Postmortem Graph Analysis on the Temporal Graph
Md Maruf Hossain, Erik Saule
mbossa10, esaule@uncc.edu
UNC Charlotte.

Abstract
Traditional graph analysis that expects a fixed data set is not sufficient for these kinds of temporal graphs. Streaming graph algorithms are more popular to handle this kind of problem. In this poster, we are going to show the performance analysis of Pagerank on the temporal graph. Most of the previous research study shows that incremental way more popular for the temporal graph. But the question is if the data shows the offline nature that means if the whole data available at the beginning of the process, and need to perform a series of graph analysis for a list of time interval, should we still choose streaming graph algorithm for the temporal graph analysis? To find the answer, we need to look at another graph analysis, which is called Postmortem graph analysis. In the Postmortem analysis, one performs graph analysis on multiple subgraphs based on the well-defined time interval. In this study, we are going to show the Postmortem graph analysis can provide better Pagerank performance on the temporal graph than streaming graph analysis.

Streaming Pagerank
In the streaming system, a batch of edges arrive in the system and based on the epoch and window size an edge can insert or delete from the system. Based on the batch size, the algorithm will enable to perform Pagerank. We choose STINGER [Rie16] to perform the streaming version of Pagerank. STINGER also support shared memory parallelization for the dynamic Pagerank.

Postmortem Pagerank
In this paper, we are going to show the performance analysis of the streaming, naive and postmortem graph analysis on the sx-stackoverflow network for window-size = 0.8.

Experimental Settings
1. Processor: Intel SkylakeX.
2. Number of Sockets: 2.
3. Cores: each socket contains 12 cores.
4. Threads: each core has 2 hyper-threads.
5. Cache: L1(32K), L2(1024K) and L3(19712K).
7. Processor: Intel SkylakeX.
8. Cores: each socket contains 12 cores.
9. Threads: each core has 2 hyper-threads.

Naive Pagerank
In the Naive Pagerank, the PageRank is calculated once for each time interval. The algorithm is simple and straightforward, and it can be easily parallelized. However, it is not efficient for large graphs or real-time applications.

Algorithm 1 Naive Pagerank
Input: G = (V, E), α, T1, T2, T3
1. For each node v in V:
2. rank(v) = 1
3. prevRank(v) = 0
4. pagerank = 0
5. for t in T3:
6. for v in V:
7. pagerank = pagerank + prevRank(v)
8. prevRank(v) = pagerank
9. end for
10. end for
11. end for
12. end for

Graphs
Table: Graphs and temporal analysis information for pagerank.
<table>
<thead>
<tr>
<th>Name</th>
<th>Epoch</th>
<th>Window Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>stack-overflow</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wiki-talk</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0.5</td>
</tr>
</tbody>
</table>

References

Acknowledgment
This work is supported by grant from the National Science Foundation CCF-1652442 and was made possible by a computing allocation given by TACC through XSEDE.