CMPT 431: Distributed Systems (Fall 2021)

Assignment 3 - Report

Name	Yu Ke
SFU ID	keyuk

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:
   For RMAT, time taken by each thread is almost the same, however, for LJ, time taken by each thread is quite different, and one thread in LJ cost most time but other cost few time.
   The reason it happens is that each thread in RMAT has similar number of edges. However, for LJ, different threads have different edges and this difference is quite large, which makes this partition not quite good.

   Triangle Counting on LJ: Total time = 54.18582 seconds.

thread_id	num_vertices	num_edges	triangle_count	time_taken
0	1211892	42920131	339204160	0.985934
1	1211892	15515692	213398914	3.445207

2	1211892	7141449	84872361	11.243382
3	1211895	3416501	45558451	54.185321

Triangle Counting on RMAT: Total time = _3.20462_ seconds.

thread_id	num_vertices	num_edges	triangle_count	time_taken
0	6249999	12650749	7	3.204143
1	6249999	12546666	5	3.165825
2	6249999	12399926	6	3.133270
3	6250002	12402659	9	3.131511

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:

The difference of time taken by each thread is not that large compared to strategy 1.

The difference in num_edges for each thread is so small, and each thread has almost the same number of edges.

They are correlated. Because each thread has an almost equal number of edges, then the tasks need to be done for each thread are close compared to strategy 1, and this is why the time improves so much.

Triangle Counting on LJ: Partitioning time = 0.00826192 seconds. Total time = 25.74451 seconds.

thread_id	num_vertices	num_edges	triangle_count	time_taken
0	325540	17248417	182356766	25.743975
1	510542	17248443	219729921	20.599533
2	904043	17248432	144912958	15.036298
3	3107446	17248481	136034241	8.540688

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:
   The time taken by each thread is almost the same.
   The differences in edges between each thread are huge.
   Yes, the time of each thread is almost the same.

   PageRank on LJ: Total time = 66.02180 seconds.

thread_id	num_vertices	num_edges	time_taken
0	24237840	858402620	66.020894
1	24237840	310313840	66.020859
2	24237840	142828980	66.020803
3	24237900	68330020	66.006887

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:
   The time of barrier1_time for each thread is not the same and the differences are quite huge. This barrier1_time is the main cause of the running time.
   The differences in edges for each thread are quite huge.
   They are correlated. Because a different number of edges means different numbers of tasks, which means some will finish first and others will finish late, then the barrier 1 time will increase by it.

PageRank on LJ: Total time =66.02180 seconds.

thread_id	num_vertices	num_edges	barrier1_time	barrier2_time	time_taken
0	24237840	858402620	0.000684	0.012043	66.020894
1	24237840	310313840	40.244817	0.005016	66.020859
2	24237840	142828980	53.741630	0.003323	66.020803
3	24237900	68330020	60.488241	0.001115	66.006887

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:

Compared to strategy 1, the barrier2 time in strategy 2 in longer.

This is because in strategy 1, the vectices for each thread are evenly distributed, which make the second loop ends almost the same time.

However, in strategy 2, the number of the vertices for each thread are quite different, which means some threads need to wait for others to finish, then the time increases.

PageRank on LJ: Total time = _33.02646995__ seconds. Partitioning time = _0.00865984__ seconds.

thread_id	num_vertices	num_edges	barrier1_time	barrier2_time	time_taken
0	6510800	344968340	0.000700	1.6890180	33.010630
1	10210840	344968860	0.490669	1.568135,	33.025610
2	18080860	344968640	1.538642	1.327963	33.025583
3	62148920	344969620	2.940599	0.000670	33.169456