EdgeFrame: scalable worst-case optimal joins for graph-pattern matching in Spark

Presented by Per Fuchs

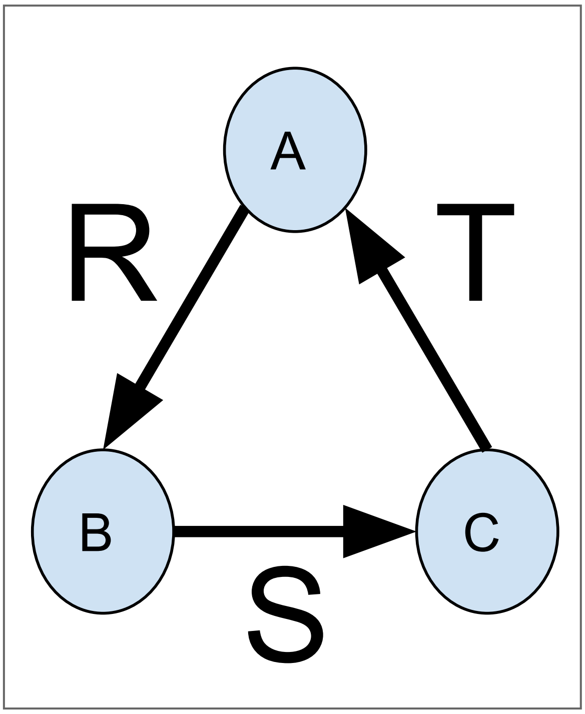
Supervised by Peter Boncz and Bogdan Ghit

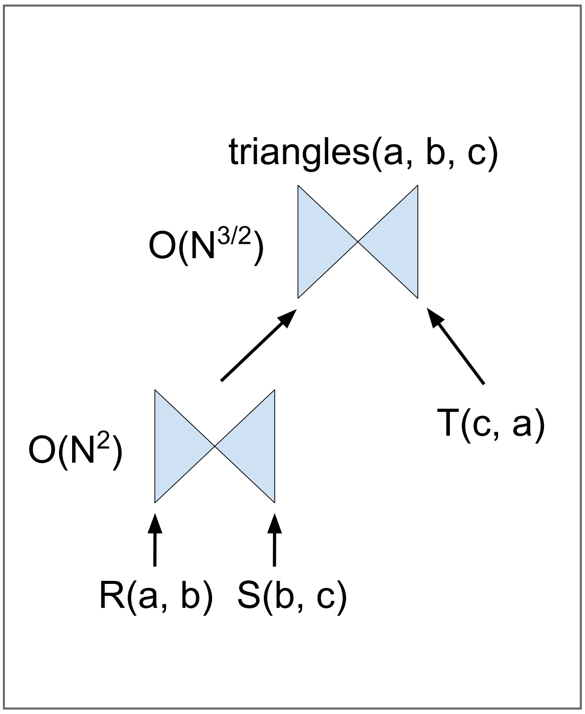
Master thesis in Computer Science

Visit **https://perfuchs.github.io/master-thesis-presentation/** for HTML version with correct layout.

PDF export of my presentation software is experimental!

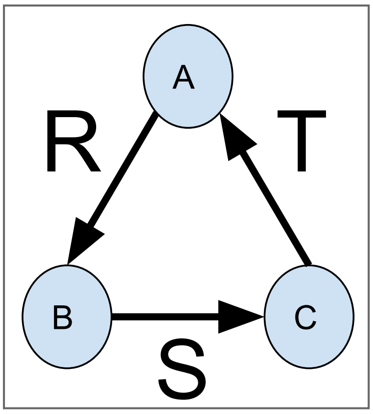
Cyclic queries in graph-pattern matching pose new challenges to relational engines





triangles(a, b, c) <- R(a, b), S(b, c), T(c, a)

Worst-case optimal joins to the rescue



- — proven to be worst-case optimal by AGM bound, e.g. for triangles in O($N^{3/2}$)
- — no intermediary results
- — Idea: build the join by a *variable-at-a-time* approach
- — superiority for graph-pattern matching is well established [1, 2, 3]

[1] Join Processing for Graph Patterns: An Old Dog with New Tricks, Dung Nguyen et al, Grades 2015

[2] From Theory to Practice: Efficient Join Query Evaluation in a Parallel Database System, Shumo Chu et al, Sigmod 2015

[3] Distributed Evaluation of Subgraph Queries Using Worstcase Optimal Low-Memory Dataflows, Khaled Ammar et al, VLDB 2018

Our contributions

1. designing a scalable WCOJ for Spark
   — Which distribution scheme to use?
   — open-source
   — integrate the WCOJ with Cypher on Apache Spark (stretch goal)
2. specializing WCOJ to graph pattern matching
   — former literature indicates that this is the main use case

1st contribution: designing a scalable WCOJ in Spark

Background: Spark

- Spark distributes data over workers
- computation is organized in exchanging steps of local computations and shuffles
- joins work by shuffling the data such that the distribution allows local join algorithms

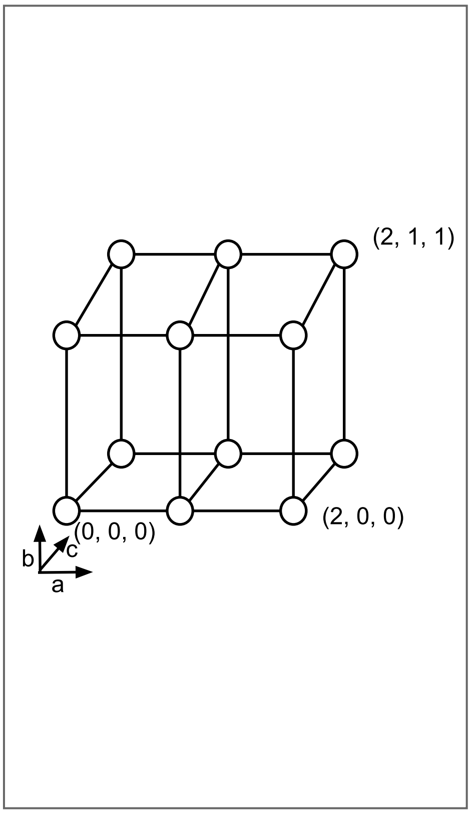
Hypercube shuffle: optimal distribution for n-ary joins

Idea[1]

- organize p workers in a hypercube
- one dimension per variable
- configurable k_i size per dimension
- such that $p = \prod_i k_i$
- proven to be communication optimal

[1] Optimizing Joins in a Map-Reduce Environment, Foto Afrati and Jeffrey Ullman, 2010

triangles(a, b, c) <- R(a, b), S(b, c), T(c, a)

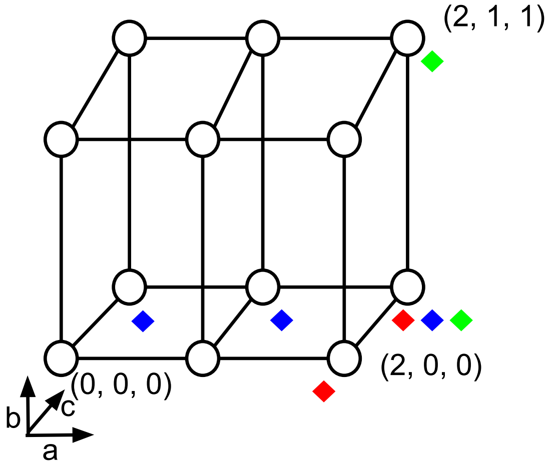


Hypercube shuffle: optimal distribution for n-ary joins

triangles(a, b, c) <- R(a, b), S(b, c), T(c, a)

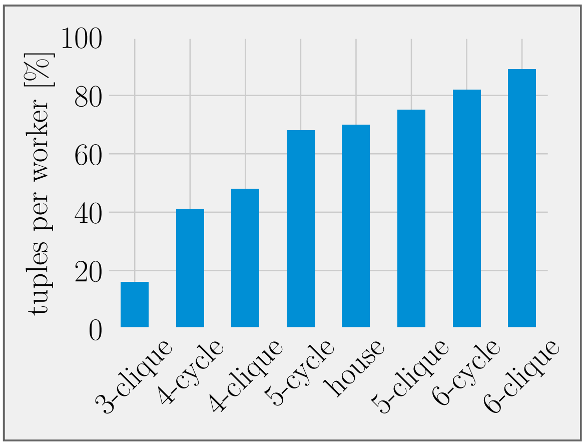
a	b	b	c	c	a
1	2	1	2	1	2
2	3	2	3	2	3
2	4	2	4	2	4
3	1	3	1	3	1

(2, 0, *) (*, 0, 1) (2, *, 1)



Hypercube shuffle converges to full replication for larger queries

- analysis by theoretic estimation and simulation
- a lot of duplicated work
- not scalable in query size
- although being optimal



Our Solution: replicated *EdgeFrame*

— DataFrame specialized for edge relationship
— replicated on all workers
— shuffle free worst case optimal join operation
— uses compressed sparse row representation
— easily integrable into existing Spark projects
— open source
— *logically partitioned* (open research)

Parallelization via logical partitionings

- parallelization via logical partitioning: full dataset is on each worker but each worker only considers parts of it
- partition on the first attribute to bind by the WCOJ
  - fight skew with Intel's adaptive query execution[1]

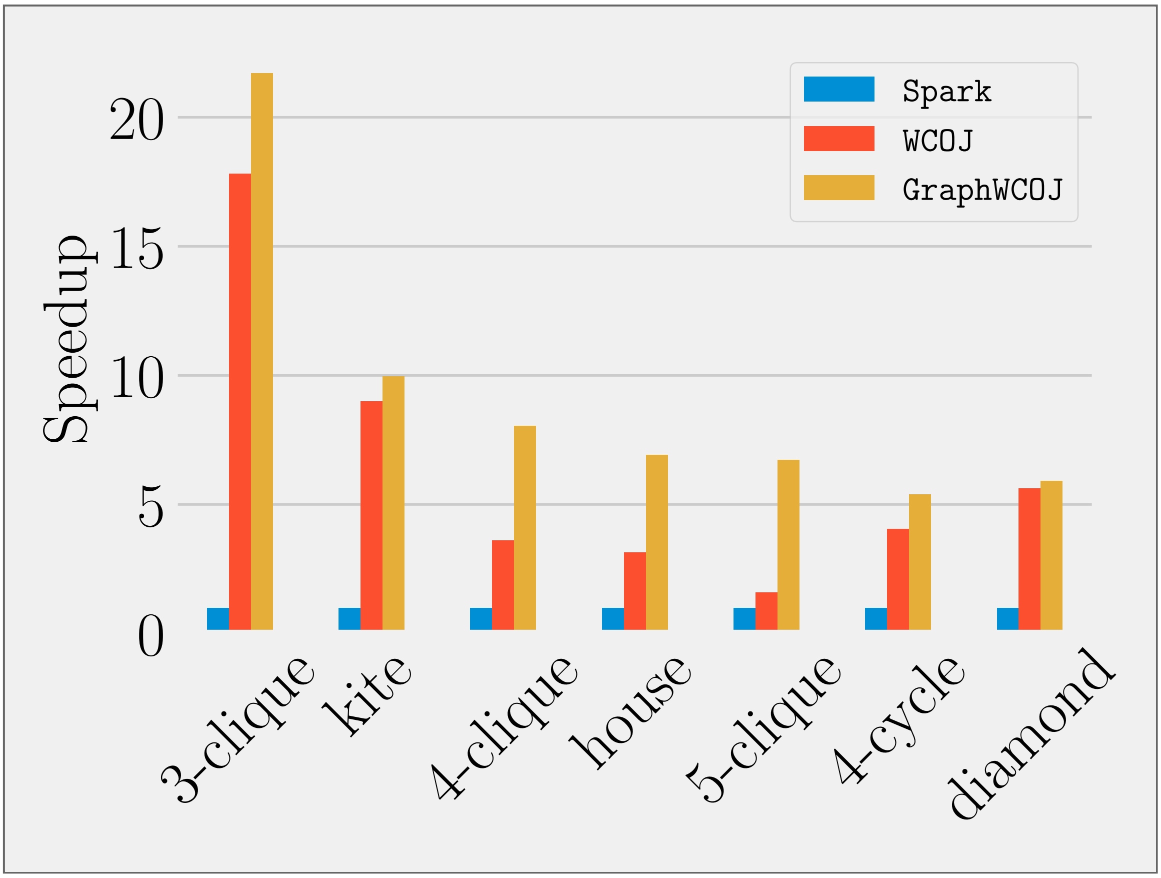
[1] Spark SQL Adaptive Execution at 100 TB, Carson Wang, 2018

2nd contribution: specializing WCOJ's to graph-pattern matching

Specializing WCOJ's to graph-pattern matching: idea

- backing data structure: compressed sparse row (CSR)
- code specialization
    - self-joins only
    - two attributes only
- logical optimizations

Specializing WCOJ's to graph-pattern matching: results



Where to find my work?

https://github.com/PerFuchs

Also, I'm looking for PhD opportunities or challenging positions in industry. Passionate about distributed systems and graphs!

Take aways

— optimal distribution scheme does not scale
— therefore, replicate
— WCOJ should be specialized to graphs
— open source

List of datasets

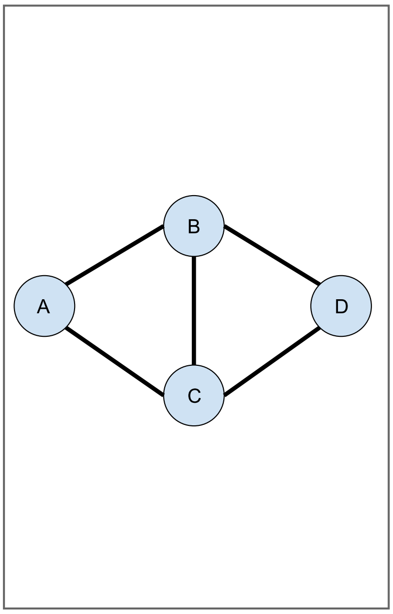
Name	Variant	Vertices	Edges
Social Network Benchmark[1]	scale factor 1	10,278	453,032
Amazon co-purchase[2]	2nd March	262,111	1,234,877
Twitter[2]	social-circles	81,306	2,420,766
Amazon co-purchase[2]	1st June	403,394	3,387,388

[1] The LDBC Social Network Benchmark: Interactive Workload, Orri Erling et al, 2015

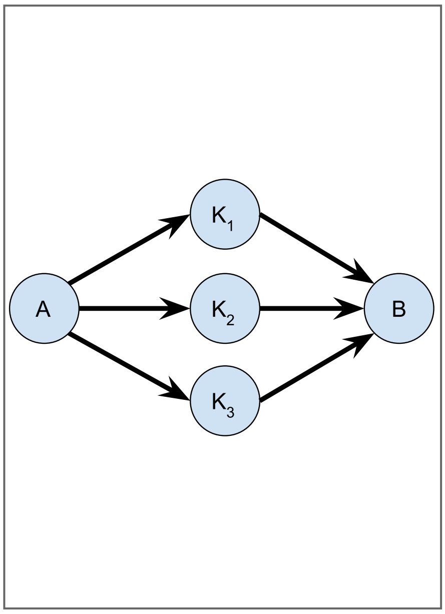
[2] SNAP Datasets: Stanford Large Network Dataset Collection, Jure Leskovec and Andrej Krevl, 2014

Why are cyclic patterns important?

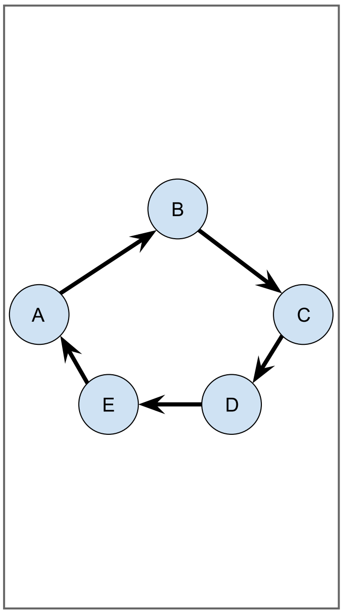
Facebook friends



Twitter followers[1]



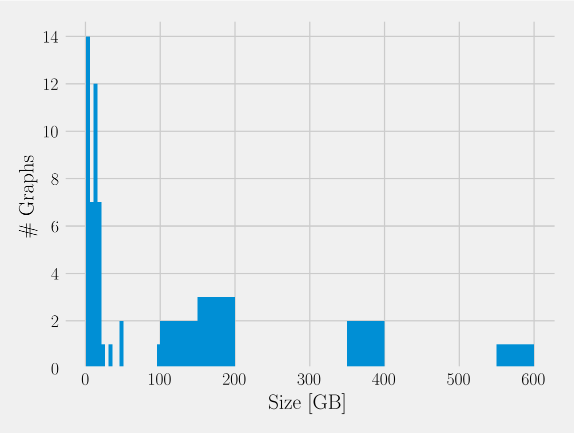
Bank fraud[2]



[1] Real-time twitter recommendation: online motif detection in large dynamic graphs, Pankaj Gupta et al, 2014

[2] Fraud detection: Discovering connections with graph databases, Gorka Sadowski and Philip Rathle, 2015, Whitepaper

Do graphs fit into main memory?



- study of openly available graph datasets
  - SNAP Datasets[1]
  - Laboratory for Web Algorithms[2]
- total number of graphs: 154
- all but 3 fit into 256GB of RAM
- maximum: 552 GB (Facebook 2011)

[1] SNAP Datasets: Stanford Large Network Dataset Collection, Jure Leskovec and Andrej Krevl, 2014

[2] The WebGraph Framework I: Compression Techniques, Paolo Boldi and Sebastiano Vigna, 2004