DynoGraph: Benchmarking Dynamic Graph Analytics

[Extended Abstract]

Eric Hein
Georgia Institute of Technology
North Avenue, Atlanta, GA 30332
ehein6@gatech.edu

Tom Conte
Georgia Institute of Technology
North Avenue, Atlanta, GA 30332
tom@conte.us

ABSTRACT

Large scale graph processing is the key to understanding complex relationships, ranging from the interaction of people on social media to the flow of data through a corporate computer network. Graph analytics present unique challenges for HPC system designers since they lack data locality and are difficult to partition into equally-sized units of work. In addressing these challenges, many researchers have opted to work with static graphs instead of the more difficult case of dynamic graphs that change rapidly during the analysis. Dynamic graphs require an entirely different memory layout, leading to degraded performance as algorithms traverse a fragmented, unsorted graph data structure. DynoGraph provides a standard for benchmarking dynamic graph analytics engines, bringing needed focus to this important class of applications.

Keywords

dynamic graphs, benchmark, data analytics

1. INTRODUCTION

The way a graph is stored in memory affects the performance of graph analytics. The Compressed-Sparse-Row (CSR) format stores graphs more efficiently, but an adjacency list is easier to update, making it the data structure of choice for dynamic graph applications. Despite this important difference, many researchers equate static graphs with dynamic graphs when evaluating system performance.

2. DYNOGRAPH

DynoGraph is a benchmark suite for dynamic graph analytics engines. It aims to accurately capture the unique performance characteristics of dynamic graphs by focusing not only on algorithms, but also on the performance of edge insertions and deletions and their effect on the efficiency of the graph layout. DynoGraph is designed to aid computer architects in satisfying the needs of dynamic graph applications when designing processing-near-memory accelerators for emerging high-bandwidth memory systems

SC '16 November 13–18, 2013, Salt Lake City, UT, USA

DOI: 10.475/123_4

2.1 Graph Algorithms

The DynoGraph benchmarks are defined as regions of interest within a streaming graph analysis application, which alternates between running graph algorithms and updating the graph with batches of new edges. The "input" to a DynoGraph benchmark thus includes not only the graph dataset, but also the state of the graph when the analysis was started. Each batch is considered a separate benchmark. DynoGraph algorithms are also expected to filter the graph based on a timestamp threshold. This introduces irregularity into the graph traversal logic and mimics the functionality of real streaming graph applications.

Besides the edge stream, DynoGraph does not introduce any new graph algorithms. Still, the complex interaction between a graph algorithm and the underlying data structure means that each pairing of algorithm and graph engine deserves individual study.

The following algorithms are evaluated with DynoGraph in this work: Breadth-First Search (bfs), Connected Components (cc) [5], Betweenness Centrality (bc) [4], PageRank (pagerank), Kcore decomposition (kcore), and Clustering Coefficient / Triangle Count (tc).

2.2 Graph Inputs

Each DynoGraph dataset takes the form of a continuous stream of edges that occurs over a period of time. There are many duplicate We now present three new streaming datasets for DynoGraph. Each dataset was anonymized before processing to eliminate the possibility of leaking sensitive data.

- **SC 2015 NetFlow** A team of researchers collected NetFlow data from SCinet for the duration of the conference in 2015 [2]. In this graph, each vertex represents a SCinet IP address. An edge represents that data was transferred between those two hosts. Edges are weighted with the number of bytes transferred over the analysis time window considered. This dataset is representative of the types of graphs generated when analyzing real networks for cyber security threats where the application of PageRank, betweenness centrality, and community detection are used to find botnets, emergent graph behavior, and potential distributed denial of service (DDoS) attack targets.

- **Passive DNS** This dataset is an anonymized graph

of real DNS data collected over the entire campus network of a large university. Edges in this graph represent domain name lookups for a given host as they occur in time, and track the resolution of the name requests as they traverses the hierarchy of name servers. Real-time applications of this data include using centrality and community detection to locate hackers and bot-net control networks, which maliciously manipulate DNS records to accomplish their goals.

- **Twitter** Social media remains one of the most prevalent applications of dynamic graph analysis. This data set represents the Twitter graph of mentions between Twitter users that occurred over a three week period during the 2014 World Cup. The Twitter graph was collected targeting hash tags specific to the World Cup as well as hash tags specific to the Twitter campaigns of the primary sponsors.

3. RESULTS AND CONCLUSIONS

We used DynoGraph to compare the performance of GAP [1], a graph benchmark that uses CSR, against STINGER[3], a dynamic graph engine that implements a parallel adjacency list. The benchmarks were run on shared-memory machines with dual-socket Intel Xeon processors. As expected, the static graph implementation provided by GAP had the best performance for the graph algorithms, however STINGER was able to stream in new edges more efficiently. It was also discovered that deletions had a significant effect on the performance of STINGER.

Dynamic graph analytics represents an entirely different workflow from static graph processing, and deserves separate attention. The in-memory layout of a dynamic graph degrades over time as edges are inserted and deleted. It is important to model this effect when measuring performance. DynoGraph provides a framework for accurately benchmarking dynamic graph processing on different engines and architectures.

4. REFERENCES

[1] S. Beamer, K. Asanović, and D. Patterson. The GAP Benchmark Suite. 2015.

[2] D. Campbell, D. Ediger, J. Poovey, and T. Goodyear. Real-time Traffic Classification & Graph Analytics for SCinet. misc, 2014.

[3] D. Ediger, R. McColl, J. Riedy, and D. Bader. STINGER: High Performance Data Structure for Streaming Graphs. The IEEE High Performance Extreme Computing Conference (HPEC), 2012.

[4] R. Geisberger, P. Sanders, and D. Schultes. *Better Approximation of Betweenness Centrality*, chapter 8, pages 90–100.

[5] Y. Shiloach and U. Vishkin. An O(log n) Parallel Connectivity Algorithm. *J. Algs.*, 3(1):57–67, 1982.