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 {
  ?film wdt:P31 wd:Q11424 .
  ?film rdfs:label ?title .
  FILTER (LANG(?title) = "en") .
  ?film wdt:P136 ?genre_id .
  ?genre_id rdfs:label ?genre .
  FILTER (LANG(?genre) = "en") .
}
ORDER BY ASC(?title)
OFFSET 59666
LIMIT 5
```

?title	?genre
"For Queen and Country"@en	"drama"@en
"For Queen and Country"@en	"crime film"@en
"Forrest Gump"@en	"drama"@en
"Forrest Gump"@en	"comedy film"@en
"Forrest Gump"@en	"coming-of-age fiction"@en

CONSTRUCT 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#>
CONSTRUCT {
  ?genre <http://example.com/hasFilmTitled> ?title
} WHERE {
  ?film wdt:P31 wd:Q11424 .
  ?film rdfs:label ?title .
  FILTER (LANG(?title) = "en") .
  ?film wdt:P136 ?genre_id .
  ?genre_id rdfs:label ?genre .
  FILTER (LANG(?genre) = "en") .
}
ORDER BY ASC(?title)
OFFSET 59666
LIMIT 5
```

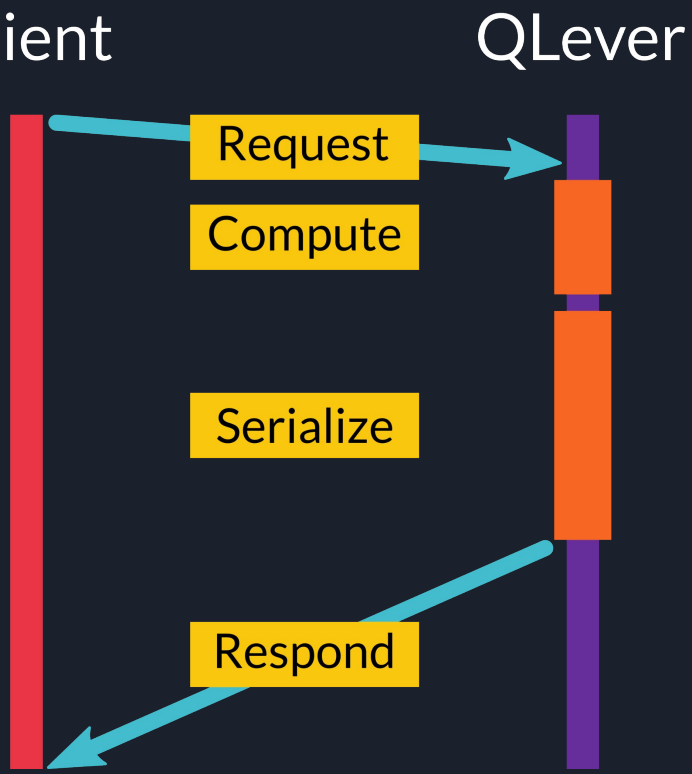
```
"drama"@en <http://example.com/hasFilmTitled>
  "For Queen and Country"@en .

"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

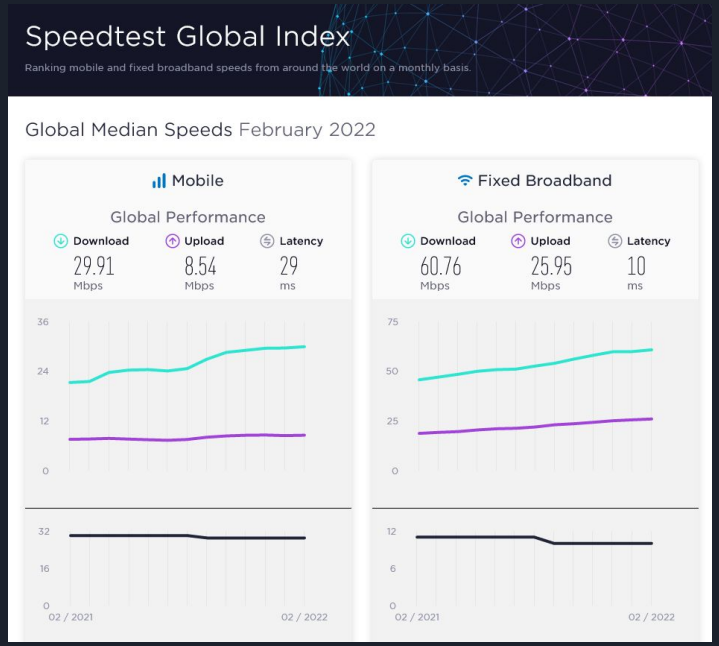
Obvious goals

- CONSTRUCT query support
- Queries as fast as possible

Memory consumption



Average Internet connection speed



Source: https://www.speedtest.net/global-index

Implementation

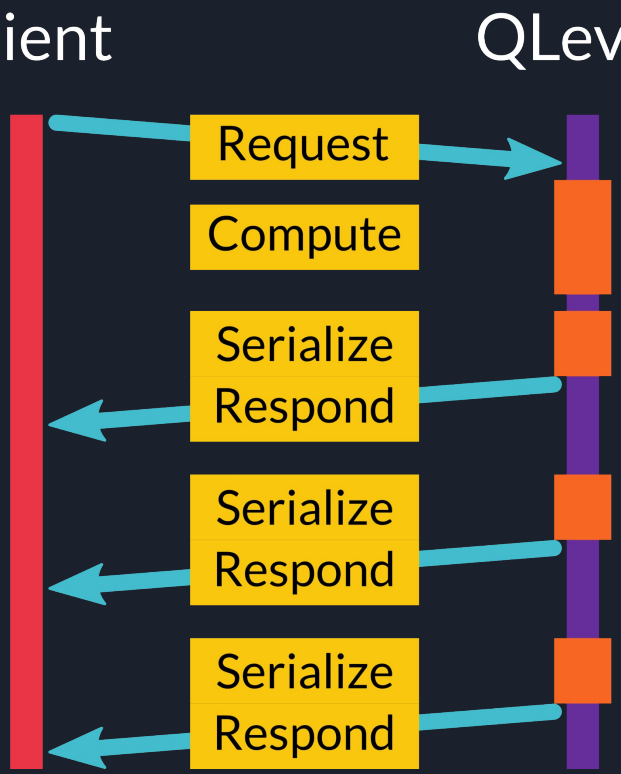
CONSTRUCT Queries

```
SELECT ?title ?genre
```

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

?title	?genre
"For Queen and Country"@en	"drama"@en
"For Queen and Country"@en	"crime film"@en
"Forrest Gump"@en	"drama"@en
"Forrest Gump"@en	"comedy film"@en
"Forrest Gump"@en	"coming-of-age fiction"@en

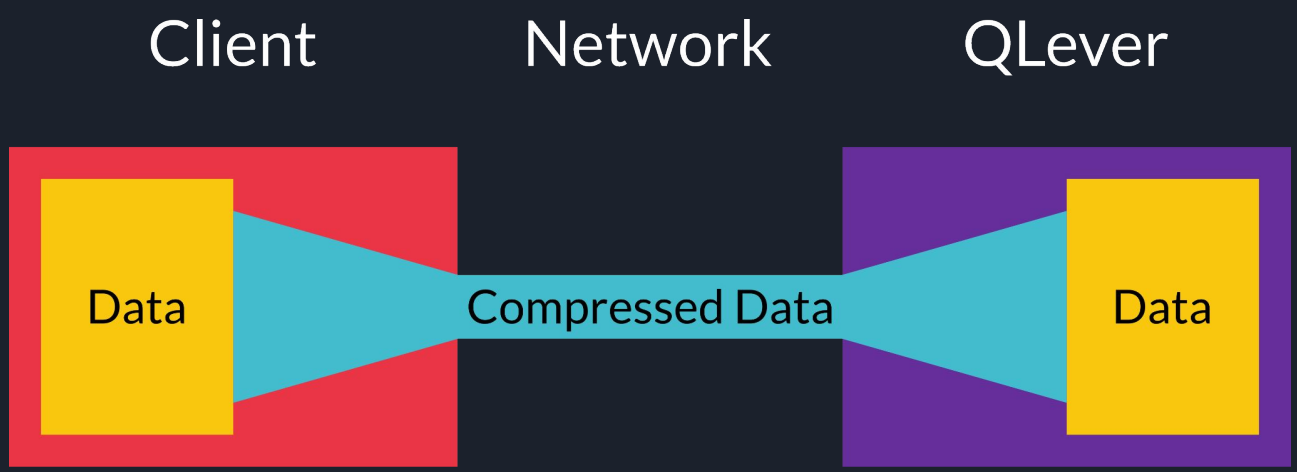
Chunked Transfer Encoding



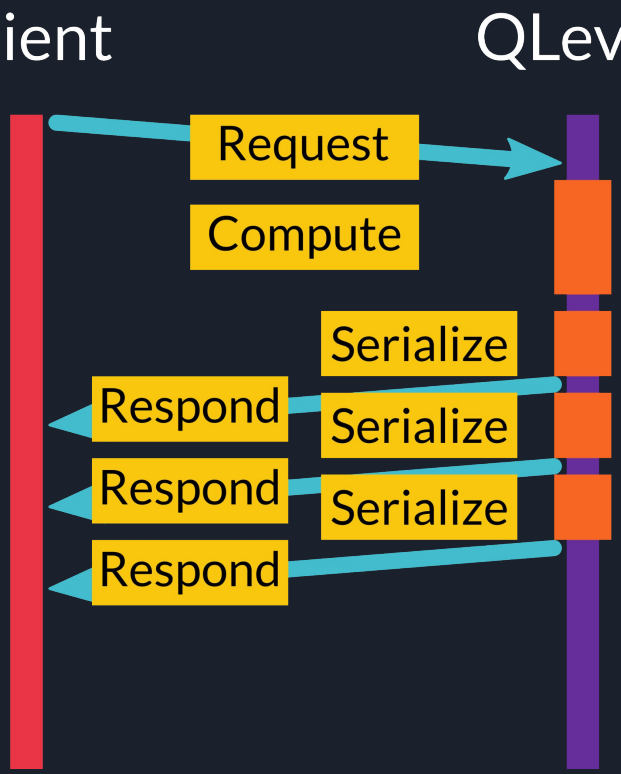
Chunked Transfer Encoding

```
HTTP/1.1 200 OK
Transfer-Encoding: chunked

6
Hello,
7
 World!
0
```

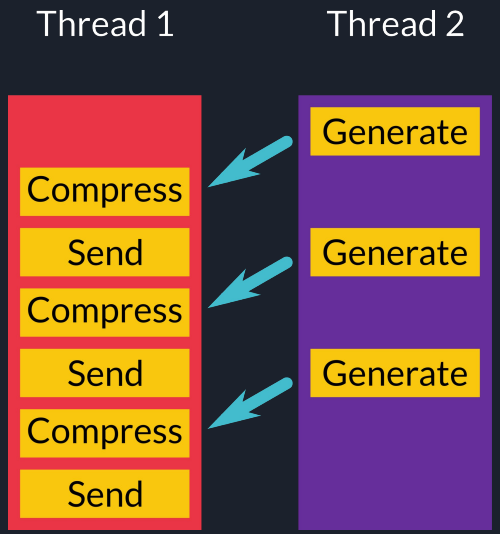
On-the-fly compression



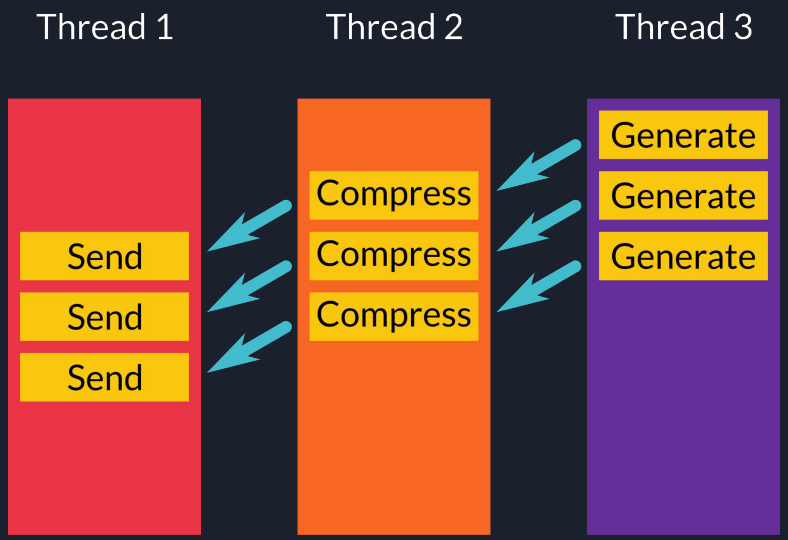
Asynchronous Processing



Asynchronous Processing



2 Asynchronous Layers



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

	None	Fast		Default		Best	
Query A	106.18	(11.16)	55.14	(6.16)	35.99	(3.02)	18.01
Query B	123.74	(8.50)	76.75	(4.75)	48.43	(2.22)	23.75
Query C	457.53	(9.94)	192.83	(4.45)	99.75	(2.09)	51.39
Query D	139.71	(3.38)	105.03	(1.90)	68.43	(1.02)	39.70

Units in MB/s

Asynchronous processing

	Compression and transport on same thread						Compression and transport on 2 threads					
	None	Fast		Default		Best		Fast		Default		Best	
Query A	109.16	(21.97)	108.48	(9.10)	53.19	(3.57)	21.35	(21.92)	108.30	(9.12)	53.28	(3.59)	21.43
Query B	126.02	(13.85)	125.01	(7.46)	75.94	(2.70)	28.90	(13.76)	124.28	(7.45)	75.89	(2.71)	29.03
Query C	477.07	(15.68)	304.25	(5.42)	121.55	(2.31)	56.76	(15.95)	309.44	(5.44)	121.95	(2.30)	56.60
Query D	141.40	(4.48)	139.10	(3.56)	128.74	(1.44)	55.87	(4.41)	137.04	(3.60)	130.19	(1.45)	56.00

Units in MB/s

Thank you for watching!

Efficient Export of SPARQL Query Results

Bachelor's Thesis by Robin Textor-Falconi

Appendix

Query A

```
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 {

  ?person_id wdt:P31 wd:Q5 .
  ?person_id wdt:P106 ?profession_id .
  ?profession_id rdfs:label ?profession .
  ?person_id rdfs:label ?person .
  FILTER (LANG(?person) = "en") .
  FILTER (LANG(?profession) = "en")
}
GROUP BY ?person_id ?person
ORDER BY DESC(?count)
LIMIT 18446744073709551615
```

Query B

```
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 WHERE {
  ?person_id wdt:P31 wd:Q5 .
  ?person_id rdfs:label ?person .
}
GROUP BY ?person_id ?person
ORDER BY DESC(?count)
LIMIT 18446744073709551615
```

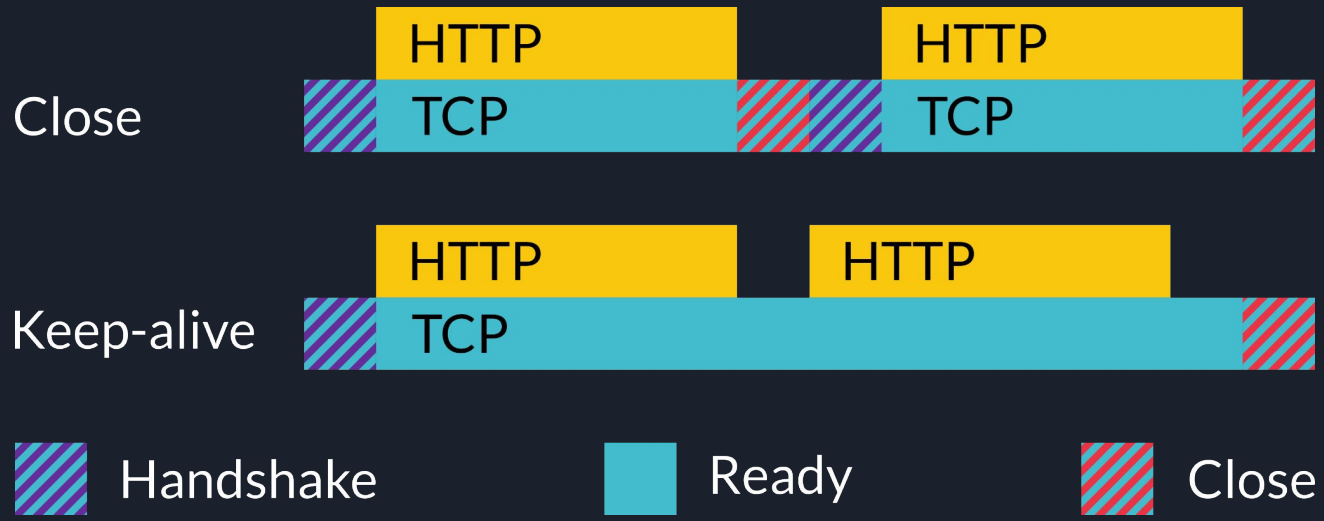
Query C

```
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
SELECT ?x ?y WHERE {
  ?x wdt:P31 ?y
}
LIMIT 18446744073709551615
```

Query D

```
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
CONSTRUCT {
  ?x wdt:P31 ?y .
  ?x wdt:P31 ?y .
  [a ()]
} WHERE {
  ?x wdt:P31 ?y
}
LIMIT 18446744073709551615
```

HTTP and TCP


Close
HTTP
TCP
HTTP
TCP
Keep-alive
HTTP
TCP
HTTP
Handshake
Ready
Close

Redundant string copies