Postmortem Graph Analysis on the Temporal Graph

Md Maruf Hossain
University of North Carolina at Charlotte
Charlotte, NC, USA
mhossa10@uncc.edu

Erik Saule
University of North Carolina at Charlotte
Charlotte, NC, USA
esaule@uncc.edu

ABSTRACT

Graphs like social networks and web graphs are evolving over time intervals. 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. Lots of algorithms have been proposed to compute *Pagerank* in the evolving 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. On the other hand, streaming mainly focus on gradually update the graph analysis results based on the current events of the edge addition or deletion, *etc.* 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.

ACM Reference Format:
Md Maruf Hossain and Erik Saule. 2021. Postmortem Graph Analysis on the Temporal Graph. In *ICPP '21: The 50th International Conference on Parallel Processing, August 9th, 2021, Chicago, Illinois, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In the modern world, almost all of the social graphs evolve over time and it is important to record the timeline of these changes. These dynamic networks are called temporal graphs. We assume all the temporal graphs split into multiple time-window and each window is called the interval. The temporal graph changes through different events, such as edge addition, deletion, vertex addition, *etc.* The interest in evolving graphs is growing over time, for instance, diameter change [3], the rank of web pages change [5] on the web.

The popularity of *Pagerank* on the temporal graph is also growing. Most of the algorithms on evolving graphs follow the streaming or incremental model. On the incremental model, graph build is a gradual process that means one interval depends on the other. But there is another interesting variation of the evolving data is the *postmortem* graph. In the *postmortem*, all the information of the data is known prior to the analysis. So, in the postmortem analysis, one already knows the future of the graph. In this poster we are going to show *postmortem Pagerank* on temporal graphs is more efficient than the streaming model.

Chicago '21, August 9th, 2021, Chicago, Illinois, USA
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 BACKGROUND

2.1 Temporal Graph

Usually, a temporal graph consists of time stamp that represents the creation or deletion time for a particular edge or vertex. So, based on this time one can generate a graph for a particular time interval and perform different graph analysis on that graph. We can call it as a window of the original graph. Now, this window can slide from past to future or vice-versa. Based on the shifting the window or the size of the window we can get different sub-graphs. The main aim of the paper to perform different graph analysis on these window graph.

2.2 Pagerank for the Streaming Graph

A directed graph $G(V, E)$ with vertex and edge set V and E can be represented by a sparse matrix A where an edge$(i \longrightarrow j)$ is represented by $a_{ij} = 1$. Now, if we represent the out degree of the graph by a diagonal matrix D then according to the [1, 2], *Pagerank* can be defined by the linear system,

$$(I - \alpha A^T D^{-1})x = (1 - \alpha)v \tag{1}$$

Where α is the "teleportation" constant, v is the initial *Pagerank* vector usually filled by $1/|v|$ and x is the *Pagerank* vector. Jason presented [4] an approximation version of *Pagerank* for the streaming graph,

$$\Delta x^{k+1} = \alpha A_\Delta^T D_\Delta^{-1} \Delta X^k + \alpha(A_\Delta^T D_\Delta^{-1} - A^T D^{-1})x + r \tag{2}$$

where modifications of the streaming graph by edge addition or deletion are represented by Δ and k is represent the previous iteration. Here r is the residual error, $r = (1 - \alpha)v - (I - \alpha A^T D^{-1})x$.

3 TEMPORAL GRAPH ANALYSIS

Temporal graph represents the dynamic graph over the time stamp. So, the common practice to perform the graph analysis on the sub-graph for a specific interval. The time stamp on every edge represents the arrival time on the graph. So if the time stamp of any set of edges greater or equal to the lower limit of the interval and less or equal to the upper limit of the interval then these edges will participate in the graph analysis. Now if the temporal dataset shows the offline characteristic then we have two options to choose for the graph analysis, one is streaming and another one is postmortem.

STINGER [3] is a multi-process framework and most common for streaming graph analysis. We are going to compare against STINGER version to our postmortem *Pagerank*. For *Pagerank* analysis, we gather the changes of the graph as a batch of edges. We fixed the `batch-size` for all the experiments. Now, based on the fixed `batch_size` the number of batches(`num_batches`) differ by the size of the edges(e), `num_batches` $= \frac{|e|}{\texttt{batch\_size}}$. Now, how many times and when we want to perform the graph algorithm is decide by the `num_epochs`. For a particular `batch_id` the algorithm `enables` by the following condition,

$$
\begin{aligned}
batches\_per\_epoch &= \frac{num\_batches}{num\_epochs}\\
batches\_before &= \left\lfloor \frac{batch\_id}{batches\_per\_epoch} \right\rfloor\\
batches\_after &= \left\lfloor \frac{batch\_id + 1}{batches\_per\_epoch} \right\rfloor\\
enable &= (batches\_after - batches\_before) > 0
\end{aligned}
$$

The next thing is the size of the sub-graph, to control the sub-graph we need another attribute `window_size`. The `window_size` represents the percentage of the total graph, that is why the max value of the `window_size` 1.0. Let say, at time t the system enables the algorithm then all the edges with a timestamp greater than $t - (window\_size * graph\_span)$ and lower than t will be included in the sub-graph. Here $graph\_span = max\_time\_stamp - min\_time\_stamp$. Although *Postmortem* does not need batch information because all the data available to the system, only need starting interval and end interval of a `window`. For a fair comparison, we perform a pre-process to extract all the exact time intervals as similar to the streaming system and then perform graph analysis.

4 PAGERANK ON TEMPORAL GRAPHS

Parallel Naive Pagerank(PNPR). As we said, *Postmortem* data has full availability during graph analysis, so one can perform a naive *Pagerank* as well. By the naive we mean, build a sub-graph for a particular epoch every time from scratch and perform *Pagerank* on that sub-graph. Otherwise can go for either traditional streaming or postmortem graph analysis. We represent *parallel naive Pagerank* by the *PNPR*.

Parallel Streaming Pagerank Model(PStPR). 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. And based on the `batch_id` the algorithm will enable to perform *Pagerank*. We choose STINGER [4] to perform the streaming version of *Pagerank*. STINGER also supports shared-memory parallelization for the dynamic *Pagerank*.

Parallel Postmortem Pagerank Model(PPoPR). Unlike the naive(*PNPR*) version, we perform *Pagerank* on the full graph. So, we do not need to extract any sub-graph for any interval. But we need an additional array data structure to support the correct analysis. In the temporal graph, the same edges can occur with a different time-stamp. In our analysis, we always treat edges between two same vertices once.

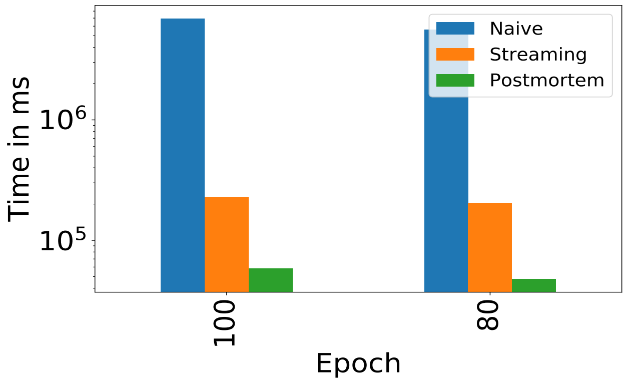

Figure 1: Pagerank performance of the streaming, naive and postmortem graph analysis on the wiki-talk network for window-size = 0.8.

Although *Postmortem* analysis shows more memory foot-print compare to the *naive* and *streaming* but it reduces significant amount of computational time.

5 EXPERIMENTAL RESULTS

All the experiments are performed on the Intel `Skylake` processor with 2 sockets and each of them contains 12 cores in total 24 cores and each core has 2 hyper-thread mechanisms. Figure 1 shows the performance of the parallel naive *Pagerank*(PNPR) against streaming(PStPR) and postmortem(PPoPR). Parallel postmortem over perform naive and streaming model. The graph-building process in the naive version requires extra time compared to the others. The streaming version only inserts and deletes the necessary edges and *Postmortem* build once for full graph analysis.

6 CONCLUSION

This study brings some insight into postmortem graph analysis on temporal graphs. Most of the state-of-art algorithms focus on the incremental version of the streaming model for event-based graphs. For online graph networks, it is obvious to use streaming data. But our study shows that *postmortem* Pagerank can overperform the streaming model on the temporal graph. In this work, we only focus on Pagerank. But in the future, we will explore other graph algorithms such as *Closeness* and *Betweenness Centrality*.

REFERENCES

[1] Gianna M Del Corso, Antonio Gulli, and Francesco Romani. 2005. Fast PageRank computation via a sparse linear system. *Internet Mathematics* 2, 3 (2005), 251–273.
[2] David Gleich, Leonid Zhukov, and Pavel Berkhin. 2004. Fast parallel PageRank: A linear system approach. *Yahoo! Research Technical Report YRL-2004-038, available via http://research. yahoo. com/publication/YRL-2004-038. pdf* 13 (2004), 22.
[3] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proc. of SIGKDD*. 177–187.
[4] Jason Riedy. 2016. Updating pagerank for streaming graphs. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 877–884.
[5] Lei Yang, Lei Qi, Yan-Ping Zhao, Bin Gao, and Tie-Yan Liu. 2007. Link analysis using time series of web graphs. In *Proc. of CIKM*. 1011–1014.