GraphZeppelin: Processing Enormous, Changing Graphs

Evan West, David Tench, Victor Zhang, Michael A. Bender, Abiyaz Chowdhury, J. Ahmed Dellas, Martin Farach-Colton, Tyler Seip, Kenny Zhang

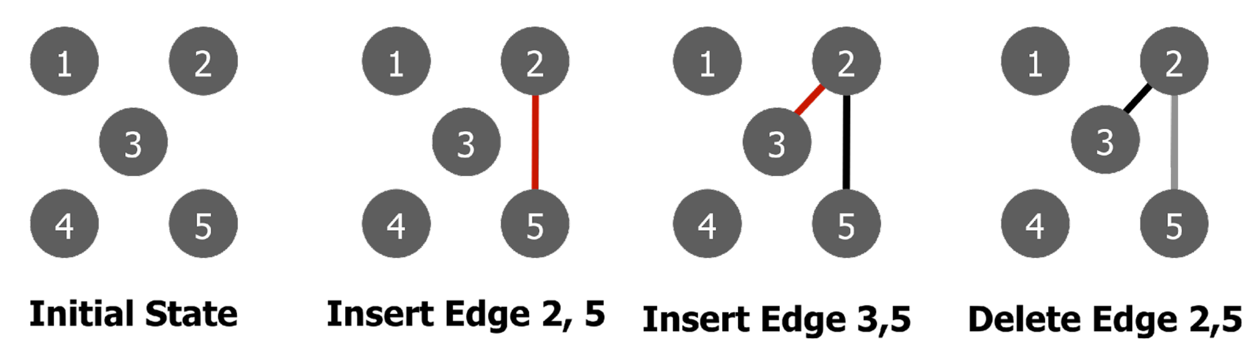
Stony Brook University, Rutgers University, MongoDB

Stony Brook University
Computer Science

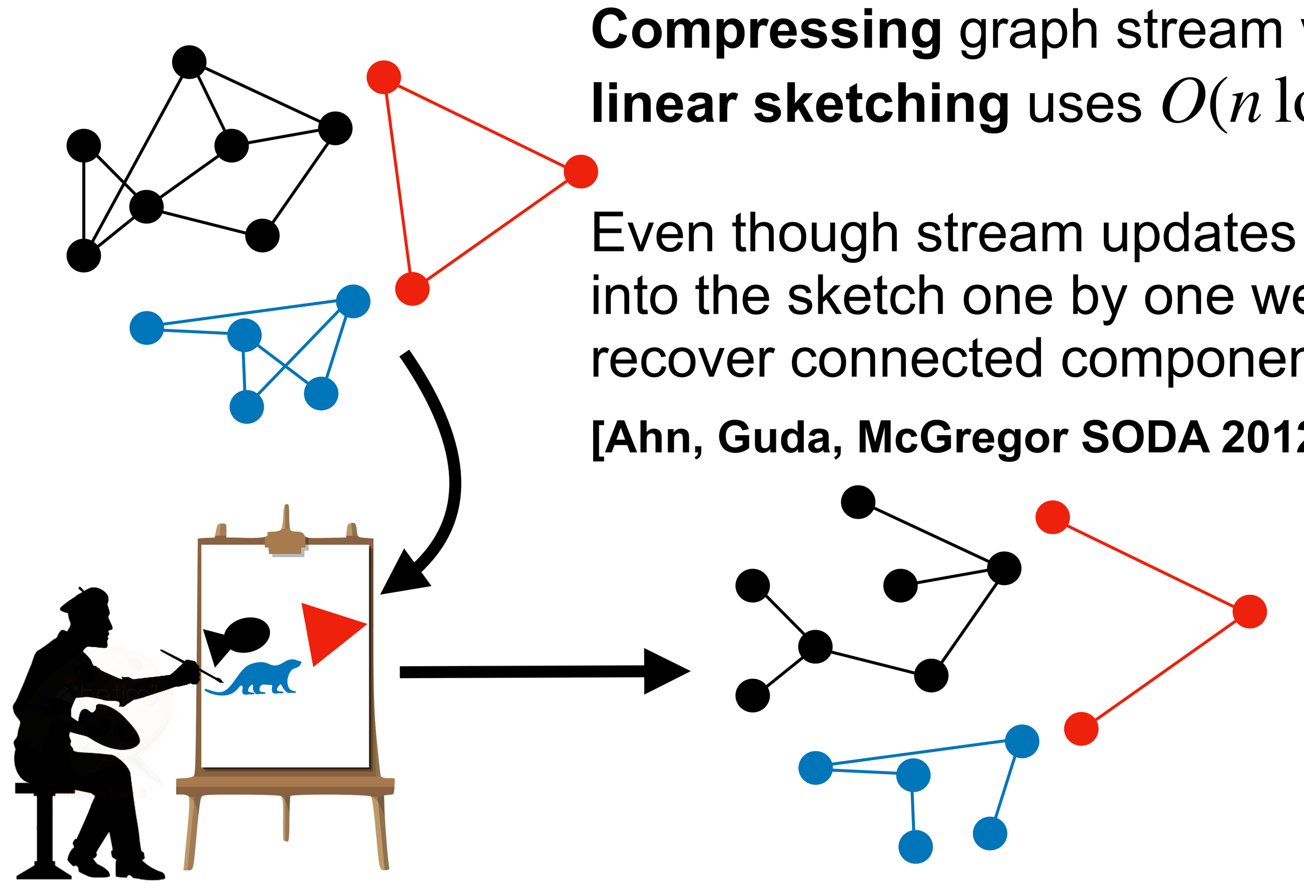
Analyzing Evolving Graphs

Example Problem: Find the connected components of a graph with n vertices subject to a **stream of edge insertions and deletions.**

Semi-Streaming Constraint: $O(n\ \mathrm{polylog}(n))$ space.



Graph Sketching Theory



Compressing graph stream via **linear sketching** uses $O(n \log^3 n)$ **space**.

Even though stream updates are compressed into the sketch one by one we can still recover connected components **w.h.p**

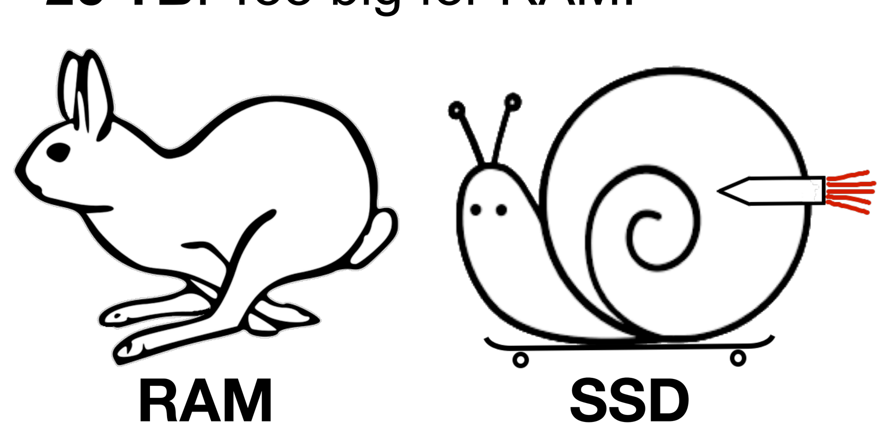
[Ahn, Guda, McGregor SODA 2012]

Sketching Theory in Practice

For a graph with a billion vertices, before considering constants $10^9 \times \log^3(10^9)$ = **25 TB**. Too big for RAM!

Modern SSDs bandwidth is approaching that of RAM but latency is high.

Can we get sketching to work on disk — without being slow?



Efficient Graph-Sketching

GraphZeppelin: C++ Library for finding the connected components of a graph stream.

Simultaneously an **optimal external memory** and **space optimal algorithm** for connected components

Graf-Zeppelin, **NOT** the Hindenburg, did not explode

GraphZeppelin uses **CubeSketch**, a new linear sketching algorithm, it is faster and more compact than AGM's algorithm.

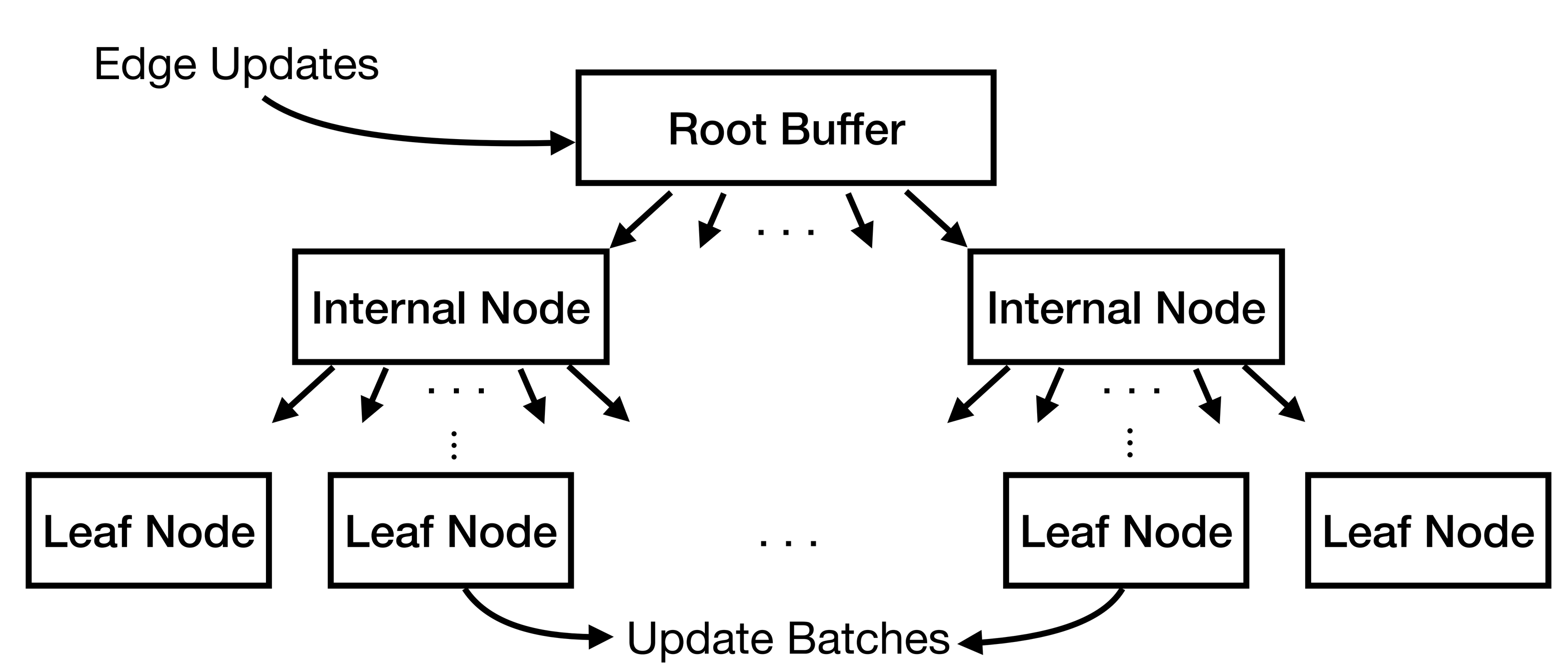
GraphZeppelin's memory usage scales with the number of vertices not the number of edges. We do best when the graph is **dense**

External Memory Data-Structures

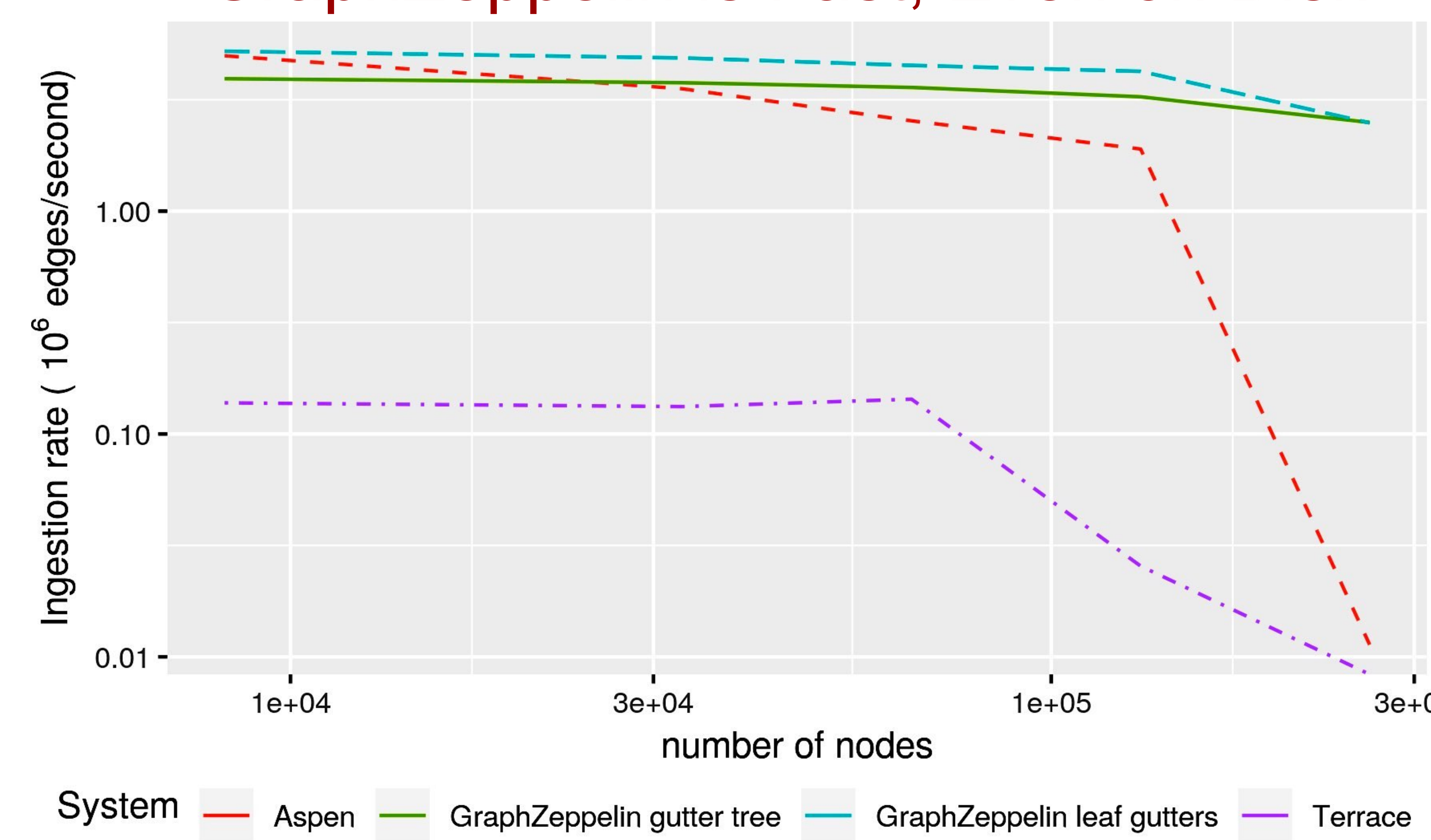
GraphZeppelin uses a **Gutter-Tree** to efficiently buffer updates in external memory.

Buffering updates allows us to keep sketches on disk with minimal performance impact.

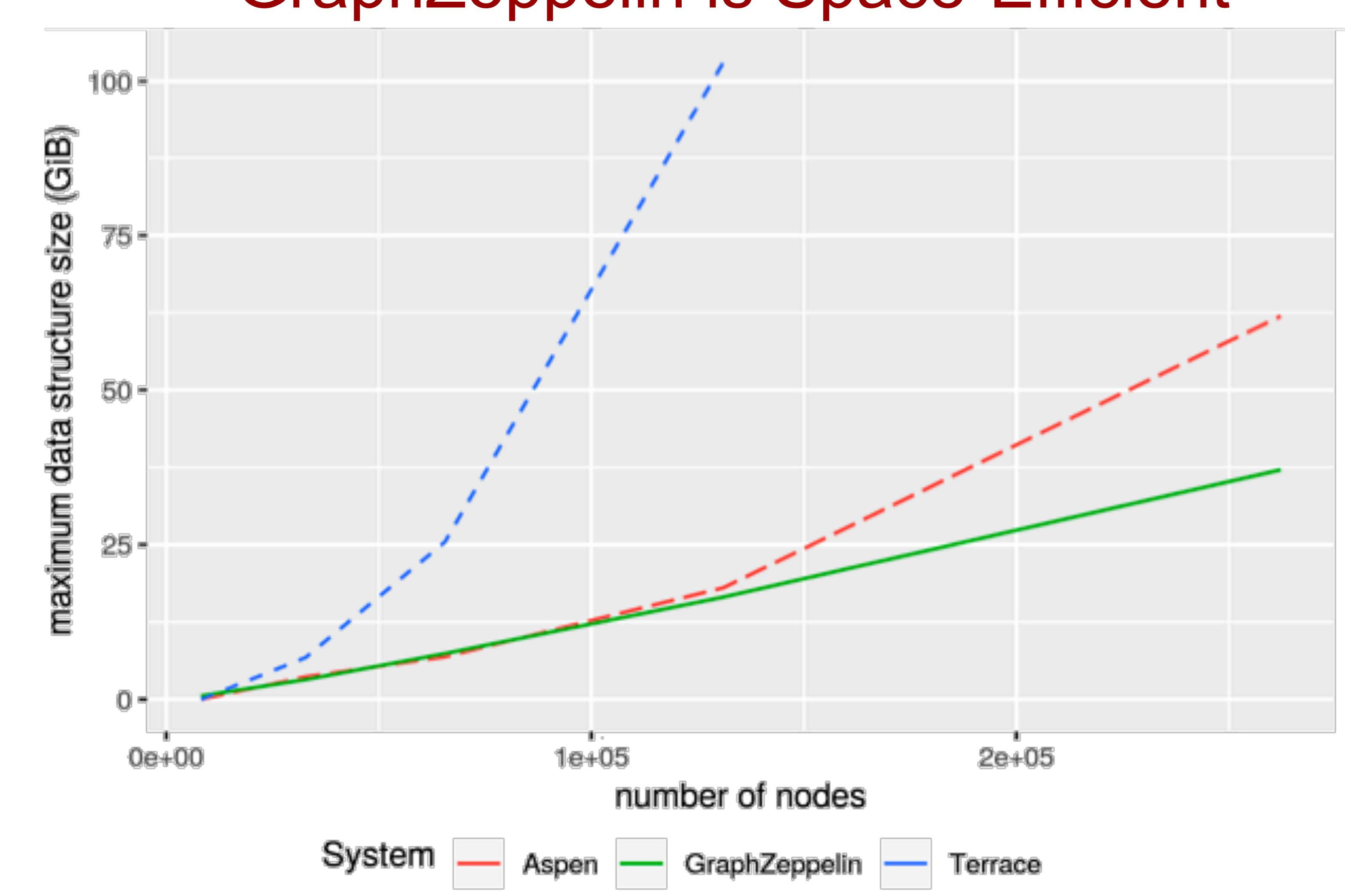
Each time we update a sketch we apply a large batch of updates to **amortize the cost of accessing the disk.**



GraphZeppelin is Fast, Even on Disk



GraphZeppelin is Space-Efficient



Implications

Graph streaming algorithms should be designed as external memory algorithms.

These new algorithms have the ability to outperform the state of the art.