JONATHAN PERRY
5/3/18

PROJECT #3

For this assignment, I created a bash script to help test my program with several different combinations of parameters – such as setting the number of threads used, setting the number of padding to be used, and setting the program to run with fix #1 (using padding on the data structure and no private local variables) or with fix #2 (using a private local variable with OpenMP for loop, but using no padding on the data structure). To run the script, adjust permissions to execute the script and then simply enter in the following command: ./script

1. Tell what machine you ran this on
   - A Linux system on OSU's FLIP server was used to run Project #3.

2. Table of Results

PERFORMANCE (MegaAdds Per Second)

		NUMPAD																
# of Threads	Fix #2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	48.1894	85.4742	85.3307	81.1039	85.6369	81.762	82.876	83.5874	85.4206	83.4464	82.2321	78.9953	82.2351	79.5527	84.2075	84.324	85.2348	83.9409
2	94.9435	50.4405	48.4059	50.462	43.2317	42.5664	67.3416	61.2847	167.388	164.598	162.117	155.762	161.01	156.245	164.282	164.48	167.887	164.045
4	187.144	44.8582	44.9767	44.0947	44.4808	46.9907	62.3464	63.9743	81.7976	89.052	79.7392	104.678	116.858	114.638	157.346	128.91	333.27	305.872

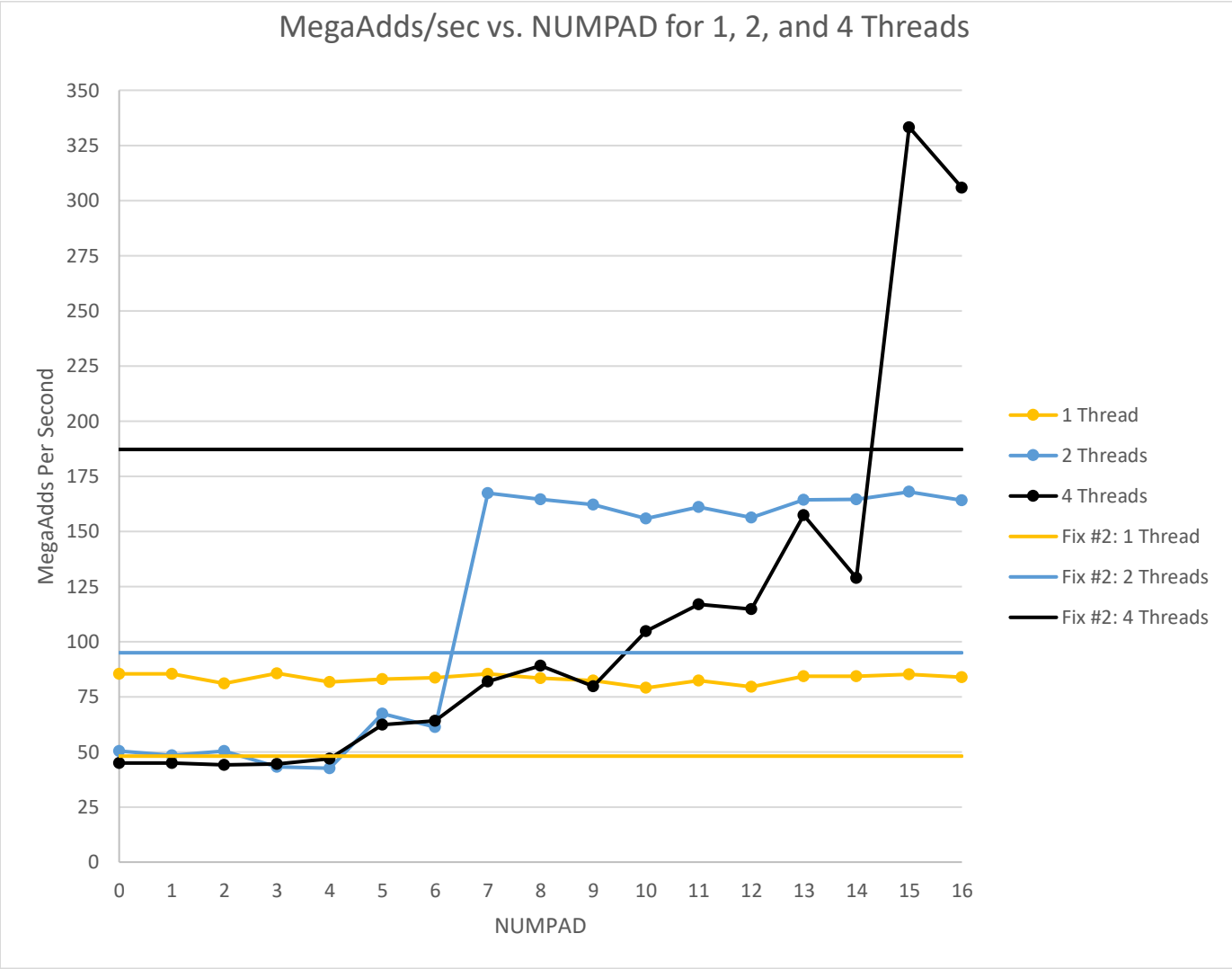
PERFORMANCE (Speedup)

		NUMPAD																
# of Threads	Fix #2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1.99908	1.46466	1.70437	0.8887	1.50144	1.59889	1.80416	1.53078	1.9264	2.02227	1.97783	1.90131	1.86621	1.96581	1.9949	1.9892	1.9314	1.95737
4	1.99336	0.77844	0.83643	1.70811	1.28323	0.61392	0.81299	0.87303	0.87474	0.97758	0.96948	1.19324	1.22385	1.14274	1.19128	0.9356	2.04428	2.04232

Elapsed Time

		NUMPAD																
# of Threads	Fix #2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	20.8027	11.6994	11.7191	12.3299	11.6772	12.2306	12.0662	11.9635	11.7068	11.9837	12.1607	12.659	12.1603	12.5703	11.8754	11.859	11.7323	11.9131
2	10.5326	19.8253	20.6586	19.8169	23.1312	23.4927	14.8497	16.3173	5.97413	6.07542	6.16839	6.42007	6.21079	6.4002	6.08708	6.0797	5.95639	6.0959
4	5.34349	22.2924	22.2337	22.6785	22.4816	21.2808	16.0394	15.6313	12.2253	11.2294	12.5409	9.55309	8.55743	8.72309	6.35541	7.7575	3.00057	3.26934

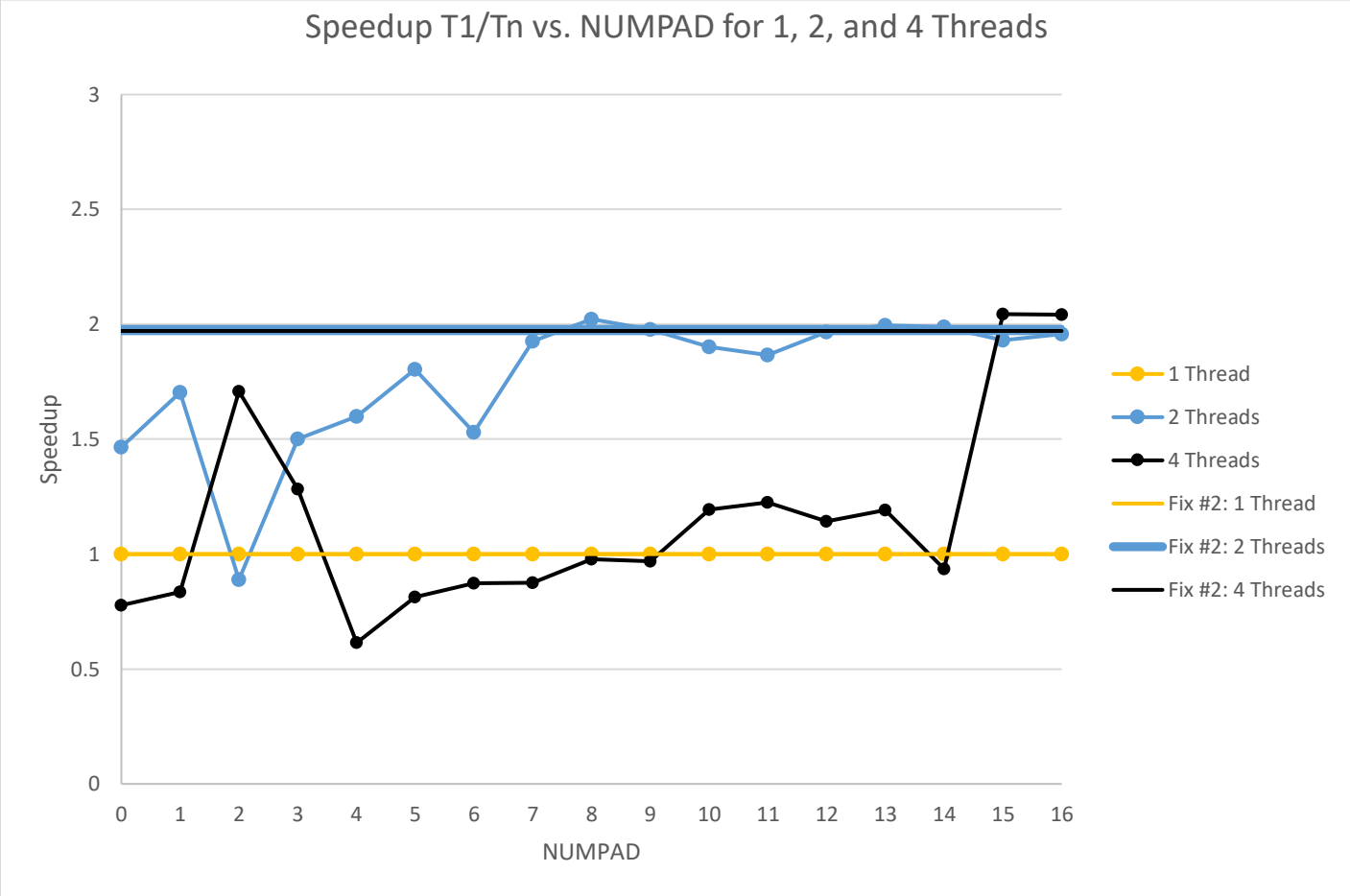
3. Graph
   - **Performance** (MegaAdds Evaluations Per Second)



JONATHAN PERRY
5/3/18

PROJECT #3

- **Performance(**Speedup T1/Tn)



4. What patterns are you seeing in the speeds?

The patterns that I'm seeing in the speeds in the graph above is that as NUMPAD is incremented from 4 to 5, we see an increase in speedup with two and four threads being used. Another pattern I'm seeing is as NUMPAD is incremented from 9 to 10 and from 14 to 15, we see an increase in speedup with four threads being used.

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

- We are seeing an increase in speedup from NUMPAD going from 4 to 5 with 2 threads because with the amount of padding increasing to 5, this places 1 of the 4 struct values from our array on a separate cache, while all 3 of the other struct values sit on another cache line.
- We are also seeing an increase in speedup from NUMPAD going from 6 to 7 with 2 threads because with the amount of padding increasing to 7, this places the 1st two of the 4 struct values from our array on a separate cache, while the last 2 struct values sit on another cache line.
- Finally, we are seeing an increase in speedup from NUMPAD going from 14 to 15 with 4 threads with the amount of padding increasing to 15, this places the each of the 4 struct values from our array on their own separate cache.
- In these three occurrences, false sharing is not occurring here as no threads conflict with each other by reading or writing from the same cache line, as each thread works on a separate cache line, where each cache line contains data that is distinct from one another.