The LDBC Social Network Benchmark: Business Intelligence workload

Gábor Szárnyas
CWI

15th TUC meeting
Data pipeline

SNB Interactive

transactional system

database dump

aggregator

snapshot

analytical system #1

analytical system #2

transaction log

aggregator

updates

SNB BI

Graphalytics: BFS, PageRank, …
LDBC SNB BI workload

A modern OLAP benchmark suite

- Correlated, temporal graph data set
- Analytical queries, including graph operations
- Inserts & deep deletes
- Parameter curation
Social network data set

- Correlated
- Temporal
Example graph

Main entities:
- Person-knows-Person network
- Forums
- Message threads

Correlations:
- Structure-level
- Attribute-level

Dynamic graph
Lifespans

The generator generates the entire temporal with creation dates * and deletion dates †.
Dynamic graph

Initial snapshot (97%) and insert/delete batches

Supporting dynamic graphs in SNB Datagen, GRADES-NDA 2020
Deleting a Forum

Deletes are heavy-hitting operations
Workload

- Workload
- Parameter curation
- Example queries
Choke points

A choke point is a difficult aspect of query processing that has a significant impact on the performance of the query.

Examples:

- Join ordering
- Data access locality
- WCOJs
- Path queries

TPCTC’12: experiences of implementing TPC-H on Vectorwise, Virtuoso, and HyPer
Workload

**Workload:** Ad-hoc graph OLAP queries with daily updates

**Batches:** 33 days of W/R operations

- **W:** apply one day’s worth of updates
- **R:** 20 complex read queries with different parameters

≈35 months  
2012-Nov-29  
2012-Nov-30  
2012-Dec-31

- **power test**  
- **throughput test**  
- **throughput test**
Parameter curation

Parameter selection is particularly important for skewed and correlated data sets:

- equality constraint: $id = \$personId$
- range constraint: $creationDate < \$maxDate$

- starting a query from a person with a low degree vs. a high degree
- cost of reachability queries if there is a path vs. no path
Umbra SF10: naïve vs. curated parameters
Q11: Triangle query – WCOJs are beneficial

Parameters: Only big countries, similar intervals
Q14: Correlations – Different runtimes/query plans

For each pair of countries, calculate the cost as a sum of cases 1-4. Cases that have a match add to the final score with the specified value. Each case only counts once, multiple matches do not increase to the score.

Case 1: score += 4
- person1: Person
- person2: Person
- Comment
- Message

Case 2: score += 1
- person1: Person
- person2: Person
- Comment
- Message

Case 3: score += 10
- person1: Person
- person2: Person
- Message

Case 4: score += 1
- person1: Person
- person2: Person
- Message

Parameters: (A) close countries (B) far-away countries
Q19: Multi-source weighted shortest path

**Find the shortest paths between all pairs of Persons in city1 and city2**

- **city1:** City
  - id = $city1id
  - locatedIn
  - person1: Person

- **city2:** City
  - id = $city2id
  - locatedIn
  - person2: Person

**Parameters:**

(A) Cities from the same country
(B) Cities from different countries

**Case i1: Reply from personA to Person B's Message**

- personA: Person
  - knows
  - personB: Person
  - hasCreator
  - c: Comment
    - replyOf
    - m: Message

**Case i2: Reply from personB to personA's Message**

- personA: Person
  - knows
  - personB: Person
  - hasCreator

The weight of a knows edge is based on the number of interactions between its Persons:

\[
\text{knows.weight} = \frac{1}{\text{count}(i1) + \text{count}(i2)}
\]
Q20: Single-source weighted shortest path

Parameters:
(A) There is a path between $company employees and $person2
(B) There is no path between $company employees and $person2
# BI Implementations

<table>
<thead>
<tr>
<th>System</th>
<th>Data Model</th>
<th>Language</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>neo4j</td>
<td>graph</td>
<td>Cypher</td>
<td>495</td>
</tr>
<tr>
<td>UMBRA</td>
<td>relational</td>
<td>SQL</td>
<td>755</td>
</tr>
<tr>
<td>TigerGraph</td>
<td>graph</td>
<td>GSQL</td>
<td>832</td>
</tr>
</tbody>
</table>
Execution and scoring
Workload execution

- Power test: sequential query execution

- Throughput tests: concurrent query execution
  - Concurrent RW
  - Disjoint RW
Scoring metrics: Power

Geometric mean ensures all queries are of equal importance

\[
power@SF = \frac{3,600}{\sqrt[29]{w \cdot q_1 \cdot \ldots \cdot q_{20a} \cdot q_{20b}}} \cdot SF
\]
Scoring metrics: Throughput

Run throughput batches for at least 1 hour and extrapolate to one day.

\[
\text{throughput@SF} = (24 \text{ hours} - t_{load}) \cdot \frac{n_{batches}}{t_{batches}} \cdot SF
\]
Scoring metrics: Price-performance

Power and throughput metrics, taking the total cost of ownership into account, using TPC’s pricing.

\[
power@SF/\$ = power@SF \cdot \frac{1,000}{TCO}
\]

\[
throughput@SF/\$ = throughput@SF \cdot \frac{1,000}{TCO}
\]
Scalability

BI workload scales up to SF10k: 10,000 GiB CSV data sets.

- Larger than SF10k results are rare even for TPC-H (~14% in the last decade)

Economics of SF10k generation:

- Data generation: $64
- Parameter generator: <20 minutes on a single machine
Summary
Conclusion

State-of-the-art OLAP benchmark

- Scales to SF10k (10,000 GiB) graphs
- Paper with specification and experiments submitted

Plans:

- Start audits
- Generate SF30k+ data sets
- Backport improvements to SNB Interactive
Query design

Choke points and parameters

- Choke point analysis
- Query templates
- Parameter curation
Scalability

Spark-based data generator for increasing scale factors 1, 3, 10, …

The benchmark needs to be economical.

Generating the SF10k data set:

- AWS Elastic MapReduce
- 100 instances with 128GiB RAM
- 1.5 hour runtime

Cost: $74
Comparison of workloads

<table>
<thead>
<tr>
<th></th>
<th>Interactive v1.0</th>
<th>Business Intelligence v1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>focus</strong></td>
<td>OLTP</td>
<td>OLAP</td>
</tr>
<tr>
<td><strong>typical query</strong></td>
<td>2-3 hop neighbourhood queries with filtering</td>
<td>multi-hop/path/subgraph queries with filtering &amp; aggregation</td>
</tr>
<tr>
<td><strong>refresh operations</strong></td>
<td>inserts</td>
<td>inserts and deletes</td>
</tr>
<tr>
<td><strong>target metric</strong></td>
<td>total compression ratio, implying throughput (ops/s)</td>
<td>throughput (ops/day)</td>
</tr>
</tbody>
</table>
Graph data management systems

GDMSs provide a graph-aware UI and support graph processing features.

- Property graph data model
- Graph query language
- Graph visualization
- Relational queries
- Subgraph matching
- Path queries
- Graph algorithms
Subgraph matching

The complexity of a triangle query with binary joins is provably suboptimal: $O(|E|^2)$

Triggered by many-to-many edges and skewed distributions.

Worst-case optimal multi-way join algorithms are needed, which have a complexity of just $O(|E|^{1.5})$ for this query.
Path queries

Implementing an efficient BFS/shortest path algorithm is non-trivial:

- direction optimizing BFS (push-pull) SC 2012
- landmark labelling for distance queries SIGMOD 2013
- multi-source batched BFS VLDB 2014

GDMSSs rarely support these optimizations.
## BI Implementations

<table>
<thead>
<tr>
<th>System</th>
<th>Data Model</th>
<th>Language</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neo4j</td>
<td>graph</td>
<td>Cypher</td>
<td>495</td>
</tr>
<tr>
<td>TigerGraph</td>
<td>graph</td>
<td>GSQL</td>
<td>832</td>
</tr>
<tr>
<td>Umbra</td>
<td>relational</td>
<td>SQL</td>
<td>755</td>
</tr>
</tbody>
</table>