CS 6968: Data Str & Alg Scalable Comp Spring 2023

Lecture Streaming Graphs Incremental Computations — 3/29, 2023

Prof. Prashant Pandey *Scribe: James McMahon*

1 Overview

In the last lecture we covered different representations of graphs and their trade offs.

In this lecture we discussed streaming graphs and different approaches to implementing them

2 Streaming Graphs

Streaming Graphs receive a stream of queries/updates and must process both with low latency.

Things that we want in a Streaming Graph:

- small space
- high concurrency
- low cost of updates/queries

Existing Streaming Graphs processing frameworks are divided into two types:

- *Phased* – Process updates/queries in phases, i.e: updates wait for existing queries to finish. (single writer *or* multiple readers)
- *Concurrent* – single writer and multiple readers(on previous snapshot). Snapshots represent the graph at a specific point in time. Queries are isolated and run on a snapshot and updates generate new snapshots.

The following sections describe different systems/implementations of streaming graphs:

2.1 Stinger: Ediger et al. HPEC 2012 [3]

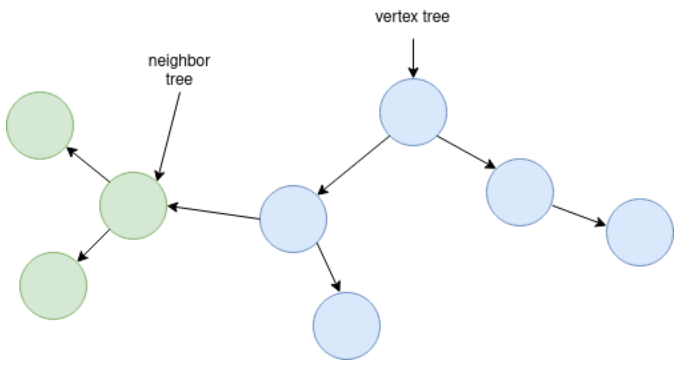
- Stinger uses fine grained locking, this was good in 2012, but with the locking and high overheads it starts to suffer.
- Stinger uses an adjacency list, specifically Compressed Sparse Rows (CSR).
- Stinger supports point updates (can be concurrent)

2.1.1 LLAMA: Mackeo et al. ICDE 2015 [2]

- Supports batch processing (more like a phased approach)
- single writer, multiple readers
- each batch of updates take a snapshot. O(N) + O(k) extra space needed for every snapshot (N = number of nodes, k = number of edge updates in this batch)
- also uses CSR

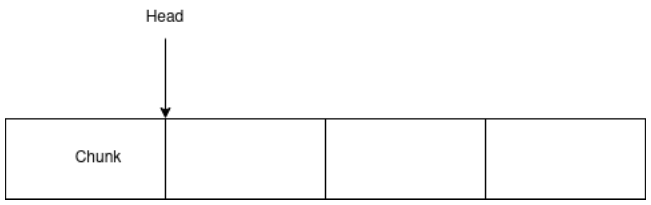
2.1.2 Aspen: Dhulipala et al. PLDI 19 [1]

- snapshot based system
- supports batch processing
- uses a purely functional balanced search tree
  - In a purely functional tree, acquiring a snapshot is like acquiring a pointer to the root of the vertex tree. (there is one pointer to the whole structure, updating the pointer can point to a different structure)
- It uses a tree of trees model.
  - First, there is a search tree over vertices
  - Second, for each vertex, there is a search tree over incident edges.

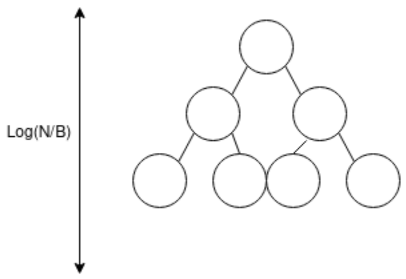


Compared to CSR Aspen uses a structure called a C-Tree. It is a probabilistic tree that is in between Binary and B trees. For all items in a tree, use a hash function to get approximately $\frac{N}{B}$ items.
H:k $\rightarrow$\{0...N\}
$\frac{N}{B} items \rightarrow$ heads

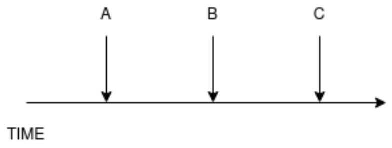
Everything that is in between the heads is a chunk:



The height of a C-Tree is $\log \frac{N}{B}$



Serializability: Operations/Queries occur in a specific order. It is fine if the result of a query is received at a later step, but we want to ensure that we see everything that occurred before us, and not anything that occurred after, the end result needs to be a single order of operations.



References

[1] Laxman Dhulipala, Guy E. Blelloch, Julian Shun Low-latency graph streaming using compressed purely-functional trees *PLDI 2019*, 918–934, 2019.

[2] Peter Macko, Virendra J. Marathe, Daniel W. Margo, Margo I. Seltzer LLAMA: Efficient graph analytics using Large Multiversioned Arrays *ICDE 2015*, 363–374, 2015.

[3] David Ediger, Robert McColl, E. Jason Riedy, David A. Bader STINGER: High performance data structure for streaming graphs *HPEC 2012*, 1–5, 2012.