On Data Placement Strategies in Distributed RDF Stores

Int. Workshop on Semantic Big Data (SBD 2017)

Daniel Janke, Steffen Staab, Matthias Thimm
19.05.2017
Distributed RDF Stores

- Requirement for trillion triples stores arose in the last years
- Scalable RDF stores in the cloud

Challenges:
- Data placement strategies
- Distributed query processing
- Handling failures of compute nodes
Distributed RDF Stores

- Requirement for trillion triples stores arose in the last years
- Scalable RDF stores in the cloud

Challenges:
- **Data placement strategies**  
  Focus of our research
- Distributed query processing
- Handling failures of compute nodes
Data Placement Strategies and Scalability

SELECT ?org ?name WHERE

{?org ex:employs ?pers . ?pers foaf:givenname ?name}

Horizontal containment

• Computation of individual query results on local data
• Indicator for robust query processing when scaling horizontally

Vertical parallelization

• Parallel computation of different query results on different compute nodes
• Indicator for query processing scaling with growing result set sizes when scaling horizontally
Data Placement Strategies and Scalability

SELECT ?org ?name WHERE
{?org ex:employs ?pers . ?pers foaf:givenname ?name}

Compute Node 1

gesis:Dog

gesis:wanja

foaf:knows

“Martin”

Commonly held belief:
Horizontal containment dominates query processing effort (cf. [Huang2011SSQ, Lee2013EDP, Zhang2013ETS, ...])

Horizontal containment
• Computation of individual query results on local data
• Indicator for robust query processing when scaling horizontally

Vertical parallelization
• Parallel computation of different query results on different compute nodes
• Indicator for query processing scaling with growing result set sizes when scaling horizontally
Outline

1) Data Placement Strategies
2) Benchmark methodology showing the interdependencies of data placement strategies and query processing
3) Analysis indicating that vertical parallelization may dominate horizontal containment
4) Conclusion
Graph Cover

Assignment of each triple to at least one compute node

Graph chunk
Set of triples assigned to a single compute node
Common Graph Cover Strategies

Hash cover [e.g. Harth2007YAF]

- Triple placement bases on subject hash modulo number of compute nodes

Hierarchical cover [Lee2013SQO]

- Triple placement bases on hash of subject IRI prefixes

Minimal edge-cut cover [Karypis1998AFA]

- Assign vertices (subjects and objects) to partitions such that
  - Number of edges between vertices of different partitions is minimized and
  - Each partition contains approximately $\frac{|V_G|}{|C|}$ vertices
Common Evaluation Strategies

1) Evaluations of graph cover strategies using different databases
=> other components might bias evaluation results
  e.g. [Wu2014SAS, Zeng2013ADG]

Car 1 using fuel A

Car 2 using fuel B

Does fuel A or B allow for a higher speed?

Images from https://openclipart.org
Common Evaluation Strategies

1) Evaluations of graph cover strategies using different databases
   => other components might bias evaluation results
   e.g. [Wu2014SAS, Zeng2013ADG]

2) Usage of slow communication means like Hadoop File System
   => Increased importance of horizontal containment
   e.g. [Huang2011SSQ, Lee2013EDP]

Images from https://openclipart.org
Benchmark Methodology

**Goal**: Investigating effect of graph cover on the scalability
Strategy for Generating Queries

Query Generator: **SPLODGE** [Görlitz2012SSG]

- Generates SPARQL queries for arbitrary datasets
- Generates queries based on
  - Number of joins
  - Join pattern
  - Selectivity
  - Number of data sources
Query Execution Strategy

- Query optimizers fitting for arbitrary graph covers difficult
- Execution of several query execution trees:

Bushy

```
1 2 3 4
```

Left-linear

```
1 2
3 4
```

Right-linear

```
1
2
3 4
```
Koral

- Graph cover independent distributed RDF store
- Inspired by TriAD [GurajadaTheobald2014TAD]
Evaluation Measures

Overall performance
• Query execution time

Horizontal Containment
• Data transfer $T$:
  variable bindings transferred between compute nodes

Vertical Parallelization (VP)
• Workload Entropy $W$:
  entropy of join comparisons on each compute node

<table>
<thead>
<tr>
<th>$T$ low</th>
<th>$T$ high</th>
</tr>
</thead>
<tbody>
<tr>
<td>low VP</td>
<td>low VP</td>
</tr>
<tr>
<td>high VP</td>
<td>low-medium VP</td>
</tr>
</tbody>
</table>

Benchmark

Dataset | Queries | Query execution strategy |
--------|---------|--------------------------|
Distributed RDF store for arbitrary graph covers | Evaluation measures |
Benchmark |
Experimental Setup

Compared graph cover strategies:
- Hash, hierarchical hash and minimal edge-cut cover

Dataset:
- 1 trillion triple subset of BTC2014 [Käfer2014BTC]

Queries:
- Number of joins: 2 and 8 triple patterns
- Join pattern: path-shaped and star-shaped
- Selectivity: 0.001% and 0.01% (1M and 10M triples)
- Number of data sources: 1 and 3

Computer environment:
- 1 Master à 4 cores, 8 GB RAM, 1 TB HDD
- 20 Slaves à 1 core, 2 GB RAM, 300 GB HDD
- 1 Gbit ethernet
Graph Cover Creation Time

- Minimal edge-cut cover requires most time for creation
- Hash cover is created the fastest
Overall Query Performance

- Bushy query execution outperforms other execution strategies
- Minimal edge-cut causes slowest query execution in most cases
- None of the hash-based covers is faster in general
Horizontal Containment

- Star-shaped queries produce no data transfer
- Minimal edge-cut covers produces less data transfer
- Hash-based covers similar data transfer
Vertical Parallelization

- Minimal edge-cut cover has the least balanced workload
- Hash-based covers have similar balanced workloads
Conclusion

- Minimal edge-cut cover
  - Longest cover creation time
  - Lowest data transfer => high horizontal containment
  - Lowest workload balance => low vertical parallelization
  - Overall performance worse than hash-based covers
- Hash-based covers have similar performance
- Vertical parallelization might be more important than horizontal containment

Future work:
Benchmarking of workload-aware graph cover strategies
Thank you for your Attention!

On Data Placement Strategies in Distributed RDF Stores

Daniel Janke, Steffen Staab, Matthias Thimm

Contributions:

1) Benchmark methodology showing the interdependencies of graph cover strategies and query processing

2) A flexible open-source platform for performing the benchmark

3) Analysis indicating that vertical parallelization may dominate horizontal containment
References

[Görlitz2012SSG]

[GurajadaTheobald2014TAD]

[Harth2007YAF]

[Huang2011SSQ]
References

[Käfer2014BTC]

[Karypis1998AFA]

[Lee2013EDP]

[Lee2013SQO]

[Wu2014SAS]
References

[Zeng2013ADG]

[Zhang2013ETS]
Hash Cover [e.g. Harth2007YAF]

Triple placement bases on subject hash modulo number of compute nodes

\[
\text{hashCover}(<s, p, o>) := \text{hash}(s) \mod |C|
\]
Hierarchical Hash Cover \cite{Lee2013SQO}

Triple placement bases on prefixes of subject IRIs

\[
\text{hashCover}(< s, p, o >) := \begin{cases} 
\text{hash}(\text{prefix}(s)) \mod |C| , \text{ if } s \in I \\
\text{hash}(s) \mod |C| , \text{ otherwise}
\end{cases}
\]

- **IRI**: \texttt{http://www.w3.org/1999/02/22-rdf-syntax-ns#type}
- **Path hierarchy**: \texttt{org/w3/www/1999/02/22-rdf-syntax-ns/type}
- **Determine path hierarchy prefix such that**
  - There exist at least \(|C|\) hierarchy prefixes
  - That are shared by at least \(\omega\%\) of all triples
Minimal Edge-Cut Cover

Tries to solve the k-way graph partitioning problem [Karypis1998AFA]

1) Assign vertices (subjects and objects) to partitions such that
   - Number of edges between vertices of different partitions is minimized and
   - Each partition contains approximately \( \frac{|V_G|}{|C|} \) vertices

2) Assign triple to the partition its subject is assigned to
Chunk Sizes

- Minimal edge-cut cover has most unbalanced chunks
- Hash-based covers have equally-sized chunks