CMPT 770/479: Distributed Systems (Fall 2021)

Assignment 3 - Report

Instructions:

- This report is worth 45 points.
- Answer in the space provided. Answers spanning beyond 3 lines (11pt font) will lose points.
- Input graphs used are available at the following location.
  - live-journal graph (LJ graph): **`/scratch/input_graphs/lj`**
  - RMAT graph: **`/scratch/input_graphs/rmat`**
- All your answers must be based on the experiments conducted with 4 workers on slow nodes. Answers based on fast nodes and/or different numbers of workers will result in 0 points.
- All the times are in seconds.

---

1. [12 points] Run Triangle Counting with **`--strategy=1`** on the LJ graph and the RMAT graph. Update the thread statistics in the tables below. What is your observation on the difference in time taken by each thread for RMAT and that for LJ? Why does this happen?

Answer:

The LJ graph is much more dense than the RMAT graph, this can be seen by the (almost equal) number of edges and vertices across the threads in strategy 1, so the workload is almost identical. Whereas, in LJ graph the workload is not divided equally so there is a divide in the thread timings.

Triangle Counting on LJ: Total time = 53.81312 seconds.

thread_id	num_vertices	num_edges	triangle_count	time_taken
0	1211892	42920131	339204160	53.812647
1	1211892	15515692	213398914	11.230427
2	1211892	7141449	84872361	3.457746
3	1211895	3416501	45558451	0.979338

Triangle Counting on RMAT: Total time = 3.18986 seconds.

thread_id	num_vertices	num_edges	triangle_count	time_taken
0	6249999	12650749	7	3.189352
1	6249999	12546666	5	3.156983
2	6249999	12399926	6	3.128009
3	6250002	12402659	9	3.121901

2. [9 points] Run Triangle Counting with **`--strategy=2`** on LJ graph. Update the thread statistics in the table below. Partitioning time is the time spent on task decomposition as required by **`--strategy=2`**. What is your observation on the difference in time taken by each thread, and the difference in num_edges for each thread? Are they correlated (yes/no)? Why?

Answer:

This time around the number of edges are much more comparable, and it can be seen that the time taken by each thread is much less than strategy 1 but the divide is still present in the timings of the threads, it might be because the actual counting is still done only on the basis of number of vertices.

Triangle Counting on LJ: Partitioning time = 0.0 seconds. Total time = 28.01175 seconds.

thread_id	num_vertices	num_edges	triangle_count	time_taken
0	325540	17248443	136034326	27.966894
1	510545	17248443	144912885	21.973001
2	904041	17248443	219730109	15.745213
3	3107444	17248444	182356566	8.891030

3. [9 points] Run PageRank with `--strategy=1` on LJ graph. Update the thread statistics in the table below. What is your observation on the difference in time taken by each thread, and the difference in num_edges for each thread? Is the work uniformly distributed across threads (yes/no)? Why?

Answer:

Here it can be seen that the divide between the number of edges is a lot, but the time taken by the threads are comparible, that is because of the barrier waiting. The work is not uniformly divided across the threads, as the number of edges have a difference of order of 10 across the threads.

PageRank on LJ: Total time = 41.94746 seconds.

thread_id	num_vertices	num_edges	time_taken
0	24237840	858402620	41.937171
1	24237840	310313840	41.937136
2	24237840	142828980	41.937097
3	24237900	68330020	41.922869

4. [9 points] Run PageRank with **`--strategy=1`** on LJ graph. Obtain the cumulative time spent by each thread on **`barrier1`** and **`barrier2`** (refer pagerank pseudocode for program 3 on assignment webpage) and update the table below. What is your observation on the difference in barrier1_time for each thread and the difference in num_edges for each thread? Are they correlated (yes/no)? Why?

Answer:

It can be seen that the 0^{th} thread has very little wait time on the barriers because it has the most work to do, the rest of the threads wait on 0^{th} thread before exiting the thread function. First for loop deals with the edges so barrier 1 time is higher for the threads > 0 (0^{th} thread has most edges to compute).

PageRank on LJ: Total time = 41.94746 seconds.

thread_id	num_vertices	num_edges	barrier1_time	barrier2_time	time_taken
0	24237840	858402620	0.000727	0.014058	41.937171
1	24237840	310313840	21.909496	0.010574	41.937136
2	24237840	142828980	31.171523	0.007112	41.937097
3	24237900	68330020	36.604755	0.004408	41.922869

5. [6 points] Run PageRank with **`--strategy=2`** on the LJ graph. Update the thread statistics in the table below. Update the time taken for task decomposition as required by **`--strategy=2`**. What is your observation on barrier2_time compared to the barrier2_time in question 4 above? Why are they same/different?

Answer:
Barrier 2 time is calculated after the for loop which runs on the vertices, and in edge distribution strategy there is a divide between the vertices and it is of 10 order across the threads, so the last thread has the most vertices, so its barrier 2 time is low so the other threads have to wait for barrier 2.

PageRank on LJ: Total time = 26.11097 seconds. Partitioning time = 0.0 seconds.

thread_id	num_vertices	num_edges	barrier1_time	barrier2_time	time_taken
0	6510800	344968340	1.928278	3.455517	26.055284
1	10210820	344968840	1.920440	3.222096	26.017214
2	18080880	344968660	0.539212	2.720491	25.949236
3	62148920	344969620	0.020534	0.000622	25.949169