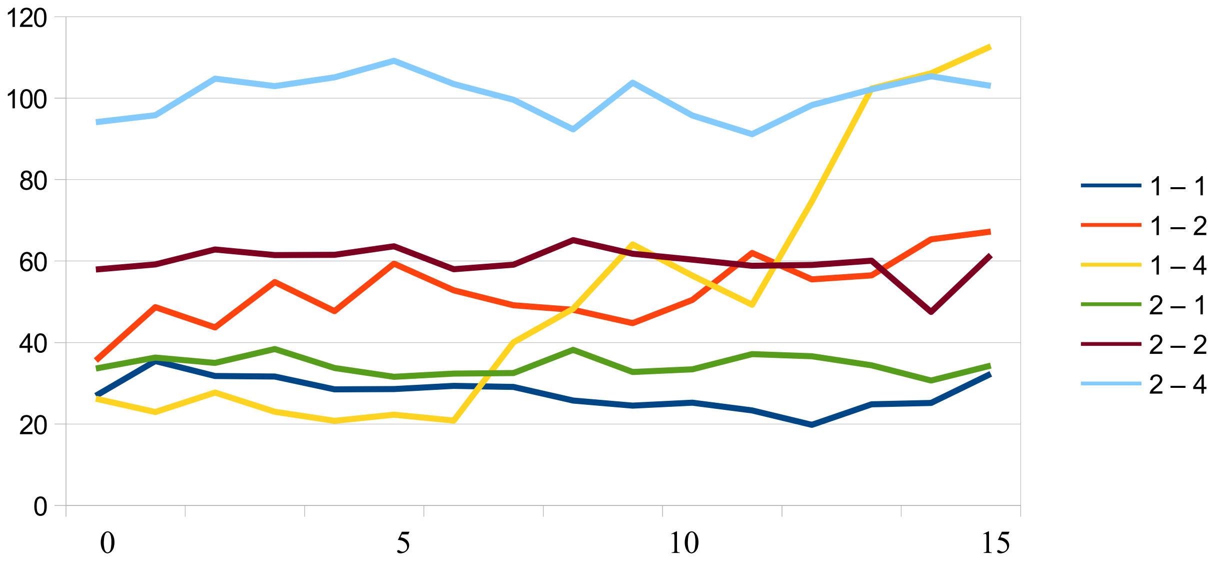
Dylan Washburne
Project 3

This was run on the ENGR flip3 server, which is a very powerful machine machine.

	Fix 1			Fix 2		
	1 – 1	1 – 2	1 – 4	2 – 1	2 – 2	2 – 4
0	26.91	35.58	26.19	33.63	57.93	94.09
1	35.42	48.7	22.92	36.3	59.18	95.83
2	31.78	43.72	27.71	34.97	62.82	104.8
3	31.64	54.83	23.01	38.41	61.45	102.92
4	28.52	47.71	20.78	33.76	61.5	105.12
5	28.57	59.35	22.26	31.58	63.65	109.16
6	29.34	52.82	20.82	32.36	58.01	103.45
7	29.1	49.12	40.05	32.48	59.12	99.59
8	25.76	47.99	48.28	38.23	65.1	92.33
9	24.53	44.74	64.1	32.79	61.8	103.77
10	25.21	50.42	56.42	33.44	60.34	95.75
11	23.34	62.01	49.3	37.15	58.86	91.15
12	19.8	55.47	74.63	36.61	59.03	98.29
13	24.8	56.48	102.34	34.41	60.07	102.19
14	25.17	65.34	106.11	30.68	47.49	105.33
15	32.32	67.22	112.73	34.35	61.45	103.03

With all measurements in MegaIncr/sec:



In general, I am seeing the padding start lower, even dipping a bit, and then jumping up to become much more efficient, especially at higher thread counts. Private variables (fix 2) on the other hand remain consistent in their processing speed, within standard deviation, which I honestly account on the server's speed being somewhat inconsistent as the tests were run.

I believe this to be because at too few trials the false sharing has not kicked in, a few trials in it dips as false sharing slows it down, and then the fixes start to kick in, bringing its performance up. Private variables on the other hand are consistent by their basic function, and ideally should be an almost flat line for performance with that many threads.