Efficient Export of SPARQL Query Results

Bachelor’s Thesis by Robin Textor-Falconi
Introduction
What is this thesis about?

- QLever development
  - New features
  - Enhancement of existing features
  - Evaluation of different approaches
What is QLever?

- SPARQL engine
- Suited for knowledge bases with billions of triples
- Read-only (for now)
SPARQL

- A W3C recommendation
- Protocol based on HTTP/1.1
- Variety of queries
  - SELECT
  - CONSTRUCT
  - ASK
  - DESCRIBE
SELECT Queries

PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?title ?genre WHERE {
  FILTER (LANG(?title) = "en") .
  FILTER (LANG(?genre) = "en") .
}
ORDER BY ASC(?title)
OFFSET 59666
LIMIT 5

<table>
<thead>
<tr>
<th>?title</th>
<th>?genre</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;For Queen and Country&quot;@en</td>
<td>&quot;drama&quot;@en</td>
</tr>
<tr>
<td>&quot;For Queen and Country&quot;@en</td>
<td>&quot;crime film&quot;@en</td>
</tr>
<tr>
<td>&quot;Forrest Gump&quot;@en</td>
<td>&quot;drama&quot;@en</td>
</tr>
<tr>
<td>&quot;Forrest Gump&quot;@en</td>
<td>&quot;comedy film&quot;@en</td>
</tr>
<tr>
<td>&quot;Forrest Gump&quot;@en</td>
<td>&quot;coming-of-age fiction&quot;@en</td>
</tr>
</tbody>
</table>
CONSTRUCT Queries

```sparql
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
CONSTRUCT { ?genre <http://example.com/hasFilmTitled> ?title } WHERE {
  FILTER (LANG(?title) = "en") .
  FILTER (LANG(?genre) = "en") .
}
ORDER BY ASC(?title)
OFFSET 59666
LIMIT 5
```

```
drama"en <http://example.com/hasFilmTitled>
"For Queen and Country"en .

"crime film"en <http://example.com/hasFilmTitled>
"For Queen and Country"en .

"drama"en <http://example.com/hasFilmTitled>
"Forrest Gump"en .

"comedy film"en <http://example.com/hasFilmTitled>
"Forrest Gump"en .

"coming-of-age fiction"en <http://example.com/hasFilmTitled>
"Forrest Gump"en .
```
Problem Definition
Obvious goals

- CONSTRUCT query support
- Queries as fast as possible
Memory consumption

Client

Request
Compute
Serialize
Respond

QLever
Average Internet connection speed

Source: https://www.speedtest.net/global-index
Implementation
### CONSTRUCT Queries

```
SELECT ?title ?genre

CONSTRUCT {
  ?genre <http://example.com/hasFilmTitled> ?title
}
```

<table>
<thead>
<tr>
<th>?title</th>
<th>?genre</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;For Queen and Country&quot;</td>
<td>&quot;drama&quot;@en</td>
</tr>
<tr>
<td>&quot;For Queen and Country&quot;</td>
<td>&quot;crime film&quot;@en</td>
</tr>
<tr>
<td>&quot;Forrest Gump&quot;</td>
<td>&quot;drama&quot;@en</td>
</tr>
<tr>
<td>&quot;Forrest Gump&quot;</td>
<td>&quot;comedy film&quot;@en</td>
</tr>
<tr>
<td>&quot;Forrest Gump&quot;</td>
<td>&quot;coming-of-age fiction&quot;@en</td>
</tr>
</tbody>
</table>
Chunked Transfer Encoding

Client

QLever

Request

Compute

Serialize

Respond

Serialize

Respond

Serialize

Respond
Chunked Transfer Encoding

HTTP/1.1 200 OK
Transfer-Encoding: chunked

6
Hello,
7
  World!
0
On-the-fly compression
Asynchronous Processing

Client

QLever

Request
Compute

Serialize
Respond
Serialize
Respond
Serialize
Respond
Respond
Asynchronous Processing

Thread 1
- Compress
- Send
- Compress
- Send
- Compress
- Send

Thread 2
- Generate
- Generate
- Generate
2 Asynchronous Layers

Thread 1
- Send
- Send
- Send

Thread 2
- Compress
- Compress

Thread 3
- Generate
- Generate
- Generate
Evaluation
Benchmark scenario

- System “Wolga” in docker environment (Ubuntu 20.04)
  - AMD Ryzen 7 3700X 8/16 x 3.6-4.4GHz
  - 128GB DDR4
  - 3.4TB SSD RAID 0
- 4 hand-picked queries were compared
- Duration of body transport was measured
  - Divided by result size to get comparable throughput
  - Average of 3 runs each
Comparing compression approaches

● Zlib implementation
  ○ Different compression levels
    ■ Fast - 1
    ■ Default - 6
    ■ Best - 9
  ○ Each with different overhead and compression ratio

● Effective throughput is used as a metric
  ○ Measures throughput as if the data was uncompressed
  ○ Actual throughput is listed in parentheses
# Compression benefits

<table>
<thead>
<tr>
<th></th>
<th>None</th>
<th>Fast</th>
<th>Default</th>
<th>Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query A</td>
<td>106.18</td>
<td>(11.16)</td>
<td>55.14</td>
<td>(6.16)</td>
</tr>
<tr>
<td>Query B</td>
<td>123.74</td>
<td>(8.50)</td>
<td>76.75</td>
<td>(4.75)</td>
</tr>
<tr>
<td>Query C</td>
<td>457.53</td>
<td>(9.94)</td>
<td>192.83</td>
<td>(4.45)</td>
</tr>
<tr>
<td>Query D</td>
<td>139.71</td>
<td>(3.38)</td>
<td>105.03</td>
<td>(1.90)</td>
</tr>
</tbody>
</table>

Units in MB/s
## Asynchronous processing

<table>
<thead>
<tr>
<th></th>
<th>Compression and transport on same thread</th>
<th>Compression and transport on 2 threads</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>None</td>
<td>Fast</td>
</tr>
<tr>
<td>Query A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>109.16</td>
<td>108.48</td>
<td>(21.97)</td>
</tr>
<tr>
<td>Query B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>126.02</td>
<td>125.01</td>
<td>(13.85)</td>
</tr>
<tr>
<td>Query C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>477.07</td>
<td>304.25</td>
<td>(15.68)</td>
</tr>
<tr>
<td>Query D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>141.40</td>
<td>139.10</td>
<td>(4.48)</td>
</tr>
</tbody>
</table>

Units in MB/s
Thank you for watching!
Efficient Export of SPARQL Query Results

Bachelor’s Thesis by Robin Textor-Falconi
Appendix
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?person_id ?person (COUNT(?profession_id) AS ?count)
    (GROUP_CONCAT(?profession; separator="", " ) AS ?professions) WHERE {

FILTER (LANG(?person) = "en") .
FILTER (LANG(?profession) = "en")

}

GROUP BY ?person_id ?person
ORDER BY DESC(?count)
LIMIT 18446744073709551615
SELECT ?person_id ?person WHERE {
}
GROUP BY ?person_id ?person
ORDER BY DESC(?count)
LIMIT 18446744073709551615
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
SELECT ?x ?y WHERE {
  ?x wdt:P31 ?y
}
LIMIT 18446744073709551615
PREFIX wdt: <http://www.wikidata.org/prop/direct/>

CONSTRUCT {
  [a ()]
}

WHERE {
  ?x wdt:P31 ?y
}

LIMIT 18446744073709551615
HTTP and TCP

Close

TCP
HTTP
HTTP
TCP
TCP

Keep-alive

TCP
HTTP
HTTP
TCP
TCP

Handshake
Ready
Close
Redundant string copies

Redundant Copy

Prefix

Prefix + Data

Data

Buffer

Direct Write

Prefix

Prefix + Data

Data

Buffer