A Distributed Vertex-Centric Approach for Pattern Matching in Massive Graphs

- Arash Fard
- M. Usman Nisar
- Lakshmish Ramaswamy
- John A. Miller
- Matthew Saltz

Computer Science Department
University of Georgia, Athens, GA
Outline

- Introduction
- Subgraph Pattern Matching
  - Types of Subgraph Pattern Matching
- Strict Simulation
- Distributed Algorithms
- Experiments
Introduction

- Labeled Directed Graph $G(V, E, l)$
- Versatile and expressive
  - Social networks, web search engines, genome sequencing, etc.
- Data sizes are growing rapidly
  - Facebook: 1 billion+ users, average degree of 140
  - Twitter: 200 million+ users, 400 million tweets each day
- Bigger datasets mean bigger graphs
- **Pattern (Query Graph):** the interesting small graph
- **Data Graph:** the massive graph contains the information
- **Pattern Matching:** finding the instances of the pattern in the data graph
There are different paradigms for pattern matching:

- Exact Topological Matching
  - Sub-graph Isomorphism

- Simulation Matching (meaning of the relations)
  - Graph Simulation
  - Dual Simulation
  - Strong Simulation
  - Strict Simulation

SD: Software Developer
Bio: Biologist
PM: Product Manager
Example: Graph Simulation

- Label and Children conditions
- Quadratic time complexity

Pattern Graph

PM: Product Manager
SA: Software Analyst
DB: Database Designer
AI: AI specialist
SD: Software Developer

Data Graph
Example: Dual Simulation

- Adds Parents condition
- Cubic time complexity

**Pattern Graph**

**Data Graph**

PM: Product Manager
SA: Software Analyst
DB: Database Designer
AI: AI specialist
SD: Software Developer
Strong Simulation

- Extends dual simulation with a locality condition
- A ball $G_b[v, r]$ is a subgraph of a graph $G$ containing:
  - all vertices $V_b$ within an undirected distance $r$ of the vertex $v$
  - all the edges between the vertices in $V_b$

Example: Strong Simulation

- Adds Locality condition
- Cubic time complexity

Pattern Graph

PM: Product Manager
SA: Software Analyst
DB: Database Designer
AI: AI specialist
SD: Software Developer

Data Graph
Example: Subgraph Isomorphism

- Exact Topological Match
- NP-complete in general case

Pattern Graph

Data Graph

PM: Product Manager
SA: Software Analyst
DB: Database Designer
AI: AI specialist
SD: Software Developer
Strict Simulation is a smart modification of Strong Simulation

- Smaller balls
- Faster computation
- The same quality of results or better
- Cubic time complexity

Better Efficiency

Graph Simulation  Dual Simulation  Strong Simulation  Subgraph Isomorphism

Better Match

(a) Comparison of $Q \preceq_{sim} G$, $Q \preceq_{sim}^{D} G$, $Q \preceq_{sim}^{S} G$, and $Q \preceq_{iso} G$

(b) Comparison of $Q \preceq_{sim}^{S} G$ and $Q \preceq_{sim}^{\Sigma} G$
Strict vs Strong Simulation

(a) Strong Simulation

1. Q and G as inputs
2. Apply dual simulation on G to find DualSim match set
3. Create a ball centered at each matching node via DualSim in G
4. Apply dual simulation on each ball
5. Construct a match graph for each ball
6. For each ball, extract the connected component containing the center
7. Set of MaxPGs as output

(b) Strict Simulation

1. Q and G as inputs
2. Apply dual simulation on G to find DualSim match set
3. Construct match graph for DualSim
4. Create a ball centered at each node in the match graph
5. Apply dual simulation on each ball
6. Construct a match graph for each ball
7. Extract the connected component containing the center from each ball
8. Set of MaxPGs as output
Example: Strict vs Strong Simulation

- $d_q = 2$
- Strong Simulation: $G_b[2,2]$ contains all vertices
- Strict Simulation: $G_b[2,2]$ contains $\{2,1,10,3,4\}$
- Blue and white vertices in the result of Strong Simulation
- White vertices in the result of Strict Simulation
Vertex Centric BSP

- Bulk Synchronous Parallel
- Vertex Centric
  - Each vertex of the graph is a computing unit
  - Each vertex initially knows only its id, label, and outgoing edges
  - Algorithm terminates when every vertex votes to halt.

- GPS, a free implementation of Pregel, developed in infoLab, Stanford University.
Distributed Algorithm for Graph Simulation

- **match flag**: indicating if the vertex is a candidate member of the result match set
- **matchSet**: set of vertices in $Q$ that are candidate match

**Initialization:**
- Pattern is received from the Master
- Set the **match** flag to true

<table>
<thead>
<tr>
<th>Vertex</th>
<th>match flag</th>
<th>matchSet</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>True</td>
<td>Empty</td>
</tr>
<tr>
<td>G2</td>
<td>True</td>
<td>Empty</td>
</tr>
<tr>
<td>G3</td>
<td>True</td>
<td>Empty</td>
</tr>
<tr>
<td>G4</td>
<td>True</td>
<td>Empty</td>
</tr>
<tr>
<td>G5</td>
<td>True</td>
<td>Empty</td>
</tr>
<tr>
<td>G6</td>
<td>True</td>
<td>Empty</td>
</tr>
<tr>
<td>G7</td>
<td>True</td>
<td>Empty</td>
</tr>
<tr>
<td>G8</td>
<td>True</td>
<td>Empty</td>
</tr>
</tbody>
</table>
Distributed Algorithm for Graph Simulation

Data Graph

Pattern Graph

<table>
<thead>
<tr>
<th>Vertex</th>
<th>match flag</th>
<th>matchSet</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>False</td>
<td>Empty</td>
</tr>
<tr>
<td>G2</td>
<td>True</td>
<td>Q2</td>
</tr>
<tr>
<td>G3</td>
<td>True</td>
<td>Q1</td>
</tr>
<tr>
<td>G4</td>
<td>True</td>
<td>Q3</td>
</tr>
<tr>
<td>G5</td>
<td>True</td>
<td>Q2</td>
</tr>
<tr>
<td>G6</td>
<td>True</td>
<td>Q3</td>
</tr>
<tr>
<td>G7</td>
<td>False</td>
<td>Empty</td>
</tr>
<tr>
<td>G8</td>
<td>False</td>
<td>Empty</td>
</tr>
</tbody>
</table>
Distributed Algorithm for Graph Simulation

Data Graph

2nd Superstep

Pattern Graph

<table>
<thead>
<tr>
<th>Vertex</th>
<th>match flag</th>
<th>matchSet</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>False</td>
<td>Empty</td>
</tr>
<tr>
<td>G2</td>
<td>True</td>
<td>Q2</td>
</tr>
<tr>
<td>G3</td>
<td>True</td>
<td>Q1</td>
</tr>
<tr>
<td>G4</td>
<td>True</td>
<td>Q3</td>
</tr>
<tr>
<td>G5</td>
<td>True</td>
<td>Q2</td>
</tr>
<tr>
<td>G6</td>
<td>True</td>
<td>Q3</td>
</tr>
<tr>
<td>G7</td>
<td>False</td>
<td>Empty</td>
</tr>
<tr>
<td>G8</td>
<td>False</td>
<td>Empty</td>
</tr>
</tbody>
</table>
**Distributed Algorithm for Graph Simulation**

**Pattern Graph**

**Data Graph**

<table>
<thead>
<tr>
<th>Vertex</th>
<th>match flag</th>
<th>matchSet</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>False</td>
<td>Empty</td>
</tr>
<tr>
<td>G2</td>
<td>True</td>
<td>Q2</td>
</tr>
<tr>
<td>G3</td>
<td>True</td>
<td>Q1</td>
</tr>
<tr>
<td>G4</td>
<td>True</td>
<td>Q3</td>
</tr>
<tr>
<td>G5</td>
<td>False</td>
<td>Empty</td>
</tr>
<tr>
<td>G6</td>
<td>True</td>
<td>Q3</td>
</tr>
<tr>
<td>G7</td>
<td>False</td>
<td>Empty</td>
</tr>
<tr>
<td>G8</td>
<td>False</td>
<td>Empty</td>
</tr>
</tbody>
</table>
Other Distributed Graph Algorithms

- **Dual simulation**
  - In addition to storing the match sets of its children, a vertex also stores the match sets of its parents
  - During match set evaluation, a vertex takes both of these into account

- **Strong simulation**
  - Run dual simulation to obtain match relation $R_d$
  - For each matching vertex $v$ in $R_d$:
    - Create a ball centered at $v$ with radius $d_Q$ containing any vertex in $G$
    - Perform dual simulation on the ball

- **Strict simulation**
  - Run dual simulation to obtain match relation $R_d$
  - For each matching vertex $v$ in $R_d$:
    - Create a ball centered at $v$ with radius $d_Q$ containing only vertices in $R_d$
    - Perform dual simulation on the ball
Experiments

- System
  - Cluster of 12 machines
  - 128 GB RAM
  - Two 2GHz Intel Xeon E-2620
  - GPS (developed in infoLab, Stanford University)

- Datasets
  - Synthesized graphs: $|V| = 100e6$, $|E| = |V|^\alpha$, $\alpha = 1.2$
  - Real world graphs
    - uk-2002: $|V| = 18e6$, $|E| = 298e6$
    - ljournal: $|V| = 5e6$, $|E| = 79e6$
    - Amazon-2008: $|V| = 7e5$, $|E| = 5e6$

http://law.di.unimi.it/datasets.php
Experiment 1: Comparison of Strong and Strict Simulation

Synthesized ($|V| = 1e6, \alpha = 1.2$)

Amazon-2008 ($|V| = 7e5, |V_q| = 20$)

Amazon-2008 ($|V| = 7e5, |V| = 5e6$)

Amazon-2008 ($|V| = 7e5, |V_q| = 20$)
Experiment 2: Running time

Quality of result:
Graph Simulation < Dual Simulation < Strict Simulation

Synthesized ($|V| = 1e8$, $\alpha = 1.2$, $|E| = |V|^\alpha$)

$uk-2002-hc$ ($|V| = 18e6$, $|E| = 298e6$)

($l = 200$ & $|V_q| = 20$)
Experiment 3: Impact of dataset

Synthesized ($\alpha = 1.2$, $|E| = |V|^\alpha$)

Synthesized ($|V| = 100e6$)

($l = 200 \& \ |V_q| = 20$)
Experiment 4: Impact of pattern

Synthesized ($|V| = 100e6$, $\alpha = 1.2$, $|E| = |V| \alpha$)

$uk-2002-hc$ ($|V| = 18e6$, $|E| = 298e6$)
Related work

- Pattern matching models
  - P-homomorphism
  - Graph Simulation, Dual and Strong
  - Bounded simulation

- Distributed algorithms for pattern matching
Conclusion

- Introducing a new Pattern Matching model
- Design, implementation, and test of 4 distributed algorithms for Pattern Matching
  - Graph Simulation
  - Dual Simulation
  - Strong Simulation
  - Strict Simulation
Questions?

Thanks