Scalable SPARQL Querying using Path Partitioning

Buwen Wu 1, Yongluan Zhou 2, Pingpeng Yuan 1, Ling Liu 3, Hai Jin 1

1 SCTS/CGCL, Huazhong University of Science and Technology, Wuhan, China
2 University of Southern Denmark, Odense, Denmark
3 Georgia Institute of Technology, Atlanta, USA

1. BACKGROUND

- Resource Description Framework (RDF)
- A simple yet powerful data model
- A set of triple: (Subject, Predicate, Object)
- A labeled and directed graph
- SPARQL
- A set of triple patterns
- Joins between triple patterns
- S-S join, S-O join and O-O join...
- Distributed joins
- Inner-partition join
- Cross-partition join
- Communication cost

2. MOTIVATION

- Existing data partitioning algorithms
- Hash partitioning (hasing on S or/and O)
- N-hop extension
- Problems
  - The necessary number of subqueries is large for complex queries
  - Data distribution is skewed
  - Data duplication is high

3. OVERVIEW

- End-to-End path
  - A path from start vertex to end vertex
  - Chain queries are inner-partition queries
- Merged vertex
  - If all paths contains v are in the same partition, v is called merged
- Query Decomposition
  - Combine two inner-partition subqueries into a new inner-partition query
  - One of the shared vertices between them is merged
  - Share a constant vertex
  - Share a variable vertex

4. PROBLEM FORMULATION

- Metrics
  - Balance, denoted as |L(P_i)|
  - Data duplication, < |W'_v|
  - Query efficiency, < |W'_v|
- (k,1)-balanced path partitioning problem
- Objective function:
  - Maximize |W'_v| s.t. |L(P_i)| ≤ (k-1) / k
- NP-Hard and APX-Hard

5. APPROXIMATE ALGORITHM

- Relaxed version of the problem
  - (k,2)-balanced path partitioning problem
  - L(P_i) ≤ (k-1)/k
- Approximation ratio
  - The number of merged vertices is at least
    1 - (k-1)/k |W'_v|
- Complexity
  - Time complexity is Θ(|E|)
  - Space complexity is Θ(|E|)

6. HEURISTIC ALGORITHM

- Aim at reducing space complexity
- Merging start vertices
- Avoid generating all paths
- Star, chain, cycle and tree queries are inner-partition queries
- Reduce space complexity
- Vertex weighting
- An estimation without generating all paths
- The weight of merging v is denoted by the number of paths that contain v:
  w(v) = L_P(v) = Σ_P(v) = Σ_P(v)
- The iteratively computation equation:
  I_P = (1 - α) + λ + α I_P
- Class-based vertex weighting
  - Combine two subqueries
  - The rdf:type information of the shared variable is known
  - Merge vertices by their class
  - The weight of a class is the average weight score of all the vertices in it
- Complexity
  - Time complexity is Θ(|E|)
  - Space complexity is Θ(|E|)

7. EXPERIMENT

- Cluster Setup: 20 nodes with 2.4GHz CPU, 6GB RAM and 500GB disk
- Datasets: LUBM, UniPort, SP2Bench, ranging from 6k triples to 2 billion
- Query: Star, chain, tree, cycle, complex...
- Competitors: Path-AX, Path-BM, Path-BMC, Path-Hash, un-one, un-two, 2f

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Data Size</th>
<th>Max Partition Size</th>
<th>Data Duplication</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUBM-2000</td>
<td>0.0253</td>
<td>11.8k</td>
<td>0.22</td>
</tr>
<tr>
<td>int-2h</td>
<td>0.0458</td>
<td>21.7k</td>
<td>2.06</td>
</tr>
<tr>
<td>2f</td>
<td>0.0456</td>
<td>4.8k</td>
<td>0.12</td>
</tr>
<tr>
<td>Path-BM</td>
<td>0.0087</td>
<td>5.9k</td>
<td>1.16</td>
</tr>
<tr>
<td>Path-Hash</td>
<td>0.0015</td>
<td>8.6k</td>
<td>0.83</td>
</tr>
<tr>
<td>2f</td>
<td>0.0072</td>
<td>8.5%</td>
<td>0.15</td>
</tr>
<tr>
<td>UniPort</td>
<td>0.0058</td>
<td>8.1%</td>
<td>2.26</td>
</tr>
<tr>
<td>Path-BMC</td>
<td>0.0021</td>
<td>5.3%</td>
<td>1.25</td>
</tr>
<tr>
<td>Path-Hash</td>
<td>0.0032</td>
<td>6.0%</td>
<td>0.20</td>
</tr>
<tr>
<td>Path-BM</td>
<td>0.0002</td>
<td>6.9%</td>
<td>1.76</td>
</tr>
<tr>
<td>LUBM-500M</td>
<td>0.0028</td>
<td>5.1%</td>
<td>0.12</td>
</tr>
</tbody>
</table>

- Query Performance
- Scalability - varying the size of datasets
- Scalability - varying the size of cluster (Path-BMC)