Graphs that change in real-time.

→ Streaming graph system recieve a stream of queries / updates and must process both with low latency.

→ Existing dynamic graph processing frameworks are divided in two types:

① Phased: process updates / queries in phases, i.e. updates wait for queries to finish.

→ Most systems take this approch.
→ Graphs can be mutated without worrying about the consistency of queries.

② Updates / queries are run concurrently:
→ The main idea is snapshot.
→ Queries are isolated an run on snapshot.
→ Upates generate new snapshot.

Stinger: [Ediger et. al. HPEC 2012]

- Spatio-Temporal Interaction Networks and Graph Extensible Representation.
- Based on linked list of blocks
- Supports both node/edge insertions.
- Fine grained Locking.

LLAMA: [Macko et. al ICDE 2015]

- Design similar to Stinger.
- However, it is designed for batch processing.
- Single-writer Multi-Reader
- supports snapshots
- Each batch creates a snapshot of O(n) space to store vertex array. O(K) space to store edge updates.

Aspen [DHULIPALA et. al PLDI 19]

- Snapshot based system
- Supports batch processing.
- Purely-functional balanced Search Trees.

- It uses a tree of tree's model.

  1. Search tree over vertices
  2. For each vertex, a search tree over incident edges.

- Purely-functional tree.
  - Acquiring a snapshot is like acquiring a pointer to the root of the vertex-tree.

Compared to CSR:
- Less locality
- Compression is hard
- Extra lgn work in search tree.

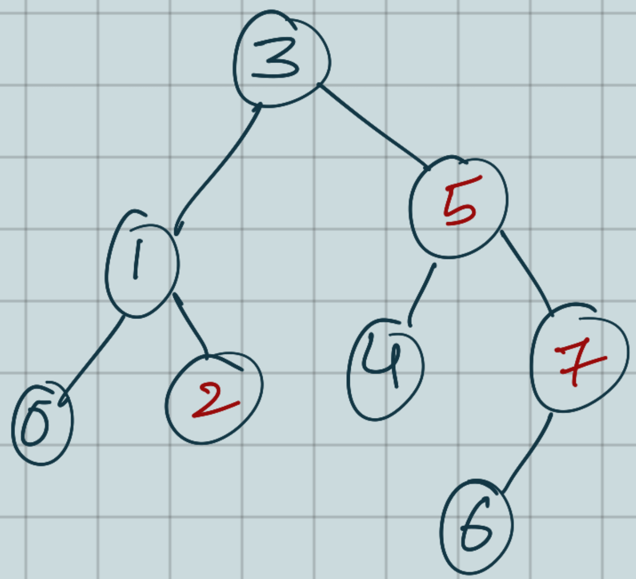
- Aspen uses C-tree.
  - Compressed purely-functional tree.
  - It uses chunking and compression inside chunks.

Purely-function Tree:-

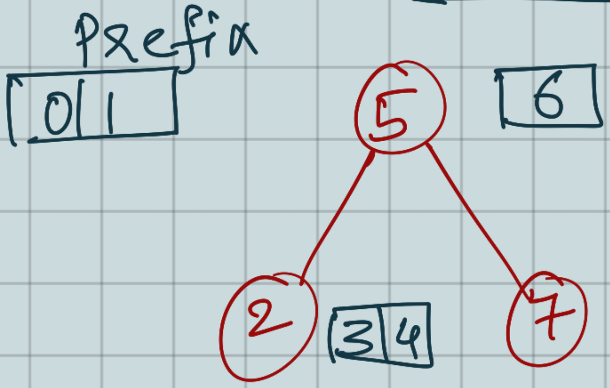
Purely-functional (or mutation-free) data structures preserve previous versions of themselves when modified and yield a new structure reflecting the update.

C-tree:

Purely function tree:-



C-tree.



→ Use a hash function to pick the head elements.

→ this guantees same heads will be picked across trees.

$h: K \rightarrow \{1 \cdots N\}$.

heads $H(E) = \{e \in E \mid h(e) \bmod B = 0\}$.

For each $e \in H(E)$

tail $t(e) = \{x \in E \mid e < x < next(H(E), e)\}$

→ Expected size of each chunk :- c

W.H.P size of each chunk :- $c \lg n$.

→ # of heads W.H.P (n/b)

→ Each chunk is further compressed in the tree to save space.

V → Vertices E → Edges.

	Ligra (CSR)	Aspen (tree).
add edge	$O(E+V)$	$O(\lg V + c^2 \lg(\deg(v)))$
find edge	$O(\lg(\deg v))$	$O(\lg V + c)$ in Exp $O(\lg V + c \cdot \lg(\deg v))$ W.H.P
get neighbors	$O(\deg v)$	$O\left(\lg V + \deg(v) + \frac{\deg(v)}{c}\right)$

Terrace :

- Hierachical storage of edges.
- Exploits the skewed degree dist. of edges in real-world graph.
- Partition nodes based on their degree and use different data structures.

- Small degree nodes → CSR.
- Medium degree nodes → PMA (packed memory array)
- Large degree nodes → B-tree.

- Nodes are moved between levels dynamically.
- No cache misses for small degree nodes
- Asymptotically optimal for large degree nodes.

Packed-Memory Array:

- Array based order maintenance data structure.

- For N elements takes
  - $O(N)$ space
  - $O(\lg^2 N)$ amortized updates
  - $O(\lg N)$ queries

- A PMA maintains an implicit complete binary tree on its cells with leaves of $\lg N$ cells each.