Tesseract: Distributed, General Graph Pattern Mining on Evolving Graphs

Laurent Bindschaedler
Jasmina Malicevic
Baptiste Lepers

Ashvin Goel
Willy Zwaenepoel
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
Example - Graph Keyword Search

Input Graph

Pattern

Labels

Matches

Video of Presenter
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

**BEFORE**

1. 4
   3
   5
   2
   6
   8

**AFTER**

1. 4
   3
   5
   2
   6
   8

**Matches**

1. 4
   3
   2
   6
   8

**Pattern**

\{○, ○, ○\}
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.
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
**Exploration May Find Duplicates**

Duplicates can occur from a single update or multiple updates.

1. The sequence (1, 2, 3, 4) or (1, 2, 4, 3) can be explored as (1, 2, 3, 4) or (1, 2, 4, 3), which results in a duplicate match from a single update.

2. The sequence can also be found from (1, 2) and (2, 3), which results in a 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

Update Stream → Work Queue → Graph Store (shared)

Distributed Workers

Contains timestamped updates

Multiversioned by timestamp

Matches
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

4-C
- 6000X - 12X

4-FSM-2K
- 1200X - 5X

Runtime (s)

0.1% 1% 10%
Increment

Runtime (h)

0.1% 1% 10%
Increment

Fractal

Tesseract

Video of Presenter

Tesseract – EuroSys’21 – 2021-04-28 – Laurent Bindschaedler, MIT CSAIL
Comparison with Delta-BigJoin [VLDB’18]

LiveJournal Graph – 8 machines

- 4-C: 1.1X
- 4-CL: 6.5X
- 4-MC: 7.3X

- 5-GKS-3: 12.3X

Runtime (s) vs. Runtime (h)

- Delta-BigJoin
- Tesseract

Video of Presenter
## Comparison on Static Graphs

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4-C</td>
<td>4.9h</td>
<td>310s</td>
<td>147s</td>
</tr>
<tr>
<td>4-MC</td>
<td>OOM</td>
<td>12.3h</td>
<td>1.9h</td>
</tr>
<tr>
<td>4-FSM-2K</td>
<td>OOM</td>
<td>23.7h</td>
<td>10.3h</td>
</tr>
</tbody>
</table>

LiveJournal Graph with 8 machines
## Incrementally Mining Large Graphs

<table>
<thead>
<tr>
<th>Metric</th>
<th>UK-2007</th>
<th>DC-2012</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processing Time (1M updates)</td>
<td>372s</td>
<td>1.5h</td>
</tr>
<tr>
<td>Output Rate</td>
<td>11.4M/s</td>
<td>7.57M/S</td>
</tr>
</tbody>
</table>

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

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