Towards Scalable Graph Analytics on Time Dependent Graphs

Suraj Poudel[2] , Roger Pearce[1] , Maya Gokhale[1]

[1]Center for Applied Scientific Computing, Livermore Lawrence National Laboratory

[2] University of Alabama at Birmingham

Abstract

The objective of this study is to **annotate temporal metadata** into the **partitioned graph topology** of a scale-free graph. Our time dependent graph has been modeled using multiple metadata per edge, where each edge metadata field encodes temporal information. We performed experiments with large scale **CAIDA datasets**, anonymized internet traces, on a **Linux HPC cluster** (**Catalyst**) by deriving flow between two IP's and annotating that flow information as metadata into the graph topology using the asynchronous visitor computation model of **HavoqGT**.

With the dataset, graph-representation, and HavoqGT, we implemented and evaluated time dependent Single Source Shortest Path (TD-SSSP) and Temporal Betweenness Centrality**.**

Metadata

CAIDA Datasets : Passive traffic traces and packet header traces saved per direction per server.

Flow Generation : Packets grouped into flows by including packets within the timeout period in each group per IP address starting first packet :

	Packet #1	**Packet #2**
Source IP	83.87.9.183	83.87.9.183
Source Port	61213	61213
Destination IP	63.8.48.222	63.8.48.222
Destination Port	1935	1935
Start Time	1395320351.575410061	~~1395320351.575469302~~
End Time	~~1395320351.575441296~~	1395320351.575482592
Size (in Bytes)	900	2008

Timeout : 0.0001s

Resulting Flow #1	Resulting Flow #2
83.87.9.183	...
61213	
63.8.48.222	
1935	
1395320351.575410061	
1395320351.575482592	
2908	

Flow as Edge Metadata

Edge :
Source IP:Port → Destination IP:Port
Metadata :
Time Interval(t): (Start Time, End Time)
Parameters: { Size in Bytes, Packet Count }

Referred as: *Edge : Metadata(t, params)*

Resulting Dataset (Graph)

1-Hour Data
Total Flows (Edges) : 1.35 Billion
Total IP's (Vertices) : 7.9 Million
Size (in GBs) : 190

Partitioned-Graph Topology

- Create *partitioned-graph topology* as in *Reference [1].*

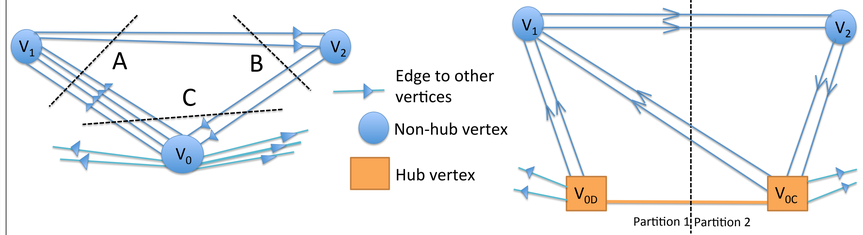


Fig: 1.a. Example Graph Topology ***Fig: 1.b. Partitioned Graph Topology***

Vertex based partitioning (A, B and/or C) produces workload imbalance!

Edge based partitioning with delegates of Hub vertex balances workload.

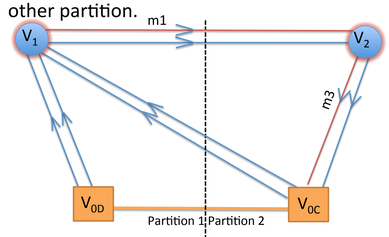
*Figure 1 : Example of a graph topology **a) before** and **b) after** edge partitioning with vertex delegates and edge balancing.*

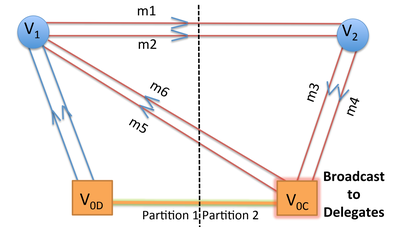
- **Partitioning Technique**: Delegate based partitioning techniques with asynchronous visitor model ***load balances***:
  - ✓ *Computation : Edge distribution yields better workload division*
  - ✓ *Communication : Delegates communication in O(partitions)*
  - ✓ *Storage: Graph partitions are distributed across compute nodes.*
- **Controller** : A representative of distributed hub vertices (per hub vertex) [e.g. V_{0C}].
- **Delegates** : All other distributed hub vertices (per hub vertex) [e.g. V_{0D}].

1. Roger Pearce, Maya Gokhale, Nancy M. Amato. *"Faster parallel traversal of scale free graphs at extreme scale with vertex delegates." Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC14*. IEEE Press, 2014.

Edge Annotation

- Read metadata files (partitioned logically) at each partition.
- Annotate local edges with locally available metadata, if any.
- Asynchronously transfer metadata of non-local edges to corresponding partition.
- With hub vertices, communicate with delegates to annotate the corresponding edge if at other partition.





Resulting time annotated partitioned graph:

- Partitioned metadata resides locally in conjunction with partitioned topology.
- Asynchronous Visitor Model (in *Reference [1]*) is complemented with temporal information locally to devise time dependent graph analytics.

Time Dependent Graphs

Let $\mathbf{E(t_s, t_e)}$ be set of edges with temporal metadata **between** time interval **[t_s : start time, t_e: end time)**. An example of a Time Dependent Graph between interval t_s and t_e depicted as $\mathbf{G_K (V, E(ts, te))}$ is below:

Figure 2 :Changes from metadata m5 to m6 and m3 to m4 is seen in the edges of the graph G1 to G2. Edges are deleted and added from G2 to G3.

Graph Analytics and Initial Results

Graph Analytics on Time Dependent Graphs :

- Time independent graph analysis models a static graph with respect to a single view of the topology.
- Traversal based time dependent graph analysis has two important time parameters that greatly affects the analytics *(as illustrated in Figure 3*).
  - ❑ ***Start Time*** : Time traversal begins
  - ❑ ***Waiting Time*** : Time traversal can wait at each vertex on a path
- 2 Time Dependent Implementations : **Single Source Shortest Path** and **Betweenness Centrality**.

Initial Scalability Test:

- Cluster : 150 TeraFLOP/s Catalyst, Linux HPC Cluster. Each compute node:
  - 24 12-core Intel Xeon E5-2695v2 processors
  - 800 GBs of NVRAM, 128G main-memory
- Test Application: Time Dependent Betweenness Centrality for higher waiting time (*higher waiting time has more workload as shown in Figure 4*).
- Results show good strong scalabilty for TD graph analytics (*in Figure 5*).

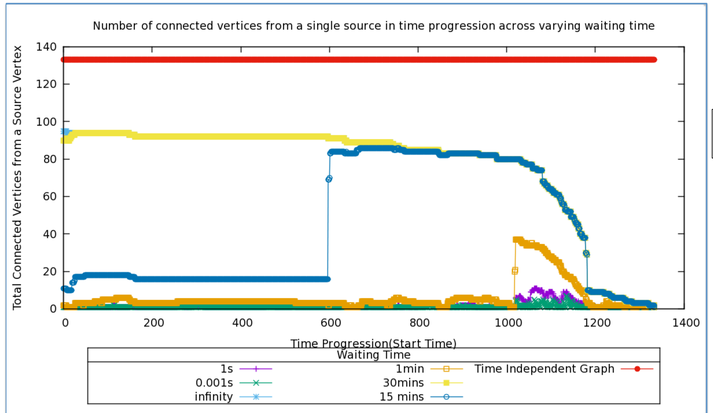


Figure 3: Number of connected vertices from a single source with change in start time of traversal. Time-Independent traversal traversed 133 connected vertices while the time dependent traversal creates significant variation that is also affected by the waiting time spent on each vertex.

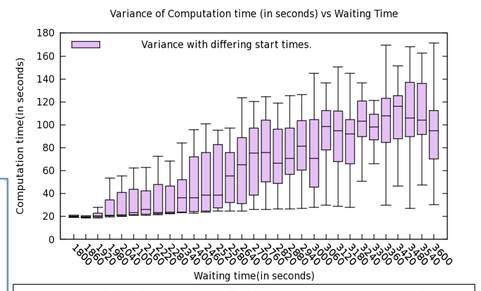


Figure 4: *Plot of variance of the computation time of BC at differing start times across various waiting time.*

Figure 5 : *Plot of variance of computation time across various start time with respect to the number of compute nodes.*

- Figure 3 shows that in this dataset more waiting time at earlier start time allows more neighboring edges to be traversed and hence has more connected vertices.
- Following from connected vertices at Figure 3, computation of time-dependent betweenness centrality (BC) showed huge variation with respect to differing start times across various waiting time *as seen in Figure 4*.
- From Figure 3 and 4 it can be seen that more computation is required for higher waiting time at earlier start times.

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

LLNL-POST-675681