Tesseract: Distributed, General Graph Pattern Mining on Evolving Graphs

Laurent Bindschaedler
Jasmina Malicevic
Baptiste Lepers

Ashvin Goel
Willy Zwaenepoel

MIT CSAIL
Massachusetts Institute of Technology
swisscom
University of Toronto
FNSNF
The University of Sydney
EPFL

Graph Pattern Mining 101

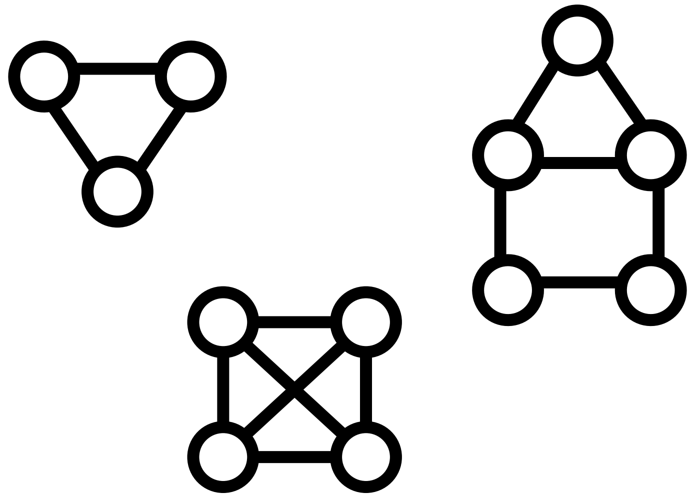
Find all instances of interesting subgraphs (patterns) in a graph

Each instance of a pattern is called a match

- Even small graphs can contain billions or trillions of matches

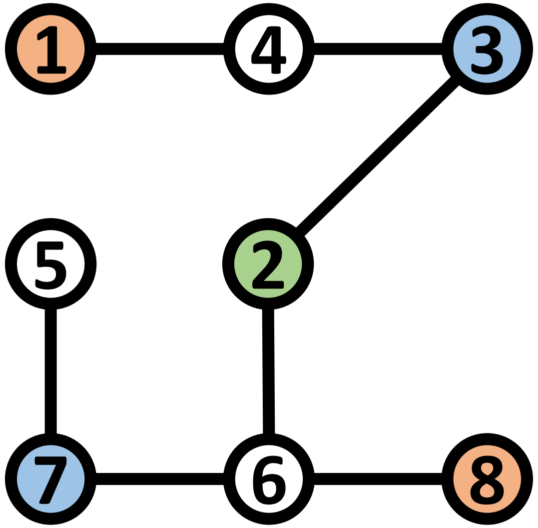
Graph Dataset

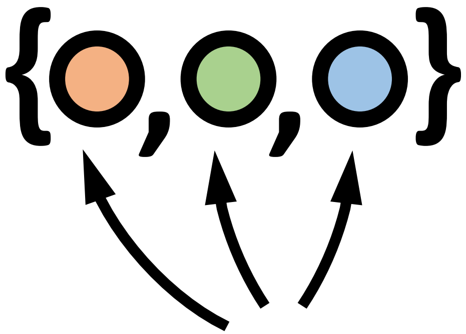
Patterns



Example – Graph Keyword Search

Input Graph

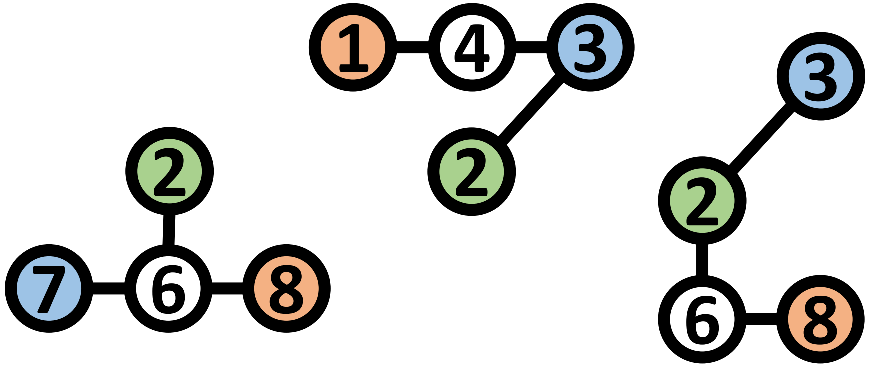


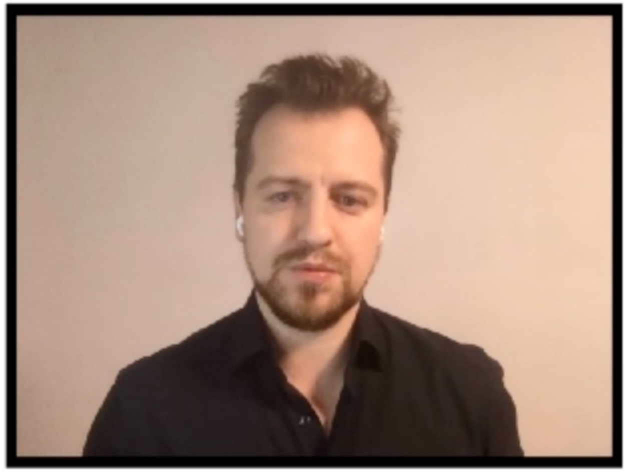


Pattern

Labels

Matches



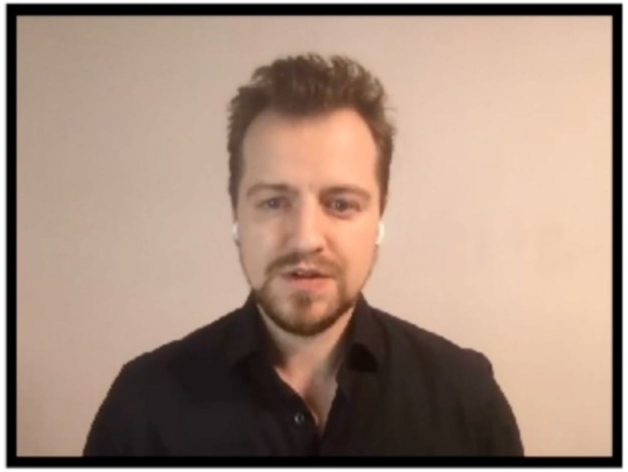


Graph Mining with Evolving Graphs

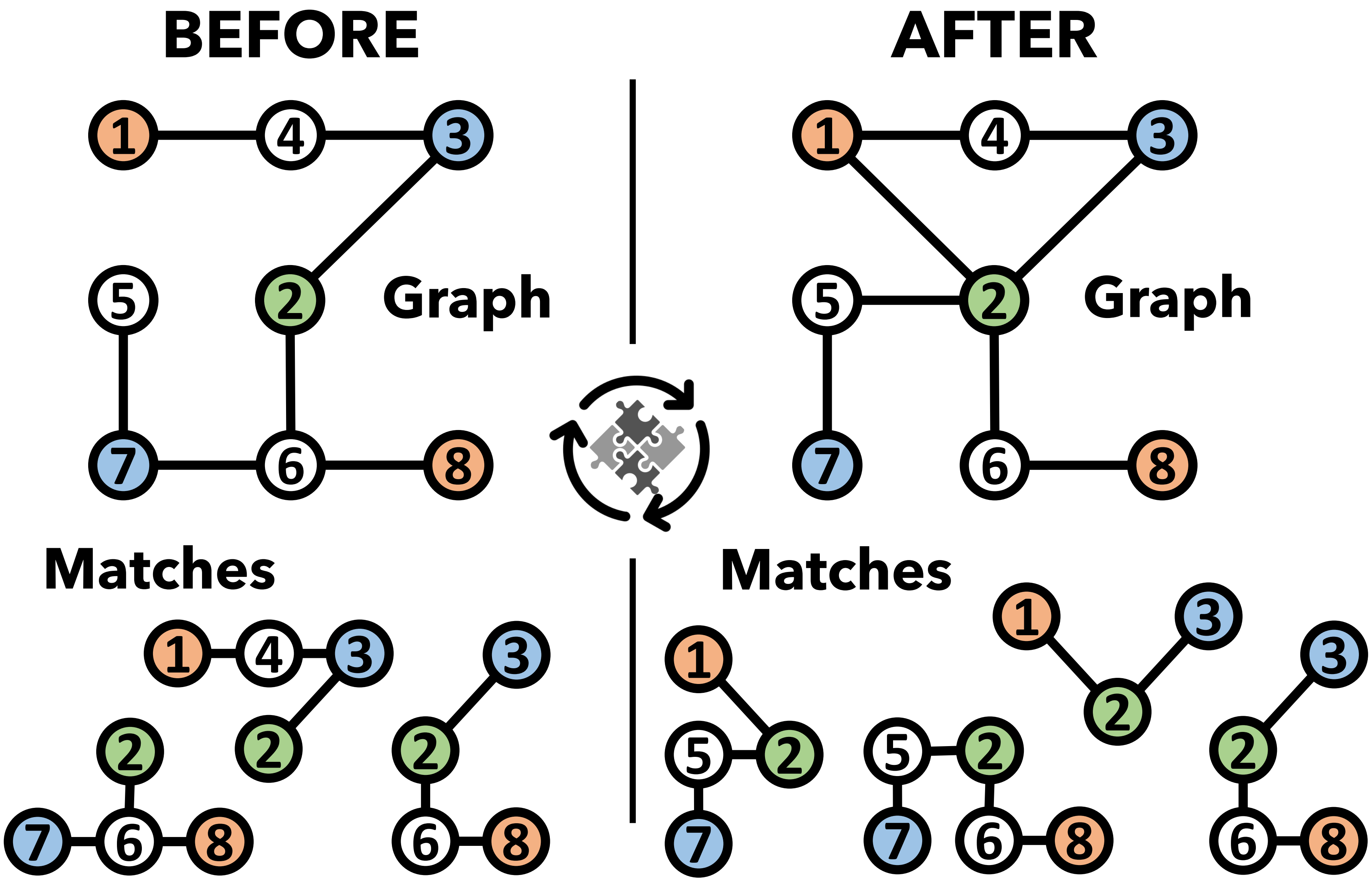
Find matches in a graph that receives a stream of updates
⇒ Incrementally maintain the match set so that it matches the input graph

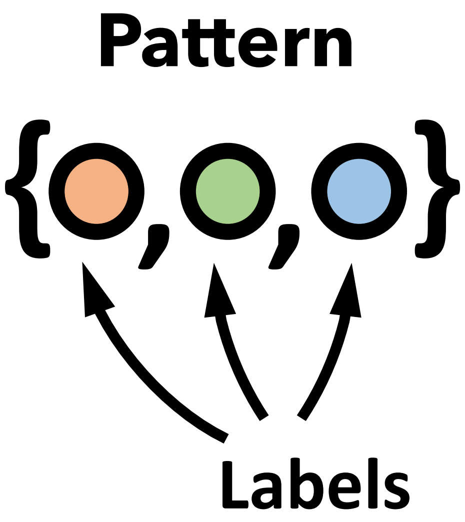
Challenges

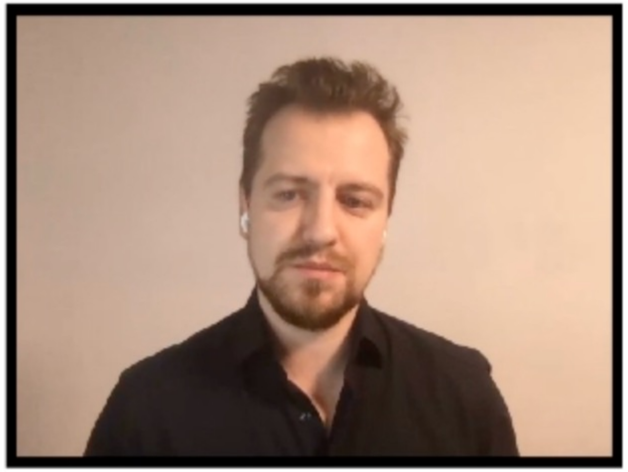
- Correctness
  - Matches in original graph + matches from updates = matches in new graph
  - Avoid exploring/outputting duplicate matches
- Efficiency
  - Expect millions of updates per second
  - Recomputation is out of the question
- Scale-out
  - Minimize data transfer between workers



Example – Incremental Graph Keyword Search







Tesseract: Evolving, Distributed, General

Tesseract incrementally mines **evolving** graphs

- Several thousands of times faster than naive complete computation

Tesseract supports **distributed** execution

Tesseract supports a **general** programming model

Tesseract is also faster for static graph mining

Key Ideas in Tesseract

Update-based exploration

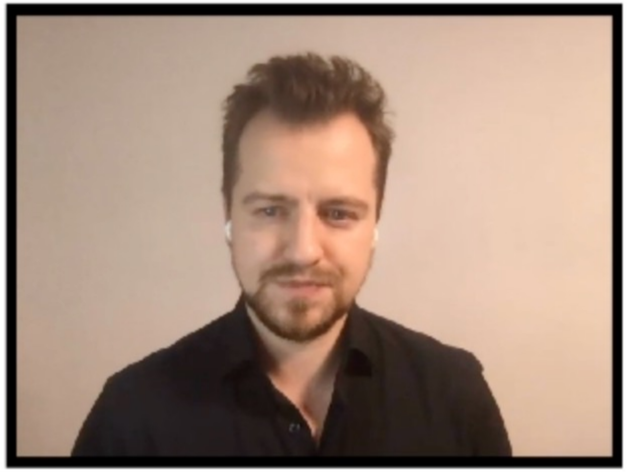
- Process one update at a time independently
- Find all corresponding changes to the match set using differential mining

Duplicate elimination

- Symmetry breaking
- Multiversioned graph store
- Snapshot-based exploration

⇒ No data exchange across workers required

⇒ No synchronization across workers required

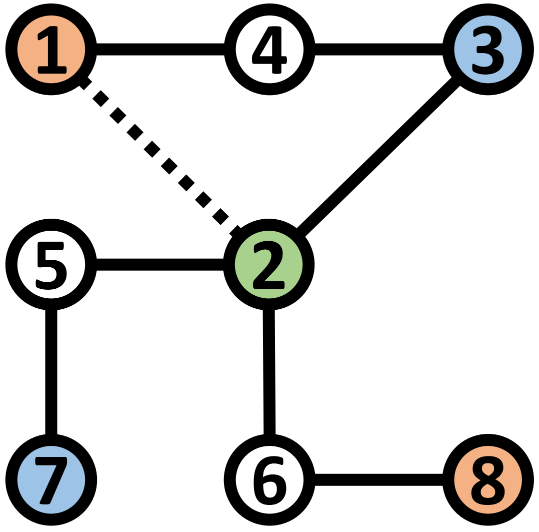


Update-Based Exploration

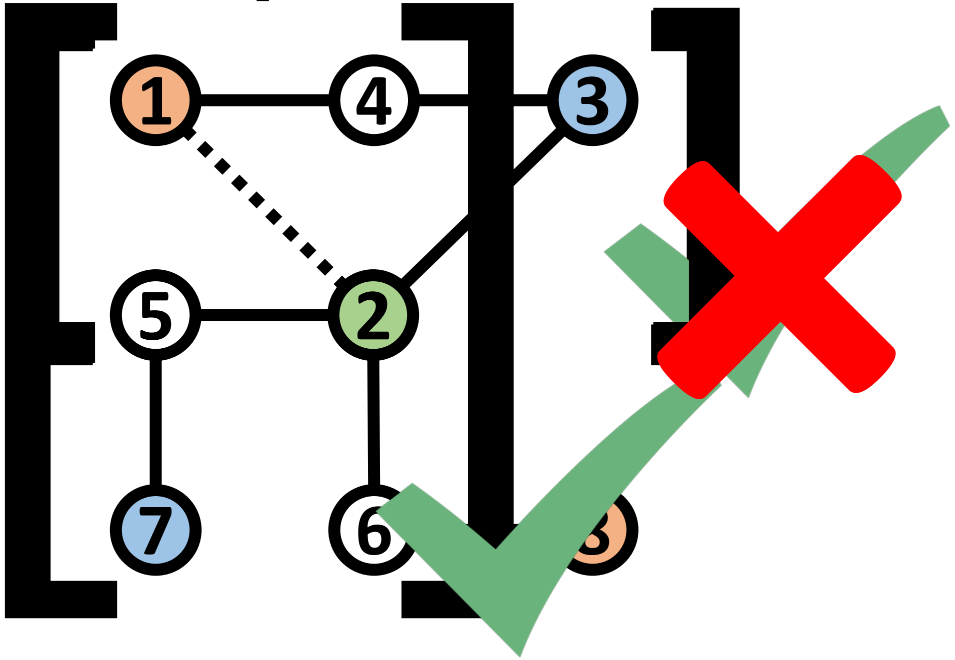
Each worker processes one update at a time

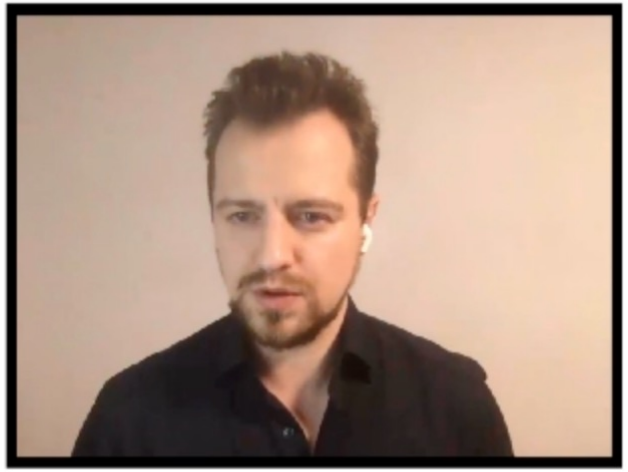
Exploration enumerates all subgraphs including the update

Input Graph



Exploration





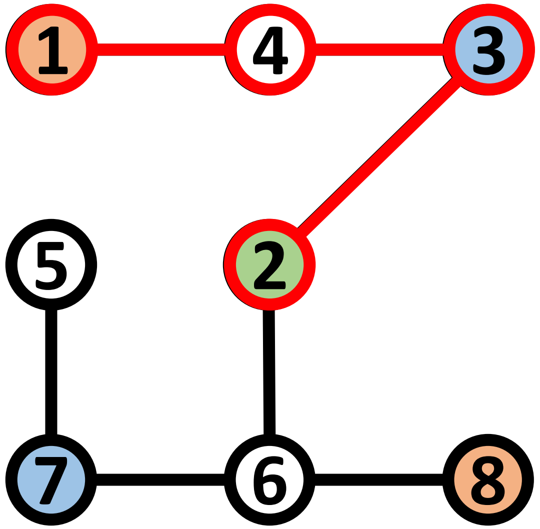
Differential Pattern Mining

Goal: find all changes involving the update being explored

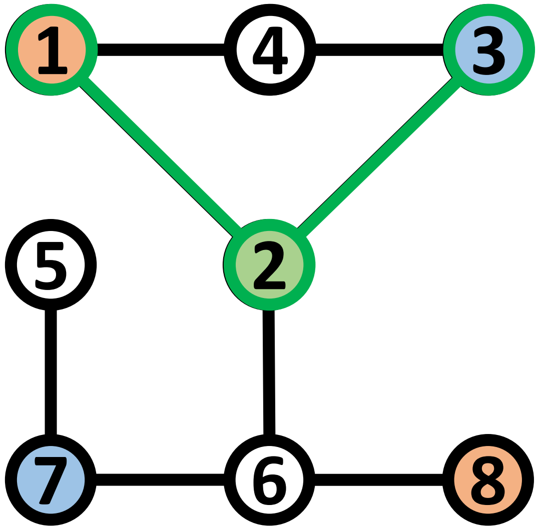
Solution: explore the pre-update and post-update graphs

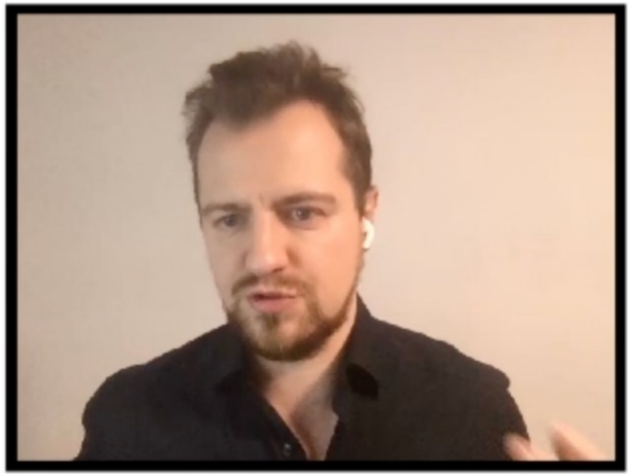
- Remove matches in pre-update graph
- Add matches in post-update graph

BEFORE UPDATE



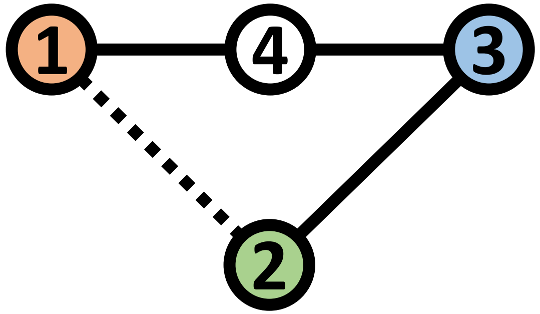
AFTER UPDATE



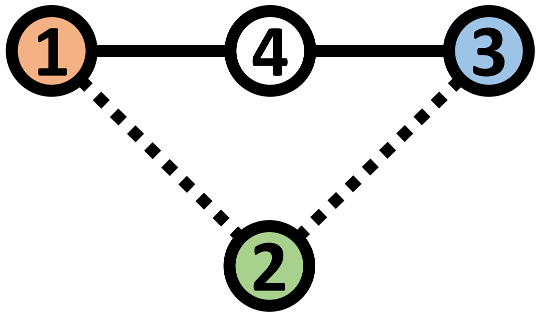


Exploration May Find Duplicates

Duplicates can occur from a single update or multiple updates



can be explored as (1, 2, 3, 4) or (1, 2, 4, 3)
⇒ duplicate match from a single update



can be found from (1, 2) and from (2, 3)
⇒ duplicate match from two updates

Avoiding Duplicates

Single update: use symmetry breaking

- Root exploration at update + enforce expansion order

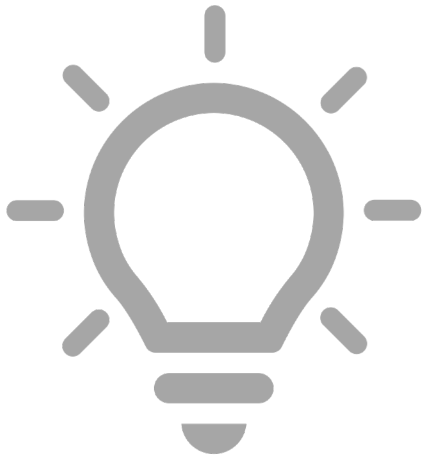
Multiple updates: leverage multiversioned graph store

- Explore each update at corresponding graph snapshot
- Total ordering based on update timestamps prevents “seeing the future”

Optimize exploration with snapshots containing multiple updates

- Decrease snapshot overhead
- Skip work for intermediate matches

Parallel Exploration Across Workers



No other changes necessary!

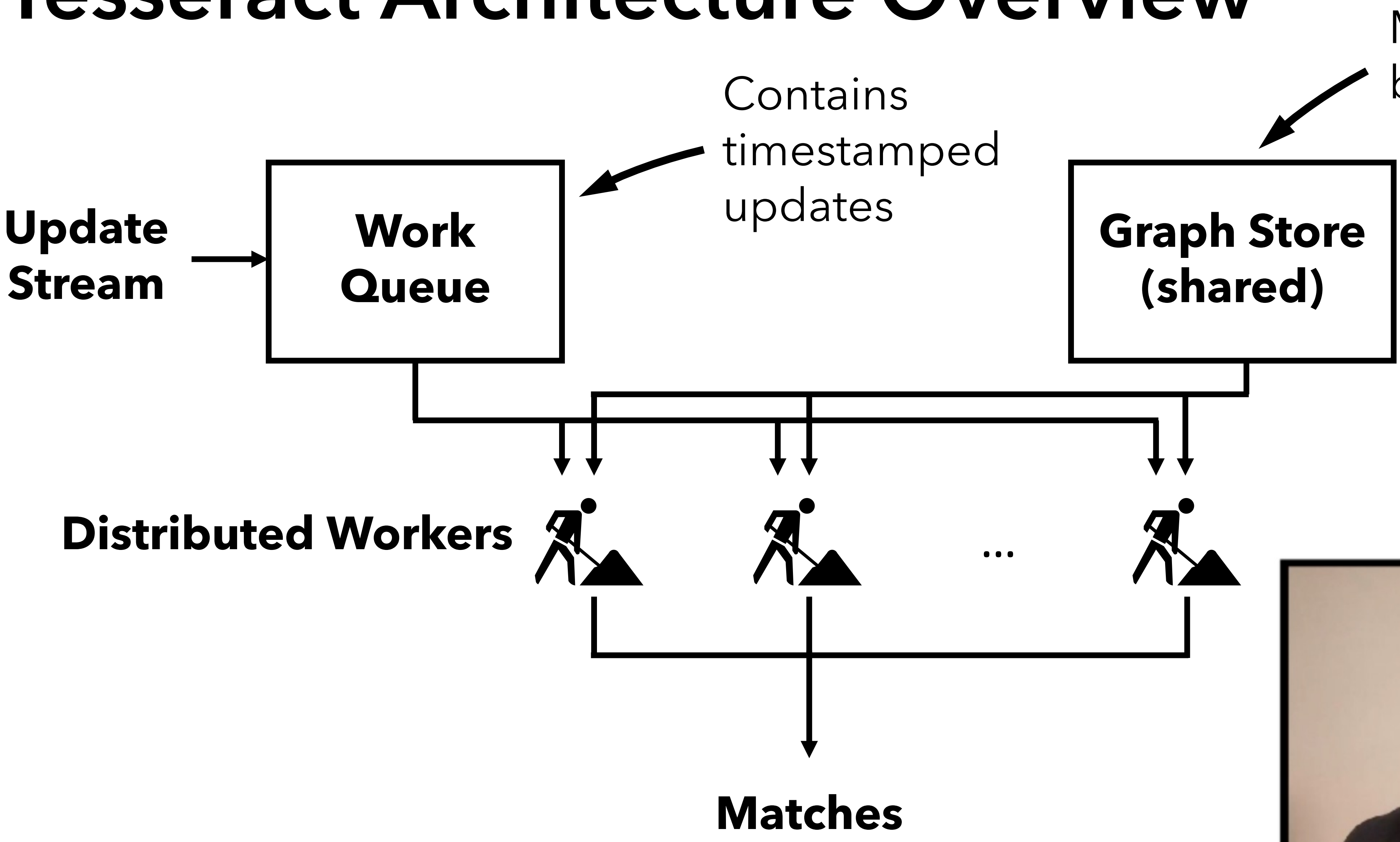
⇒ Each update is completely independent of the others

⇒ Updates can be processed in any order

⇒ Any update can be processed by any worker

(even updates in the same snapshot)

Tesseract Architecture Overview

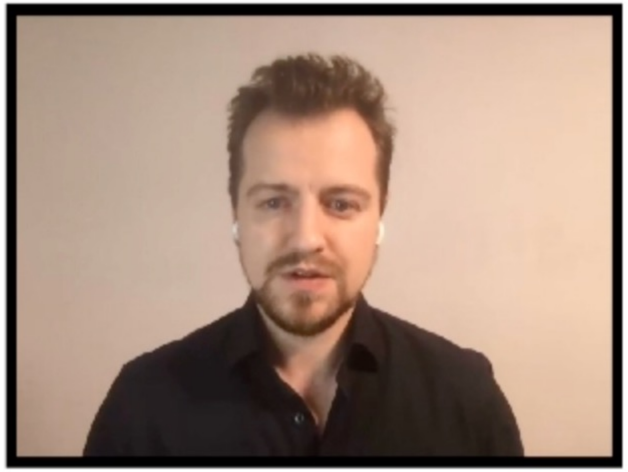


Evaluation

8 16-core machines (one rack)
128GB RAM, 2x500GB SSD, 40GigE switch

4-C: Clique Mining
4-CL: Labeled Clique Mining
4-MC: Motif Counting
5-GKS-3: Graph Keyword Search
4-FSM-2K: Frequent Subgraph Mining

LiveJournal (4M vertices, 68M edges)
UK-2007 (106M vertices, 3.7B edges)
DC-2012 (3.5B vertices, 128B edges)



Summary of Key Results

Tesseract is faster than naive computation from scratch

- 5X to 6000X faster runtime

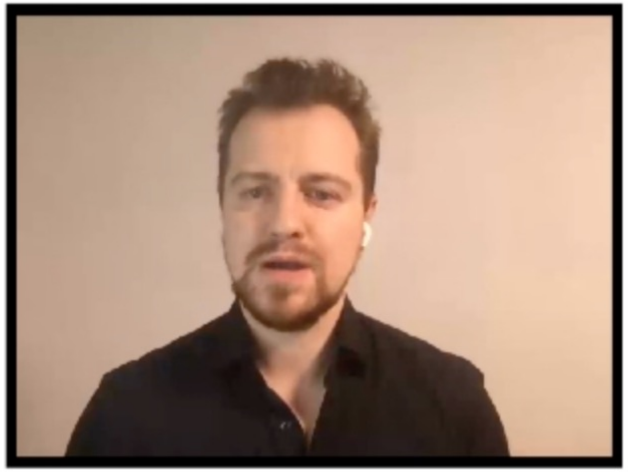
Tesseract outperforms subgraph query systems

- 1.1X to 12.3X faster than Delta-BigJoin

Tesseract is even faster on static graphs!

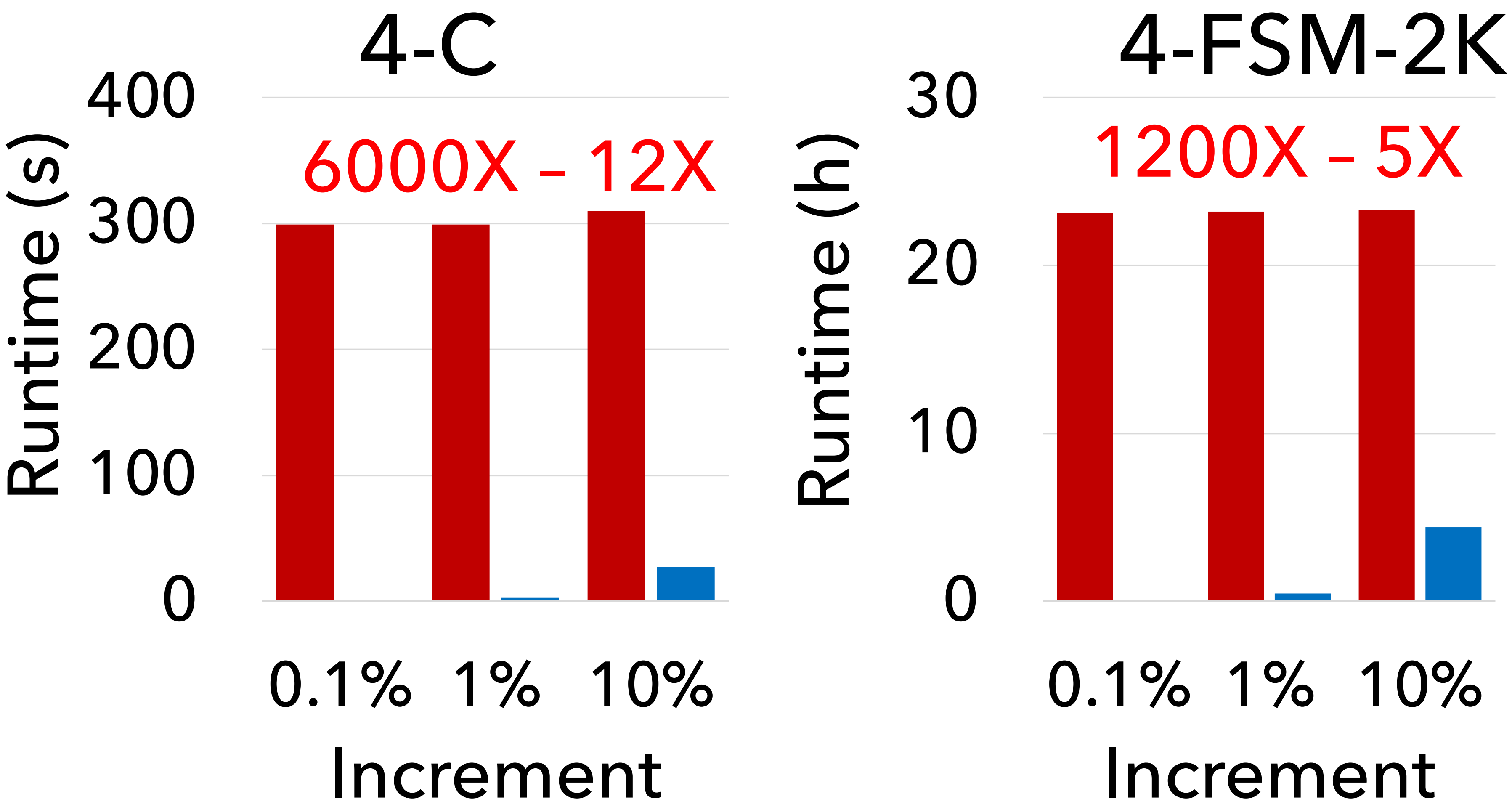
- 2X to 6X faster than static distributed mining systems

Tesseract can maintain matches in large graphs

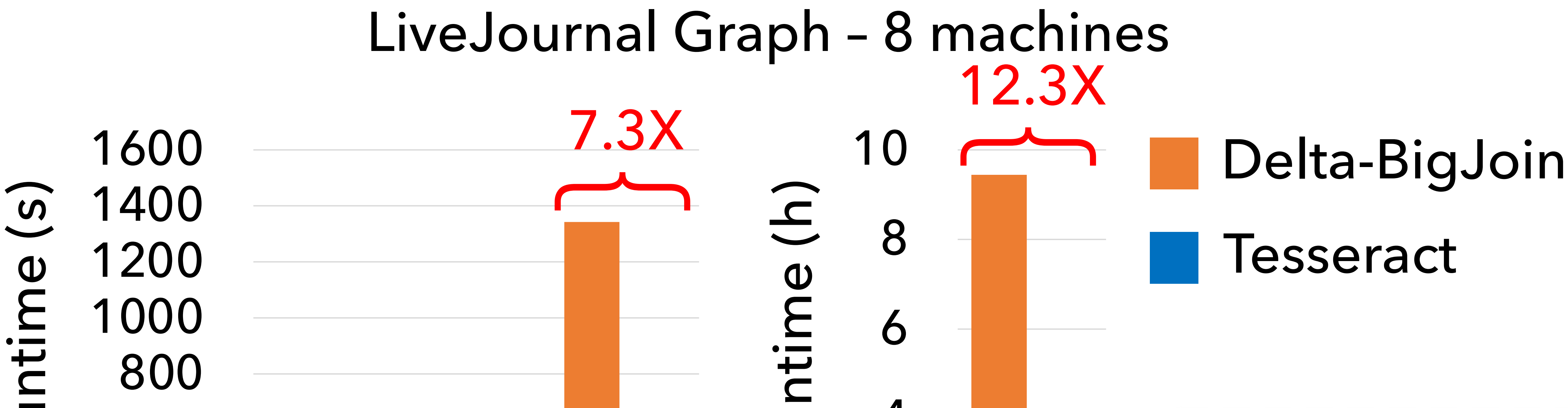


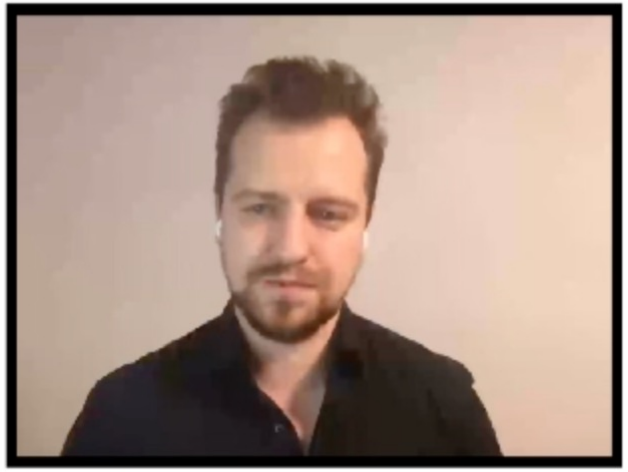
Incremental Computation Benefits

LiveJournal Graph – 8 machines



Comparison with Delta-BigJoin [VLDB'18]

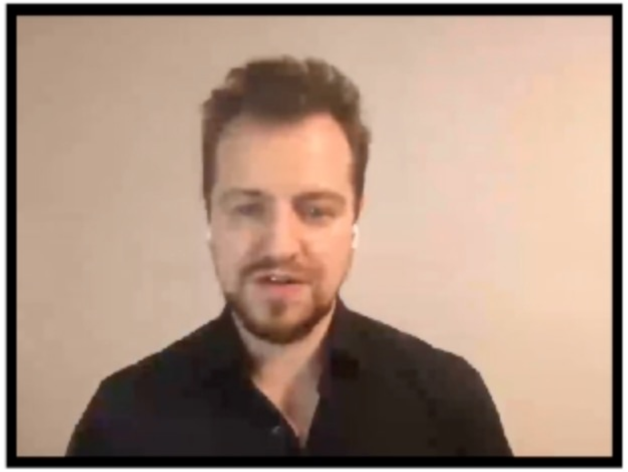




Comparison on Static Graphs

Algorithm	Arabesque [SOSP'15]	Fractal [SIGMOD'19]	Tesseract
4-C	4.9h	310s	147s
4-MC	OOM	12.3h	1.9h
4-FSM-2K	OOM	23.7h	10.3h

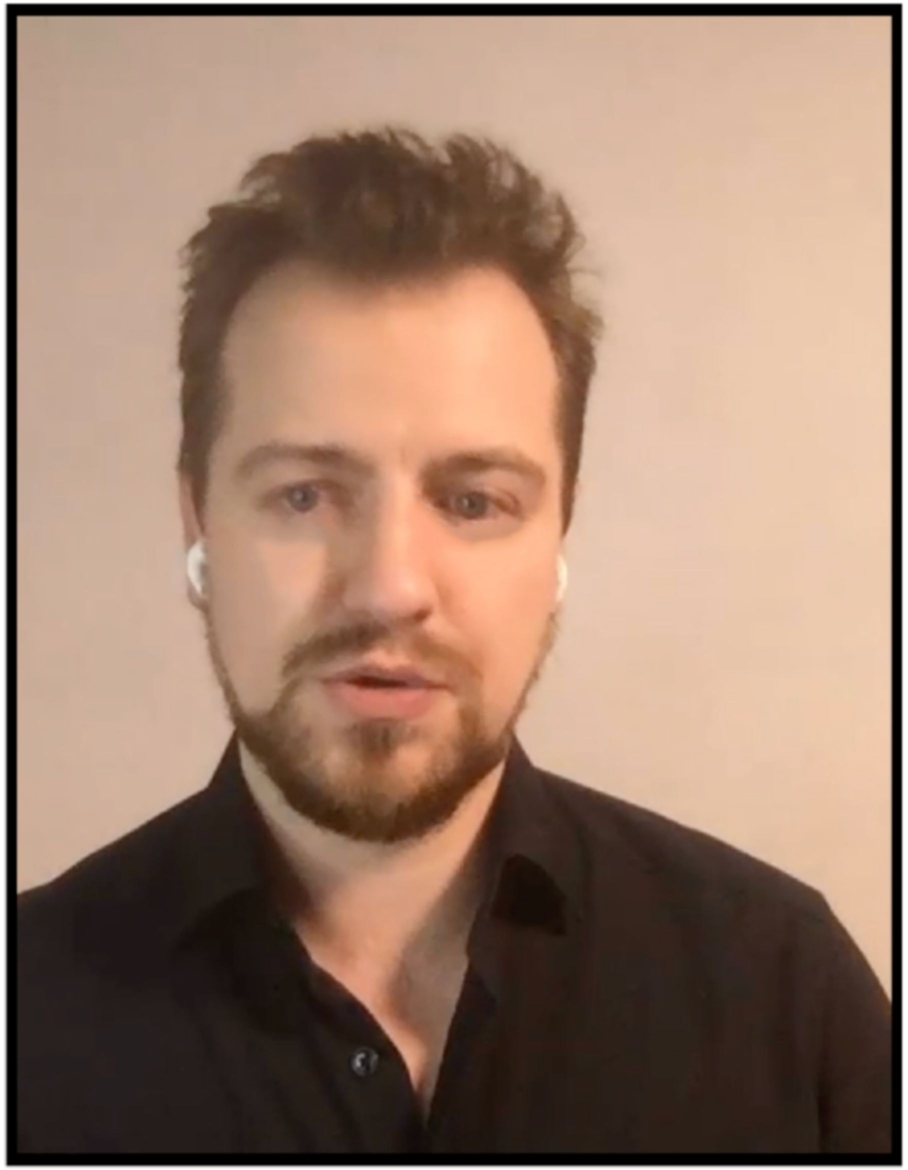
LiveJournal Graph with 8 machines



Incrementally Mining Large Graphs

Metric	UK-2007	DC-2012
Processing Time (1M updates)	372s	1.5h
Output Rate	11.4M/s	7.57M/S

5-GKS-3 algorithm with 8 machines



Additional Results in Paper

- Scalability
- Performance on static graphs
- More large graphs
- Performance breakdown
- Overhead analysis
- Ingest rate performance
- Latency
- Deletions

Conclusions

binds.ch/tesseract

bindscha@mit.edu

Tesseract supports mining evolving graphs

- General-purpose API
- Distributed execution
- Millions of updates per second

Key ideas:

- Update-based task-parallel exploration
- Duplicate elimination

Enables a new class of “interactive” graph mining applications