Aashutosh Y Taikar
OSU ID:933020005

Project #3 False Sharing

1. Tell what machine you ran this on
   NOME server OSU: nome.eecs.oregonstate.edu

2. Create a table with your results.
   Performance in megamults/sec of NUMPAD vs. Threads

   Table for Fix#1:

Threads→

↓ NUMPAD	1	2	4	8
0	147.35	210.74	203.4	206.82
1	145.66	210.73	200.61	204.66
2	141.22	211.73	211.74	212.69
3	145.7	216.56	220.04	220.39
4	145.34	221.13	222.06	230.95
5	142.92	236.12	287.77	248.9
6	145.03	247.11	295.31	279.84
7	147.13	287.84	302.56	337.35
8	145.47	287.19	320.96	300.1
9	147.63	286.92	298.94	293.36
10	137.05	260.47	290.38	313.96
11	144.23	281.13	309.56	334.14
12	135.84	264.49	283.28	300.59
13	146.61	287	297.1	354.04
14	145.03	284.97	315.67	361.57
15	147.27	289.14	387.76	417.06

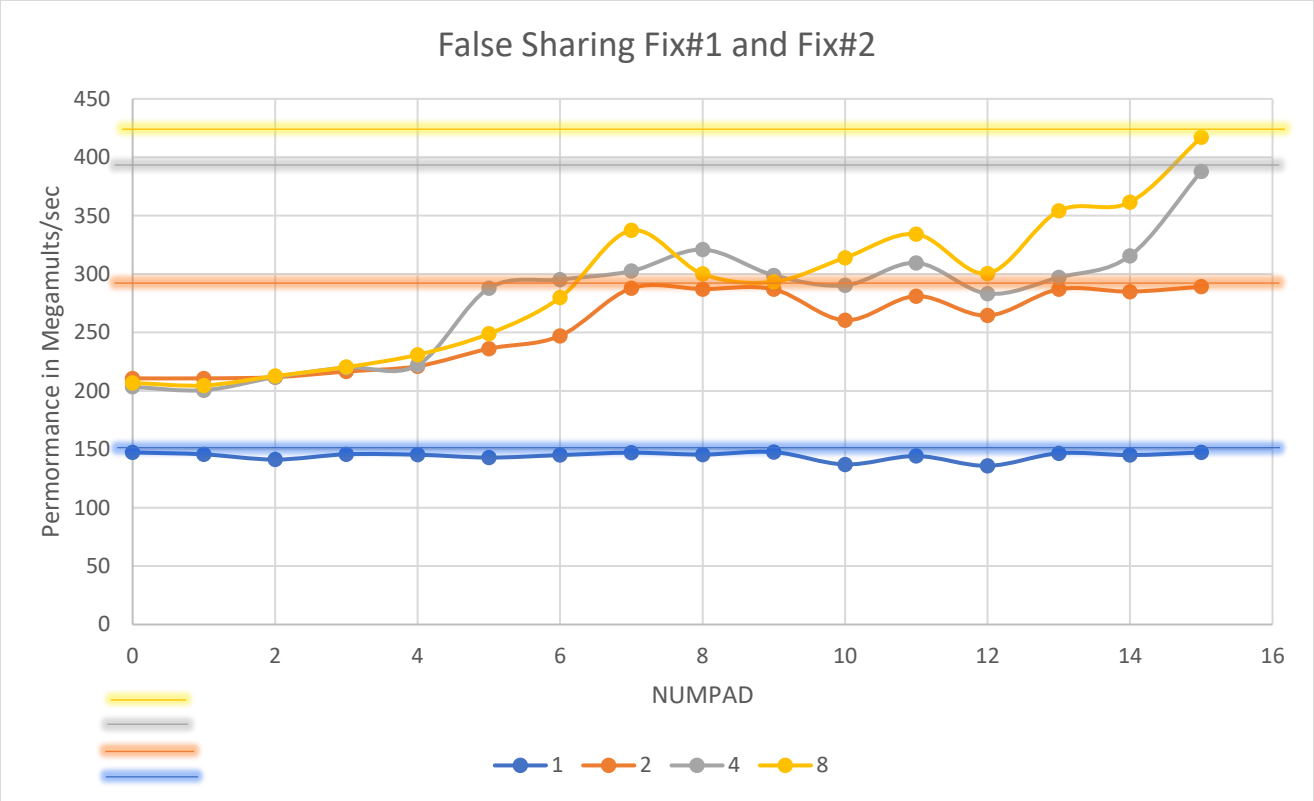
Table for Fix#2:

Threads	Performance
1	151.01
2	290.99
3	398.94
4	421.38

nome01 ~ 153% uptime

20:29:51 up 42 days, 2:00, 9 users, load average: 2.48, 1.65, 1.23

3.Draw a graph. The X axis will be NUMPAD, i.e., the amount of integers used to pad the structure. The Y axis will be the performance in whatever units you sensibly choose. There should be at least ***6 curves*** shown together on those axes:
1-3: Using padding with 1, 2, 4 and 8 threads. (Fix #1)
4-6: Using a private variable with 1, 2, 4 and 8 threads. (Fix #2)



Fix#2 1
2
4
8

3. What patterns are you seeing in the performance?
    - After running the fix#1 on a single thread over a range of 0-15 padding in the array, the performance is almost constant for every padding.
    - After running fix#1 over multiple threads, using more threads increases performance.
    - Performance of having 15 numpads in Fix#1 is almost the same as the performance of Fix#2.

4. Why do you think it is behaving this way?

- For fix#1, the performance is almost constant as there is no false sharing occurring on the cache line here as we're using a single thread. So, there will be no sense of adding padding, as false sharing will not occur.
- For fix#1 using multiple threads the performance is increased as the execution time is decreased and more number of padding will avoid false sharing as the cache line will be padded. More number of padding ensures to prevent false sharing even if the value is bigger.
- By specifying the value of 15 pads a cache line can be fully prevented from false sharing thus we get a maximum performance if we add 15 pads reserving the whole cache line for a value. Whereas, by Fix#2 we declare a variable private to each thread which completely avoids the problem of false sharing and thus the performance obtained by fix#2 is almost similar to that of fix#1 having 15 numpads.