BENCHMARKING ESSENTIAL GRAPH QUERIES

Renzo Angles

Assistant Professor, Universidad de Talca (Chile)
Post-doc, Vrije University Amsterdam

1st Workshop on Graph-based Technologies and Applications (Graph-TA)
Barcelona, February 19, 2013
Motivation

- Increasing amount of graph data
- NOSQL databases => Graph (oriented) databases
- What is the most suitable graph database?
  - Theoretical comparison (complexity and expressive power)
  - Empirical comparison (performance, usability, etc.)
  - Benchmarks (there is not a standard one)
  - The application domain is very important
- Our work
  - Empirical comparison of loading and querying data at low-scale (1K, 100K, 500K, 1M)
  - Experience on using several graph databases
Benchmark use case

- 12 essential graph queries
  - Attribute searching (Get people with a given name)
  - Node/edge adjacency (Get people that likes a given Web page)
  - Fixed-length paths (Get the friends of the friends of a given person)
  - Reachability (Is there a “friend” connection between two people?)
  - Pattern matching (Get the common friends between two people)
  - Aggregates (Get the number of friends of a given person)
Data loading test

For different data sets (N values) and each database (4Store, Allegro, Bigdata, Dex, Neo4j, OrientDB, Postgres, Virtuoso), the time (in minutes) required for data loading is recorded.

- **N=1,000**: Time = 5.874 minutes
- **N=100,000**: Time = 1,002.216 minutes
- **N=500,000**: Time = 5,735.332 minutes
- **N=1,000,000**: Time = 12,094.498 minutes
# Expressing graph queries

<table>
<thead>
<tr>
<th></th>
<th>Get the friends of a person identified by id 10</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dex</strong></td>
<td>long person_id = dex_graph.findObject(pid, dexvalue.setLong(10)): dex_graph.neighbors(person_id, friend, EdgesDirection.Outgoing);</td>
</tr>
<tr>
<td><strong>InfiniteGraph</strong></td>
<td>Person person = this.findPersonById(10); Iterator&lt;VertexHandle&gt; it = person.getNeighbors().iterator(); while (it.hasNext()) { … }</td>
</tr>
<tr>
<td><strong>Neo4j</strong></td>
<td>START p=node:peopleIdIndex(id=10) MATCH p-[:friend]-&gt;f RETURN f</td>
</tr>
<tr>
<td><strong>OrientDB</strong></td>
<td>SELECT FROM ographvertex WHERE in[label='friend'].out in (select rid from index:personIdx where key = 10)</td>
</tr>
</tbody>
</table>
Data querying test (12 queries x 100 instances)

- N=1,000  E=5.874
- N=100,000  E=1,002.216
- N=500,000  E=5,735.332
- N=1,000,000  E=12,094.498
Conclusions

• We developed a benchmark for essential graph queries
• We present
  • our experience on using current graph databases
  • preliminary results of performance for loading and querying data
• The comparison of current graph databases is not an easy task
  • There are several approaches
  • Standards are required (e.g., a graph query language)
• Opportunity:
  • Research on foundations on graph data management
  • Development of benchmarks for graph databases
  • Use and testing of graph databases in real-life applications