Impact Analysis of Data Placement Strategies on Query Efforts in Distributed RDF Stores

ISWC 2018
Appeared in

Daniel Janke, Steffen Staab, Matthias Thimm
12.10.2018

WeST
People and Knowledge Networks
Distributed RDF Stores

= RDF store distributed over several compute nodes

<g:Gesis e:employs g:wanja>
<g:wanja f:givenname “Wanja”>
<g:bello r:type g:Dog>
<g:bello e:ownedBy g:wanja>

<g:wanja f:knows w:daniel>
<w:WeST e:employs w:martin>
<g:bello e:ownedBy g:wanja>
<w:daniel f:givenname “Daniel”>
<w:daniel f:knows w:martin>
Why using distributed RDF Stores?

Example 1: Wikidata

- Dataset size: 4.9 billion triples (as of April 2018)
- Stored in distributed BlazeGraph RDF store because
  - Faster execution of queries
  - Higher availability

Example 2: BBC

- On average 1 million SPARQL queries per day (in 2010)
- Stored in distributed GraphDB RDF store because
  - Faster execution of queries
  - Higher availability
Challenges of RDF Stores in the Cloud

- How to design the architecture?
- How to distribute the data?
- How to identify compute nodes that store required data?
- How to distribute query processing?
- How to achieve fault tolerance?
- How to evaluate?

...
Challenges of RDF Stores in the Cloud

• How to design the architecture?

• How to distribute the data?

• How to identify compute nodes that store required data?

• How to distribute query processing?

• How to achieve fault tolerance?

• How to evaluate?

...
**Terminology: Graph Cover**

**Graph cover** *(aka sharding)*
Assignment of each triple to at least one compute node

**Graph chunk** *(aka shards)*
Set of triples assigned to a single compute node
Common Graph Cover Strategies

Hash cover [e.g. Harth et al. 2007]
- Triple placement bases on subject hash modulo number of compute nodes

Hierarchical cover [Lee & Liu 2013b]
- Triple placement bases on hash of subject IRI prefixes

Minimal edge-cut cover [Karypis & Kumar 1998]
- 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
**Horizontal Containment**

```
SELECT ?org ?name WHERE
{ ?org e:employs ?pers . ?pers f:givenname ?name }
```

<table>
<thead>
<tr>
<th>?org</th>
<th>?name</th>
</tr>
</thead>
<tbody>
<tr>
<td>g:Gesis</td>
<td>“Wanja”</td>
</tr>
<tr>
<td>w:WeST</td>
<td>“Martin”</td>
</tr>
<tr>
<td>w:WeST</td>
<td>“Daniel”</td>
</tr>
</tbody>
</table>

- Computation of individual query results on local data
- Indicator for robust query processing when scaling horizontally
Vertical Parallelization

```
SELECT ?org ?name WHERE
  {?org e:employs ?pers . ?pers f:givenname ?name}
```

<table>
<thead>
<tr>
<th>?org</th>
<th>?name</th>
</tr>
</thead>
<tbody>
<tr>
<td>g:Gesis</td>
<td>“Wanja”</td>
</tr>
<tr>
<td>w:WeST</td>
<td>“Martin”</td>
</tr>
<tr>
<td>w:WeST</td>
<td>“Daniel”</td>
</tr>
</tbody>
</table>

- Parallel computation of different query results on different compute nodes
- Indicator for query processing scaling with growing result set sizes when scaling horizontally
Comparing Complete RDF Stores

Car 1 using wheel A

Car 2 using wheel B

Does wheel A or B allow for a higher speed?

**Pros:**
- Easy to perform
- Reflects behaviour of real distributed RDF store

**Cons:**
- Uncertainty whether observations are related to graph cover strategy

Images from https://openclipart.org
Evaluation With Batch Processing Frameworks

Evaluate alternative implementations of, e.g. data placement strategies, within batch processing frameworks

Pros:
- Observations are related to graph cover strategy

Cons:
- Uncertainty whether observations apply for real distributed RDF stores

Which effect has amount of transferred data on query performance?

Images from https://openclipart.org
Our Novel Evaluation Methodology

- Time-dependent and -independent measurements
- Observations are related to graph cover strategy
- Observations reflect behaviour of real distributed RDF stores
Dataset & Queries

Dataset:
• 1 billion triples subset of BTC2014 [Käfer & Harth 2014] (more in the journal article)

Queries:
• SPARQL queries generated by SPLODGE [Görlitz et al. 2012]
• Query characteristics
  – #triple patterns: 2 and 8
  – Join pattern: path-shaped and star-shaped
  – Selectivity: 0.001% and 0.01% (1M and 10M triples)
Koral

- Graph cover independent distributed RDF store
- Inspired by TriAD [Gurajada et al. 2014]
Evaluation Measures

Overall performance

- Query execution time

Horizontal Containment

- Packet transfer $P$:
  Packets of 100 intermediate variable bindings transferred between compute nodes

Vertical Parallelization (VP)

Measured by combination of

- Packet transfer
- Workload imbalance $W$:
  Gini coefficient of join comparisons on each compute node

<table>
<thead>
<tr>
<th>$P$ low</th>
<th>$P$ high</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W$ high</td>
<td>low VP</td>
</tr>
<tr>
<td>$W$ low</td>
<td>high VP</td>
</tr>
</tbody>
</table>

(More measures in the journal article)
Experimental Setup

Compared graph cover strategies:

- Hash cover
- Hierarchical hash cover
- Minimal edge-cut cover

(Vertical cover, $n$-hop replication, ... in the journal article)

Computational environment:

- 1 Master à 4 cores, 64 GB RAM, 1 TB HDD
- 10, 20, 40 Slaves à 1 core, 2 GB RAM, 300 GB HDD
- 1 Gbit Ethernet
Overall Query Performance (10 Slaves)

- Minimal edge-cut causes slowest query execution in most cases
- None of the hash-based covers is faster in general
Impact Analysis of Data Placement Strategies

Horizontal Containment (10 Slaves)

- Star-shaped queries produce no transferred packets
- Minimal edge-cut covers produces up to 38% less packet transfer
- Hash-based covers similar packet transfer
**Workload Imbalance (10 Slaves)**

- Workload of minimal edge-cut cover is most unbalanced
- Hash-based covers have similar balanced workloads
## Vertical Parallelization

<table>
<thead>
<tr>
<th></th>
<th>Packet transfer low</th>
<th>Packet transfer high</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Workload Imbalance</strong></td>
<td><strong>Minimal Edge-Cut Cover</strong></td>
<td><strong>Hash Cover</strong></td>
</tr>
<tr>
<td>high</td>
<td></td>
<td>Hierarchical Hash Cover</td>
</tr>
<tr>
<td>low</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Overall performance:**

Minimal edge-cut cover slower than hash-based covers
Effect of Scaling Number of Slaves

For all graph cover strategies:

- Total computational effort spread over more compute nodes
- Workload imbalance increases
- Horizontal containment decreases
- Vertical parallelization decreases

=> scaling 10 to 20 slaves leads to faster overall performance

=> scaling 20 to 40 slaves leads to slower overall performance

(Effect of scaling dataset size in the journal article)
Conclusion

• Evaluation methodology for graph cover strategies
• Evaluation of common graph cover strategies
  – Expected results
    ▪ Minimal edge-cut has best horizontal containment
    ▪ Scaling up number of slaves reduces horizontal containment
  – Surprising Results
    ▪ Hash-based covers have a better overall performance
    ▪ Both hash-based covers perform almost the same
    ▪ Workload imbalance increases when number of slaves is increased

More in:
Conclusion

- Evaluation methodology for graph cover strategies
- Evaluation of common graph cover strategies
  - Expected results
    - Minimal edge-cut has best horizontal containment
    - Scaling up number of slaves reduces horizontal containment
  - Surprising Results
    - Hash-based covers have a better overall performance
    - Both hash-based covers perform almost the same
    - Workload imbalance increases when number of slaves is increased

More in:

Thank you for your Attention!
References

[Görlitz et al. 2012]


[Gurajada et al. 2014]


[Harth et al. 2007]


[Huang et al. 2011]

References

[Käfer & Harth 2014]

[Karypis & Kumar 1998]

[Lee & Liu 2013a]

[Lee & Liu 2013b]

[Wu et al. 2014]
References

[Zeng et al. 2013]

[Zhang et al. 2013]