to update each cell and communicate the border cells' status.

As for the greatest speedup, if we look at the 0% threshold, the greatest speed up occurs between the 2048 and 4096 task counts, where there is a 2.4x speedup. Most other cases it's around 2x speedup for the 0% threshold, if not slightly above. Looking at the higher threshold cases, the speedup is almost always right around 2x, which is what we would expect from a strongly parallel program, however, we rarely see it due to overhead costs. Realistically, I'm not entirely sure why the largest speedup occurs there, that count isn't hitting a particular breaking point for Blue Gene. It might simply be random, and if I ran this ten times, I might see different results. However, supercomputer time is expensive, so I'm going to leave it as is.



Nodes	32 Tasks Per Node	64 Tasks Per Node
1	190	131
2	93	64
4	48	32
8	24	16
16	12	8
32	5.9	4
64	2.9	1.9
128	1.5	0.95

As for comparing the 32 and 64 ranks per node case, it is rather interesting that we do not see complete obliteration from the 64 tasks per node, considering that it’s running twice as many MPI ranks as the 32. The limiting factor has to come down to overhead costs within the system for transferring memory or something of that nature, considering that each node has the ability to run 64 threads independently (16 4 way cores). In each case, the speedup gained by doubling the task count is only about 1.5x, and while that is significant, it’s not exactly what one might hope for.