Optimizing Subgraph Queries With a Mix of Tradition & Modernity

Semih Salihoglu (joint w/ Amine Mhedhbi)

July 5th, 2019
Subgraph Queries = Multiway (Self) Join Queries

MATCH (a)->(b)->(c), (a)->(c)
Traditionally: Binary Join Plans

Table(s)/Q-Edge(s)-at-a-time Joins

R1

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

...  

R2

<table>
<thead>
<tr>
<th>a</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

...  

R3

<table>
<thead>
<tr>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

...  

INT₁

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

...  

Output

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
Recent Developments (AGM ‘08, NPRR ’12): BJ Plans are Not (even) Worst-case Optimal

Any BJ Plan will take $O(m^2)$ time.

But max output is: $m^{3/2}$ (AGM Bound of Triangle Query)
Generic Join: A WCO Algorithm (NPRR, 2013)

Column/Q-Vertex-at-a-time

Order q-vertices: say: a,b,c

INT

\[ \text{a} \cap \text{b} \]

\[ \text{c} \]

\[ \text{INT}_1 \text{a} \]
Generic Join: A WCO Algorithm (NPRR, 2013)

**Column/Q-Vertex-at-a-time**

Order q-vertices: say: a,b,c

![Graph](image)
Generic Join: A WCO Algorithm (NPRR, 2013)

Column/Q-Vertex-at-a-time
Order q-vertices: say: a,b,c
Generic Join: A WCO Algorithm (NPRR, 2013)

Column/Q-Vertex-at-a-time
Order q-vertices: say: a,b,c

\[
Q = \begin{array}{ccc}
  a & b & c \\
  \hline
  1 & 2 & \text{(INT}_1 \text{)} \\
  1 & 3 & \text{ } \\
  1 & 4 & \text{ } \\
  2 & 4 & \text{ } \\
  2 & 5 & \text{ } \\
  2 & 6 & \text{ } \\
  3 & 4 & \text{ } \\
\end{array}
\]

\[
\begin{array}{ccc}
  a & b & c \\
  \hline
  1 & 2 & \text{(INT}_2 \text{)} \\
  1 & 3 & \text{ } \\
  2 & 4 & \text{ } \\
  2 & 5 & \text{ } \\
  2 & 6 & \text{ } \\
  3 & 4 & \text{ } \\
\end{array}
\]

\text{Output: } a \ b \ c
Theorem (GJ is WCO): Runtime of GJ ≤ AGM
(for any query & any q-vertex ordering)
Research Overview on WCO Joins

Serial Setting

Prototype GDBMS at UWaterloo (SIGMOD ‘17 Demo, VLDB ‘19)

Distributed Setting

Timely Dataflow (previously Naiad) (VLDB ‘18)

What should the query vertex orderings (QVOs) be?
For which queries to use?
How to mix with traditional joins?

…
Graphflow Logical and Physical Plans for SQs

- Storage: 2 adjacency lists per vertex (fw & bw) stored in memory

(Hybrid) Logical Plan

Physical Plan

(WCO) Logical Plan

QVO: c,d,b,a
Q1: Ordering QVOs: Good Orders

- Two main effects of QVOs:
  - First Effect: # intermediate partial matches.

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intermediate Matches</td>
<td>3.5M</td>
<td>40M</td>
</tr>
<tr>
<td>Run time</td>
<td>9s</td>
<td>16s</td>
</tr>
</tbody>
</table>

Soc-Epinions
Q1: Ordering QVOs: Good Orders

- **2nd Effect**: Directions of adjacency lists intersected.

<table>
<thead>
<tr>
<th>P1:QVO: a,b,c</th>
<th>P2: QVO: a,c,b</th>
<th>P3: QVO: b,c,a</th>
</tr>
</thead>
</table>

Graphflow Optimizer: Minimize estimated *intersection cost (i-cost)*

\[ I\text{-cost}: \sum \text{(size-of-adj-lists-intersected-throughout-execution)} \]

(estimate # partial matches & avg adj-list sizes w/ subgraph catalogue)
### I-cost Estimation Technique: Subgraph Catalogue

<table>
<thead>
<tr>
<th>$S$</th>
<th>Adj. Lists</th>
<th>$S'$</th>
<th>i-cost</th>
<th>selectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td><img src="image1" alt="Graph" /></td>
<td><img src="image2" alt="Adj. Lists" /></td>
<td><img src="image3" alt="Graph" /></td>
<td>57</td>
<td>4.6</td>
</tr>
<tr>
<td><img src="image4" alt="Graph" /></td>
<td><img src="image5" alt="Adj. Lists" /></td>
<td><img src="image6" alt="Graph" /></td>
<td>158</td>
<td>4.6</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Picking a fixed QVO may be suboptimal.

P1: QVO: b,c,d,a

P2: QVO: b,c,a,d

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2n</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td></td>
<td>n</td>
</tr>
<tr>
<td>0</td>
<td>2n</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>3n</td>
<td>3n</td>
</tr>
</tbody>
</table>
Q1: Ordering QVOs: Example Adaptive WCO Plan

b, c, d, a

b, c, a, d

b, c, a, d
Q1: Ordering QVOs: Effects of Adaptive WCO Plans

Soc-Epinions

Web-Google
Q2: How to mix BJs with Intersections?

- Cost-based DP optimization algorithm.

Optimum Plans for all connected sub-queries with 2 q-vertices:

1. a → b
2. a → c
3. b → c
4. c → d
5. d → b

Optimum Plans for all connected sub-queries with 3 q-vertices:

Optimum Plans for all connected sub-queries with 4 q-vertices:

Each step: consider plans with both hash joins & intersections.
Graphflow vs Empty-Headed Hybrid Plans

<table>
<thead>
<tr>
<th></th>
<th>Amzn</th>
<th>Web-G</th>
</tr>
</thead>
<tbody>
<tr>
<td>GF</td>
<td>6.7s</td>
<td>328s</td>
</tr>
<tr>
<td>EH</td>
<td>25.2s</td>
<td>&gt;5 hours</td>
</tr>
</tbody>
</table>
Q3: For Which Queries are WCO, BJ, Hybrid Plans Good?

- Size and cyclicity

![Diagram of connections between nodes a, b, c, d, e, f]

- Graphs showing run times for WCO, H, and BJ plans on Soc-Epinions and Amazon datasets.
Q3: For Which Queries are WCO, BJ, Hybrid Plans Good?
Summary

• For subgraph queries, Graphflow optimizer:
  • Adopts the *i-cost cost metric*
  • Uses a *subgraph catalogue* for i-cost estimation
  • Seamlessly mixes binary joins and WCO-style intersections with a dynamic programming optimizer
  • Adaptively changes the query vertex orderings during runtime
• Similar techniques for cont. subgraph matching (upcoming work)
Students

Amine Mhedhbi
Siddhartha Sahu
Chathura Kankanamge
References

• Efficiently Updating Materialized Views, BLT, SIGMOD Record, 1986
• An Efficient Distributed Subgraph Mining Algorithm in Extreme Large Graphs, WB, AICI, 2010
• Optimizing Multiway Joins in a Map-Reduce Environment, AU, TKDE, 2011
• Leapfrog Triejoin: a worst-case optimal join algorithm, Veldheuizen, arxiv 2012
• Tri, Tri Again: Finding Triangles and Small Subgraphs in a Distributed Setting, DLP, DISC, 2012
• Worst-case Optimal Join Algorithms, NPRR, PODS 2012
• A Distributed Graph Engine for Web Scale RDF Data, ZYWKW, VLDB, 2013
• Communication Steps for Parallel Query Processing, BKS, PODS, 2013
• Incremental Maintenance for LeapfrogTriejoin, Veldheuizen, arxiv 2013
• Naiad: A Timely Dataflow System, MMIIA, SOSP, 2013
• PATRIC: A Parallel Algorithm for Counting Triangles in Massive Network, CIKM, 2013
• Upper and Lower Bounds in the Cost of a Map-Reduce Computations, ASSU, VLDB, 2013
• Anchor Points Algorithms for Hamming and Edit Distance, ADRRSU, ICDT 2014
• Arabesque: A System for Distributed Graph Mining, SOSP, 2015
• Continuous Pattern Detection Over Billion-edge Graph Using Distributed Framework, CHCAF, EDBT, 2015
• EmptyHeaded: A Relational Engine for Graph Processing, ATOR, SIGMOD, 2016
• PTE: enumerating trillion triangles on distributed systems, PMK, KDD, 2016
• Scalable Distributed Subgraph Enumeration, LQLZC, VLDB, 2016
• Graphflow: An Active Graph Database, KSMCS, SIGMOD, 2017
• GYM: A Multiround Distributed Join Algorithm, AJRSU, 2017
• Distributed Evaluation of Subgraph Queries Using Worst-case Optimal Low-Memory Dataflows, AMS, VLDB, 2018
• Ubiquity of Large Graphs and Surprising Challenges of Graph Processing, SMSLO, VLDB, 2018
• Algorithmic Aspects of Parallel Data Processing, KSS, F&T, 2018
Thank you