

1. Tell what machine you ran this on

- Oregon State University Flip Server via Visual Studio Code

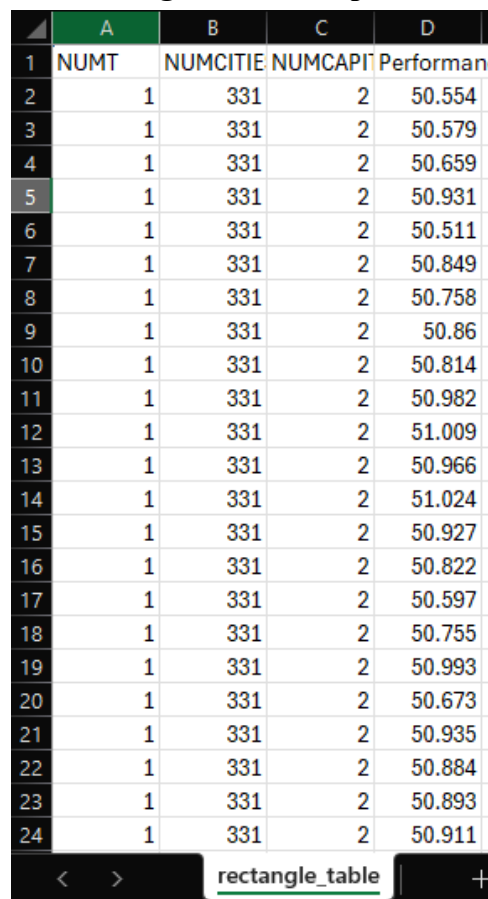
2. Tell what operating system you were using

- Linux

3. Tell what compiler you used

- g++

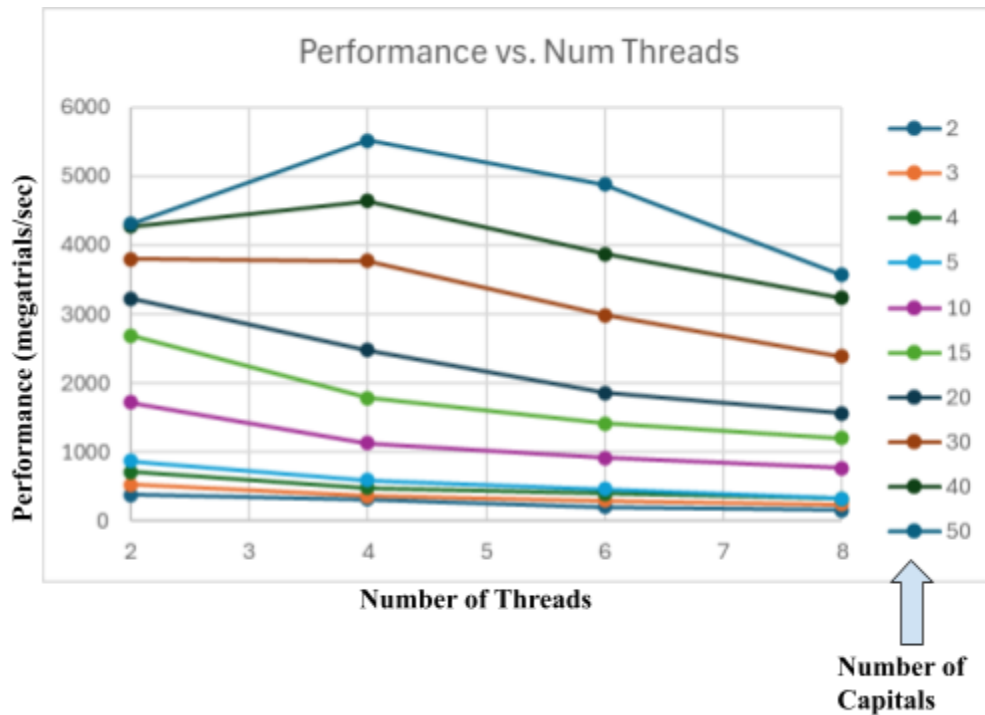
4. Include the rectangular table of performance data.



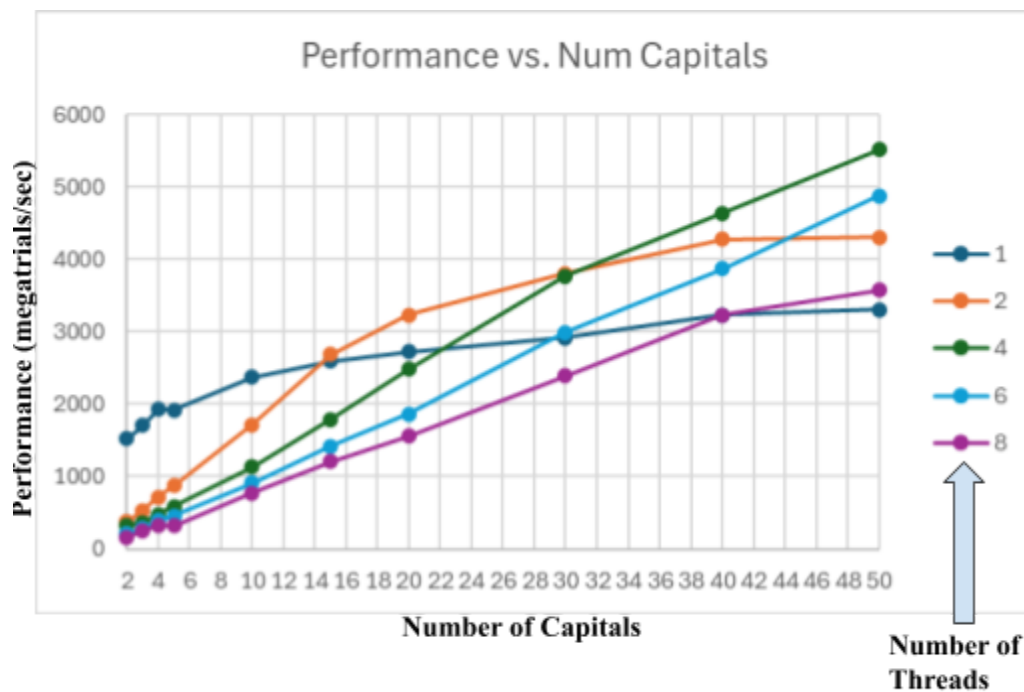
	A	B	C	D
1	NUMT	NUMCITIE	NUMCAPI	Performan
2	1	331	2	50.554
3	1	331	2	50.579
4	1	331	2	50.659
5	1	331	2	50.931
6	1	331	2	50.511
7	1	331	2	50.849
8	1	331	2	50.758
9	1	331	2	50.86
10	1	331	2	50.814
11	1	331	2	50.982
12	1	331	2	51.009
13	1	331	2	50.966
14	1	331	2	51.024
15	1	331	2	50.927
16	1	331	2	50.822
17	1	331	2	50.597
18	1	331	2	50.755
19	1	331	2	50.993
20	1	331	2	50.673
21	1	331	2	50.935
22	1	331	2	50.884
23	1	331	2	50.893
24	1	331	2	50.911

- The full table will be submitted as “rectangle.csv” since it's too big to fit here as a screenshot. (ran with bash loop).

5. Include a graph of performance vs. NUMT with the colored curves being NUMCAPITALS.



6. Include a graph of performance vs. NUMCAPITALS cities with the colored curves being NUMT.



7. Tell us what you discovered by doing this. What patterns are you seeing in the graphs?

- I notice a few things when glancing at the 2 graphs created above. The first thing I notice is the performance 1 thread has once you start increasing NUMCAPITALS. The performance starts off as the best, but then it eventually becomes the worst. This makes sense as we are making use of openmp when the number becomes larger (i.e more loops to compute distances). Another thing I notice is performance is maximized with 4 threads. I believe this has to do with Amdahl's Law. An obvious observation I notice is we get the best performance when increasing the number of capitals (for most thread counts besides 1).