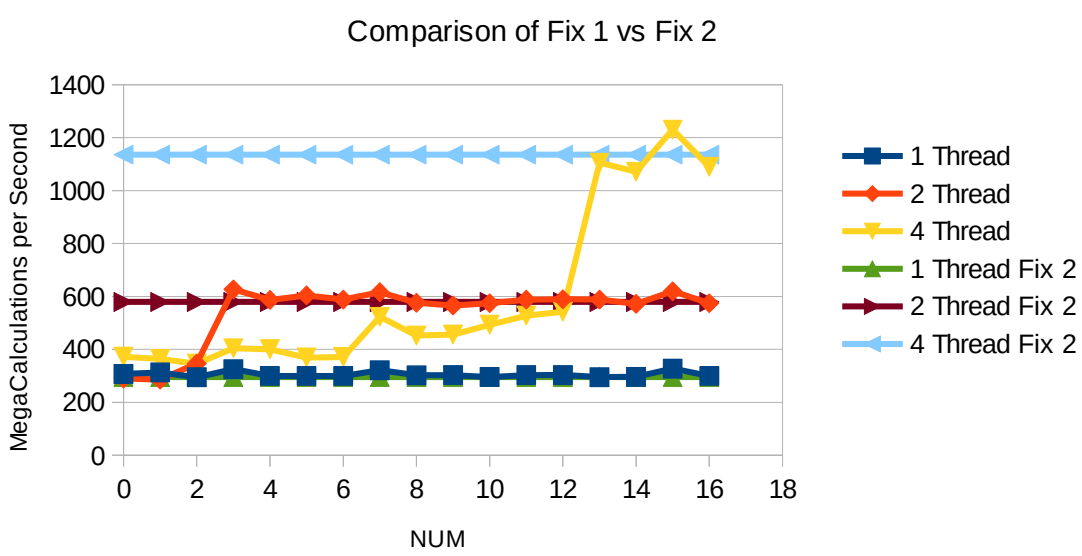
Travis Robinson
CS474
Spring 2016
Project 3
False Sharing

This machine was run on the flip servers at Oregon State.

Data Results

False Sharing
Fix 1 vs Fix 2

	1 Thread	2 Thread	4 Thread
0	307.29	290.11	372.51
1	312.75	285.89	363.6
2	294.89	346.75	346.28
3	325.17	626.79	404.38
4	299.67	587.17	400.39
5	299.5	604.21	369.26
6	299.62	588.57	371.15
7	321.22	616.34	523.1
8	302.43	576.5	452.32
9	302.62	566	456.05
10	295.69	574.58	493.18
11	302.74	587.97	527.83
12	303.26	589.32	542.59
13	295.33	588.44	1105.45
14	296.25	571.98	1071.14
15	327.68	619.65	1230.25
16	299.67	574.59	1090.52
Fix 2	295.72	579.29	1135.67



Patterns and Their Causes

What we see from looking at this data is that Fix #2 usually gives us better results than Fix #1. The exception to this is when using a padding value of 15, where Fix #1 gives slightly better results. I suspect this is due to the time lost declaring and initializing the private variables needed for Fix #2.

We also see that for low padding values, the number of threads doesn't seem to make a lot of difference. When using two threads, performance isn't too different from one thread for low padding numbers, and with four it starts off around one-thread performance, then increases to two-thread performance before jumping up to its higher performace levels. The reason for this is that there's not enough padding to create cache lines to match the number of threads, so that there are threads being idle because they have no data to work on. In essence, even if there are four threads, if the data is all on a single cache line, there's still only one thread doing any work.