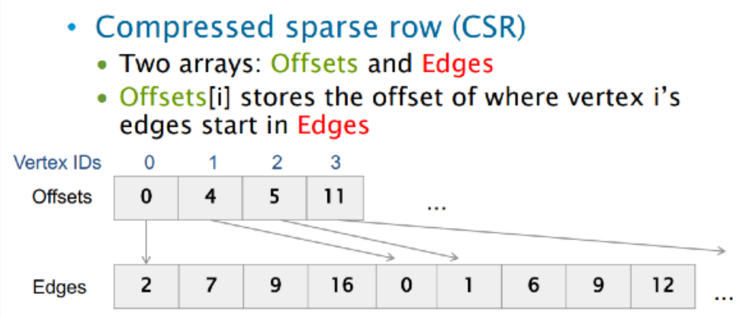
LLAMA: Efficient Graph Analytics Using Large Multiversioned Arrays

Macko, Marathe, Margo, Seltzer (2015)

Presented by Edward Fan

Compressed Sparse Row (CSR)

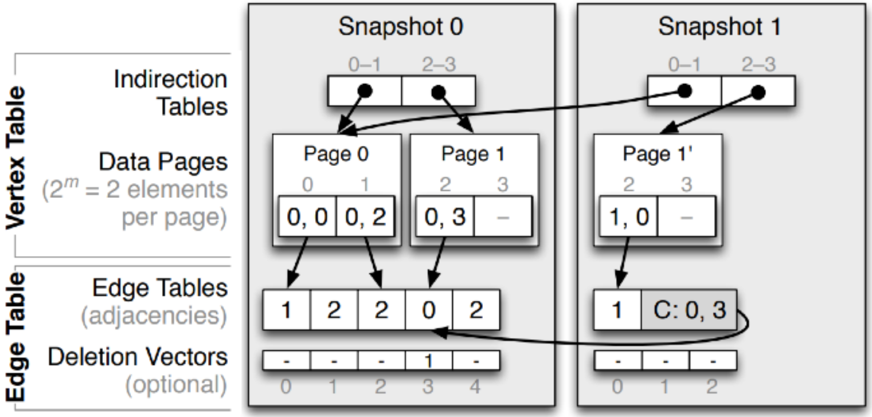
- Space efficient
- Performant, especially for vertex-centric computation
  - Authors find it to be much faster than adjacency lists or bitmaps
  - Excellent cache behavior; sorting leads to sequential access
- Problem: immutability
  - Can cache updates via delta map or use as log, but both require rebuilds



	Adjacency matrix	Edge list	Adjacency list (linked list)	Compressed sparse row
Storage cost / scanning whole graph	$O(n^2)$	O(m)	O(m+n)	O(m+n)
Add edge	O(1)	O(1)	O(1)	O(m+n)
Delete edge from vertex v	O(1)	O(m)	O(deg(v))	O(m+n)
Finding all neighbors of a vertex v	O(n)	O(m)	O(deg(v))	O(deg(v))
Finding if w is a neighbor of v	O(1)	O(m)	O(deg(v))	O(deg(v))

LLAMA

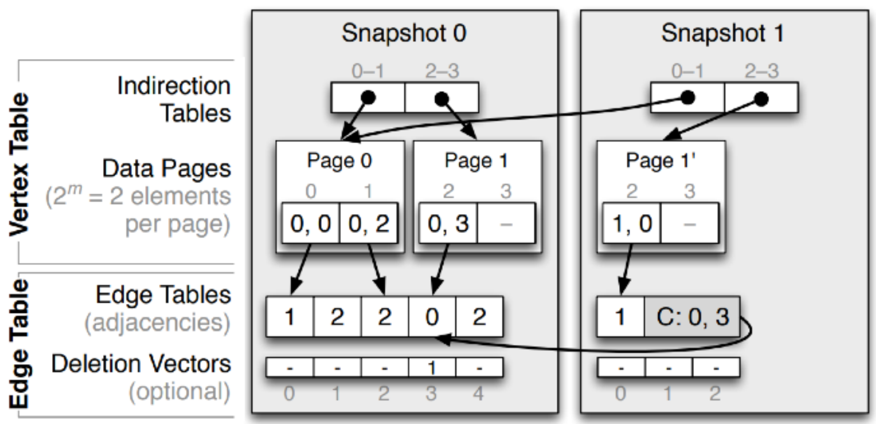
- Like CSR, but split across snapshots for mutability
- One *vertex table*, split *edge table*



(b) LLAMA Representation

Vertex Table

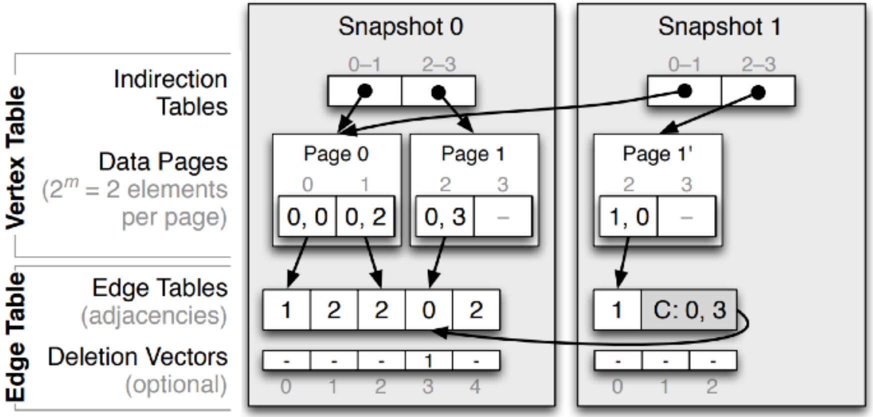
- One indirection table per snapshot
- Data pages contain vertex data, including offsets into edge tables



(b) LLAMA Representation

Edge Table

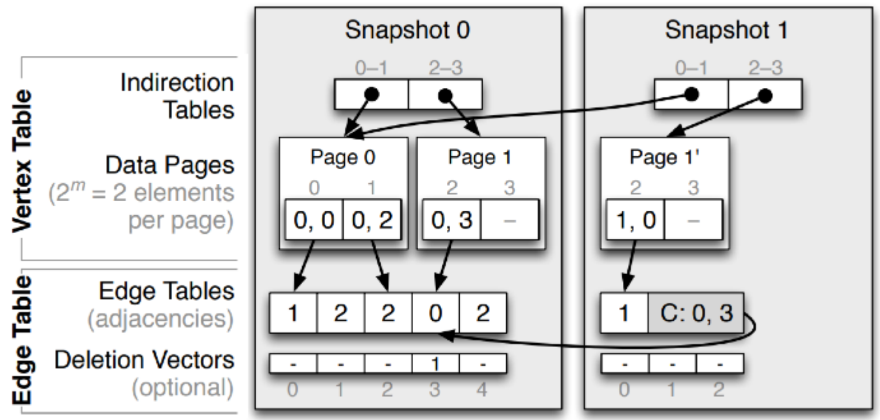
- *Continuation records* avoid duplication of entries
    - Authors tried simply copying the adjacency list, but memory size is an issue



(b) LLAMA Representation

Deletions

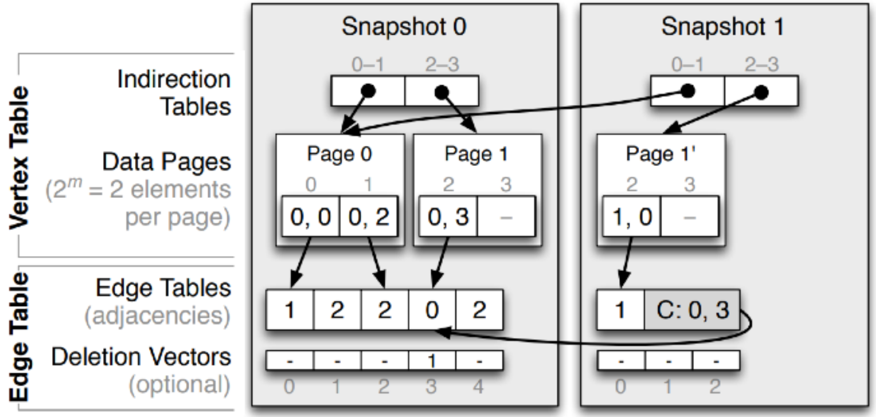
- Option 1: Use *deletion vectors* to logically mark edges as deleted
  - Can use upper bits or parallel array
- Option 2: Make copy of adjacency list with null continuation record



(b) LLAMA Representation

Miscellany

- Merging snapshots: simple traversal and formation of new LLAMA
- Incoming updates: buffered in write-optimized delta map, but not used for computation



(b) LLAMA Representation

Memory Management

- LLAMA designed to provide in-memory performance for graphs larger than memory

- Snapshots stored in files (16 snapshots per file)

- Could manage paging manually
    - Reference counting
    - Hazard pointers
    - Automatic garbage collection

- Instead, use mmap() and allow OS to manage pages
    - Almost no overhead when in memory
    - madvise(), mlock() can provide more advanced support

Performance

- On commodity machine (4 cores, 8GB RAM + SSD):
- In-memory: competitive with in-memory frameworks
- Out-of-core: significant improvements over GraphChi

System	Load	PageRank	BFS	TC
LLAMA	7.74	6.48	0.35	9.97
GraphLab	48.80	24.30	6.60	21.02
GreenMarl	6.75	5.30	0.27	9.79
GraphChi	26.00	39.54	38.84	45.81
X-Stream	–	12.74	5.65	–

(a) LiveJournal (in memory, 4 cores)

System	Load	PageRank	BFS	TC
LLAMA	311.1	607.6	233.8	2875.0
GraphLab	–	–	–	–
GreenMarl	–	–	–	–
GraphChi	760.5	1260.9	1334.9	3975.2
X-Stream	–	1942.9	1124.7	–

(b) Twitter (larger than memory, 4 cores)

Performance

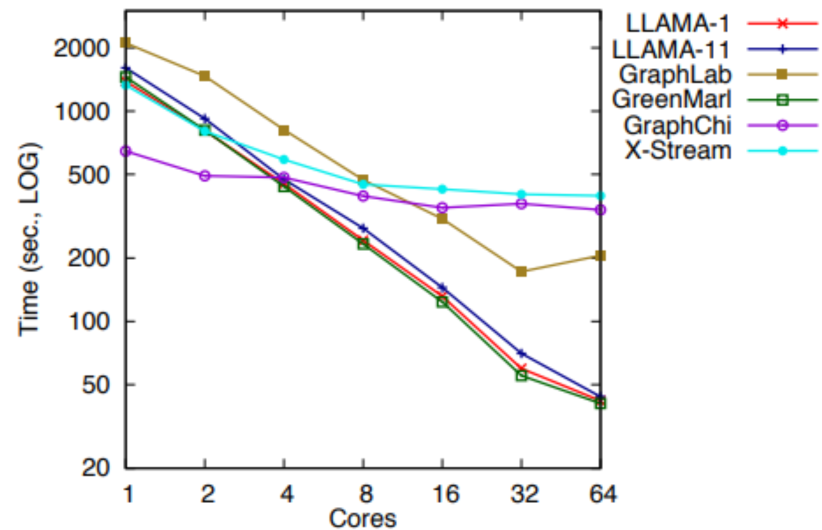
- LLAMA is CPU-bound, while GraphChi is I/O-bound
- Note: comparison with X-Stream is flawed, as LLAMA's time does not include load phase

	Time (s)			CPU Time Breakdown (%)			I/O (GB)	
System	**Wall**	**CPU**	**CPU%**	**PageRank**	**Buffer Mgmt.**	**Other**	**Read**	**Write**
LLAMA	607.6	1088.5	179	98.0	< 0.1	2.0	118.4	0.0
GraphChi	1260.9	3463.3	274	11.9	86.6	1.5	38.7	0.2
X-Stream	1942.9	7746.2	398	24.8	27.0	48.2	306.3	121.0

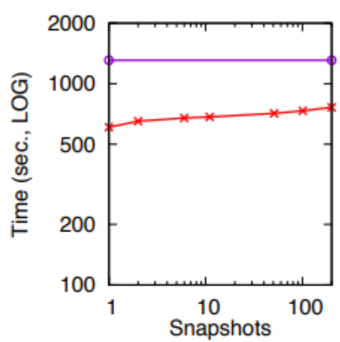
TABLE III: **PageRank on Twitter: Performance Breakdown** on the Commodity platform.

Performance

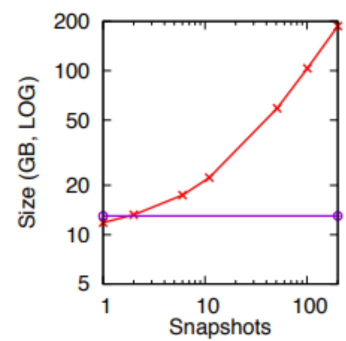
- Good scalability with more cores
- More snapshots have small effect on runtime, but take up more memory
    - Need to garbage collect and merge often



(b) PageRank



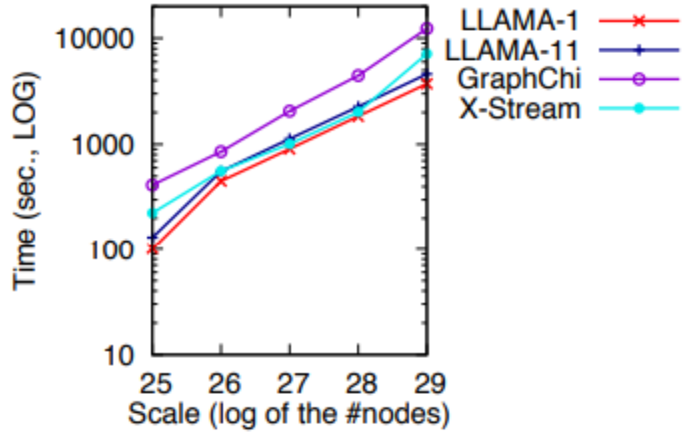
(b) PageRank (Time)



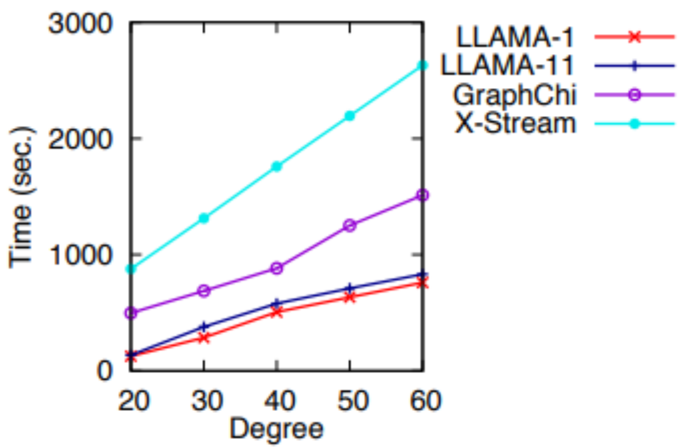
(b) Database Size

Performance

- Varying vertex count and degree does not change results



(c) PageRank



(c) PageRank

Limitations

- Difficult to use; standalone C++ library with open programming model
    - Can use GAS-like models, but very underspecified
- Missing components necessary for real-world use
    - Garbage collection not specified; no automatic GC in C++, need way of detecting when old snapshots are no longer being accessed
    - Parallel algorithms left mostly up to programmer (OpenMP)
- Project is dead
    - Code available on GitHub (https://github.com/goatdb/llama), but no commits since 2014
    - RAM cheaper than programmers

Questions?

Thanks!