Mutable Multilevel CSR Representation for Graph Databases

Peter Macko
Harvard University
& Oracle Corporation

The Two Sides

Whole-Graph Analysis

- Grace [ATC ‘12]
- GraphChi [OSDI ‘12]
- GraphLab
- Green-Marl
- Pregel [SIGMOD ‘10]
- Trinity [SIGMOD ‘13]

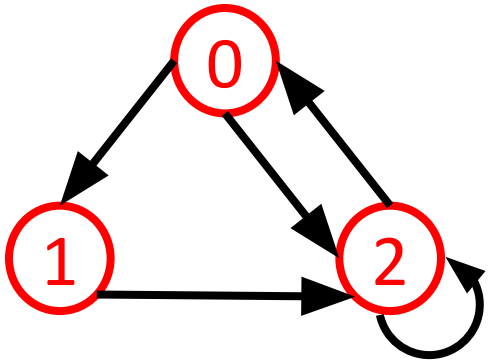
…

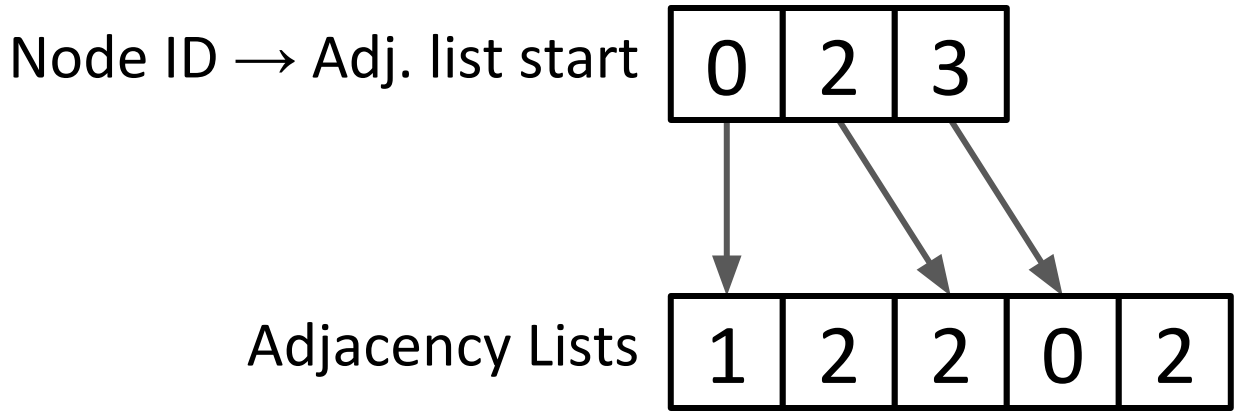
OLTP

- DEX [IDEAS ‘12]
- Neo4j

Can we do well at both?

Compressed Sparse Row (CSR)

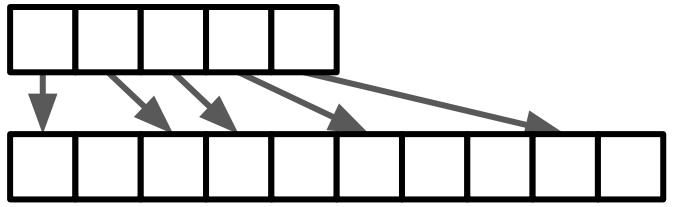




- Used by most analysis approaches
- All adjacency lists are stored in a single array, which is great for cache locality
- There are tricks to make this writable – usually at the cost of having to rebuild the data structure in order to get good read performance

What we are thinking…

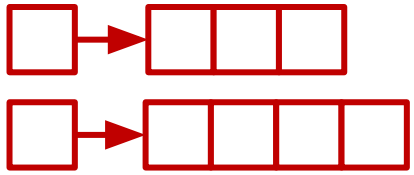
Writable CSR Representation



What we are thinking…

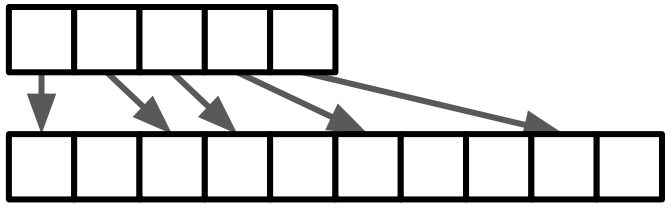
OLTP

Writable Graph Representation



Clustering

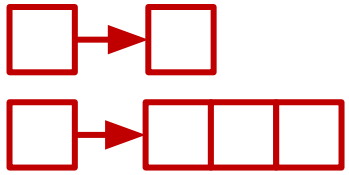
Read-Only CSR Representation



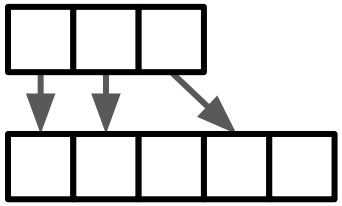
What we are thinking…

OLTP

Writable Graph Representation

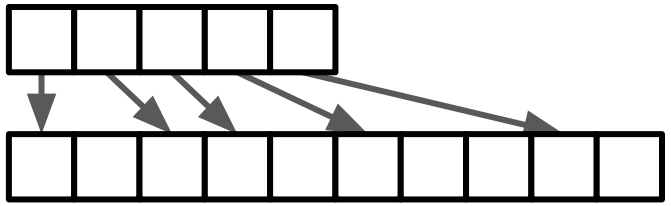


Level 1 CSR



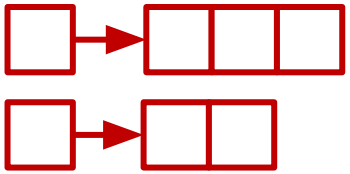
Clustering

Read-Only CSR Representation

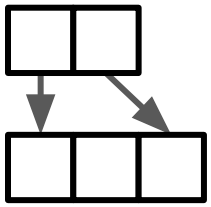


What we are thinking…

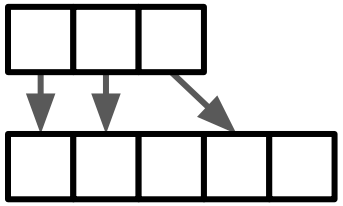
OLTP → Writable Graph Representation



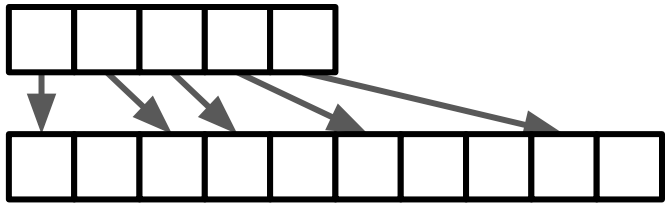
PageRank → Level 2 CSR



Level 1 CSR

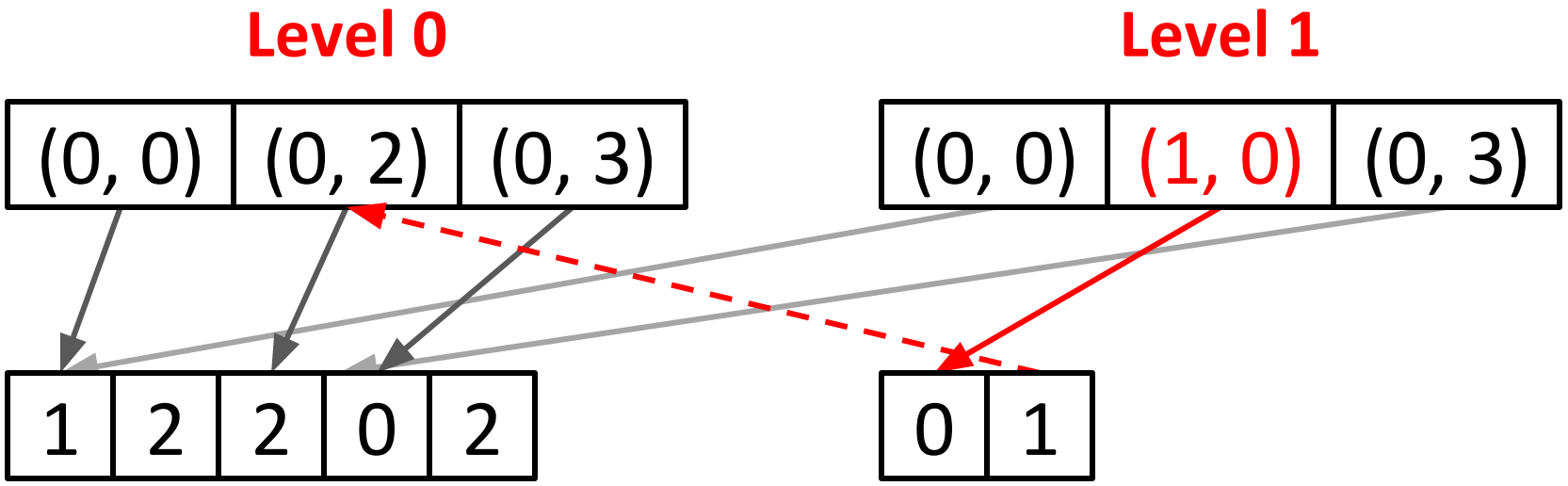


Clustering → Level 0 CSR



Challenges

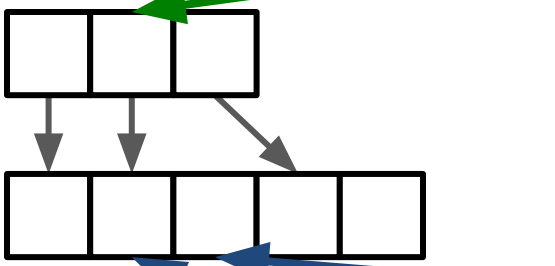
How do you represent the vertex map?



- (0, 2) means Level 0, array index 2
- How do you make it memory efficient?
  - Segment trees? COW? Continuations?

Challenges

How do you determine the end of an adjacency list?



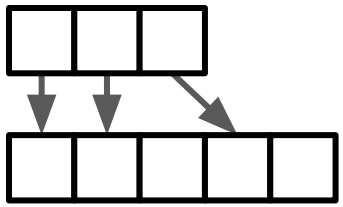
Store its length in the vertex map: 1.5% overhead

Set a bit on the last entry: 12.8% overhead

Store its length at the beginning of each adj. list: 10.1% overhead

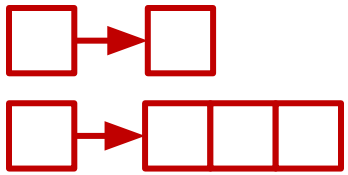
- Triangle counting on a graph with 10 mil. nodes, 50 mil. edges
- Quad-core Intel Core i5 3.30GHz, 16 GB RAM (1333 MHz)
- Overheads relative to a standard read-only CSR implementation

Challenges



What is a good representation for the intermediate CSR levels?

- Memory-efficient representation?
- Ends of adjacency lists?
- Deletions?
  - Deletion vectors? Normal LSM-like approach?
- Properties?



What is a good representation for the writable representation?

Questions? Feedback?

pmacko@eecs.harvard.edu

Preliminary Results

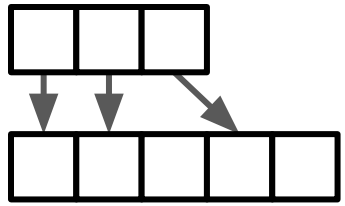
- Benchmarked triangle counting (by itself), measured overhead relative to a standard, read-only CSR
- 10 mil. nodes, 50 mil. edges, quad-core Intel Core i5 3.30GHz, 16 GB RAM

~20.9% overhead

Triangle Counting

1459 ms

Level 1 CSR



~1.5% overhead

Triangle Counting

1225 ms

Level 0 CSR