FedShop

A Benchmark for Testing the Scalability of SPARQL Federation Engines

https://github.com/GDD-Nantes/FedShop

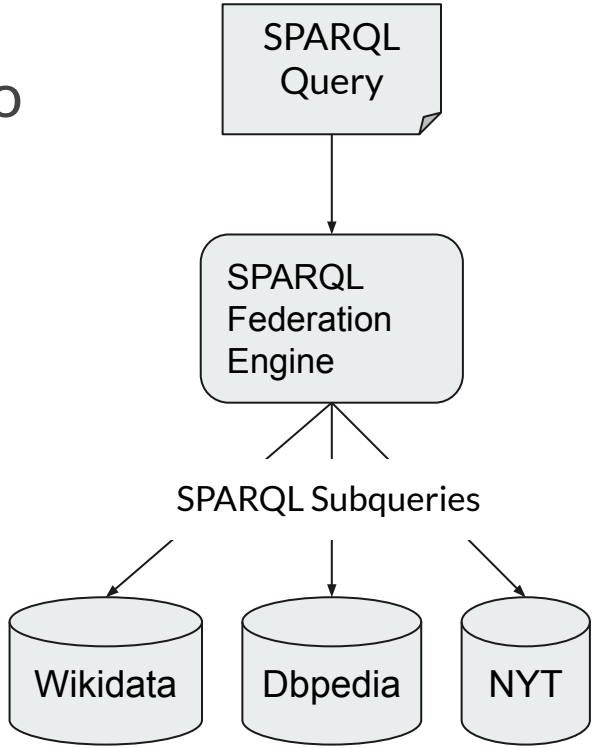
Minh-Hoang Dang , Julien Aimonier-Davat , Pascal Molli , **Olaf Hartig** , Hala Skaf-Molli and Yotlan Le Crom

Nantes Université

LINKÖPING UNIVERSITY

SPARQL Federation Engines

- SPARQL federation engines allow users to query a federation of SPARQL endpoints.
- FedX, SemaGrow, Anapsid, Splendid, CostFed, Odyssey…



FedBench Q3: US presidents, their party membership and news pages about them

```
SELECT ?president ?party ?page WHERE {
   ?president a dbo:President .
   ?president dbo:nationality dbr:United_States .
   ?president dbo:party ?party .
   ?x nyt:topicPage ?page .
   ?x owl:sameAs ?president .
}
```

SPARQL Federation Engine

```
SELECT ?president ?party ?page WHERE {
SERVICE <http://dbpedia/sparql> {
 ?president a dbo:President .
 ?president dbo:nationality dbr:United_States .
 ?president dbo:party ?party .
}
SERVICE <http://newyork-times/sparql> {
  ?x nyt:topicPage ?page .
  ?x owl:sameAs ?president .
}
```

Federation = a set of SPARQL endpoints

DBpedia, NYTimes, DrugBank, SWDogFood, GeoNames, KEGG, ChEBI, Jamendo, LinkedMDB

Schmidt, Michael, et al. "Fedbench: A benchmark suite for federated semantic data query processing." The 10th International Semantic Web Conference, ISWC 2011

FedBench Q3: US presidents, their party membership and news pages about them

```
SELECT ?president ?party ?page WHERE {
   ?president a dbo:President .
   ?president dbo:nationality dbr:United_States .
   ?president dbo:party ?party .
   ?x nyt:topicPage ?page .
   ?x owl:sameAs ?president .
}
```

SPARQL Federation Engine

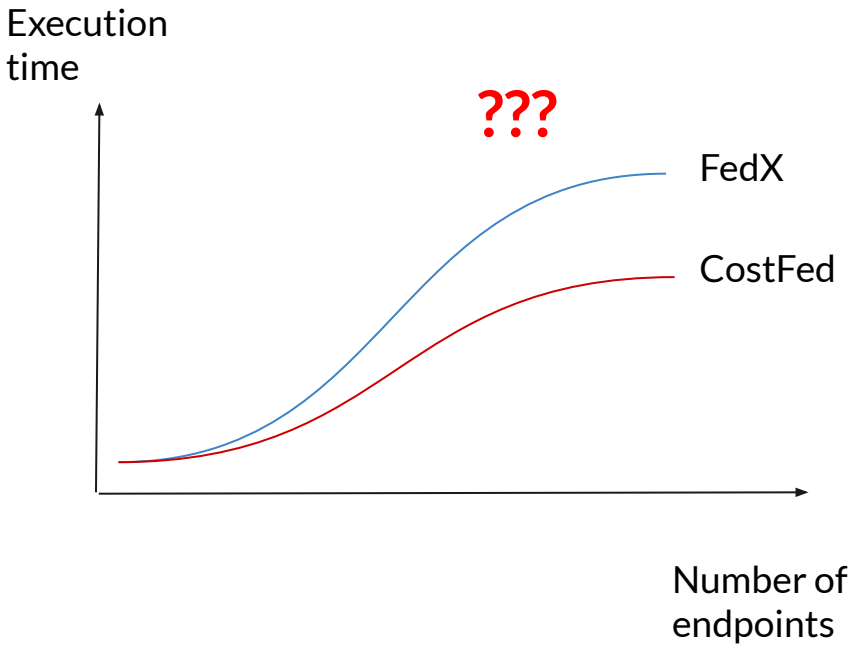
```
SELECT ?president ?party ?page WHERE {

 SERVICE <http://dbpedia/sparql> {
  ?president a dbo:President .
  ?president dbo:nationality dbr:United_States .
  ?president dbo:party ?party .
 }

 SERVICE <http://newyork-times/sparql> {
   ?x nyt:topicPage ?page .
   ?x owl:sameAs ?president .
 }
```

What happens if the size of the federation increases ?

DBpedia, NYTimes, DrugBank, SWDogFood, GeoNames, KEGG, ChEBI, Jamendo, LinkedMDB

Problem

There is **no benchmark** for evaluating the **scalability** of existing SPARQL federation engines, in terms of an increasing number of endpoints.



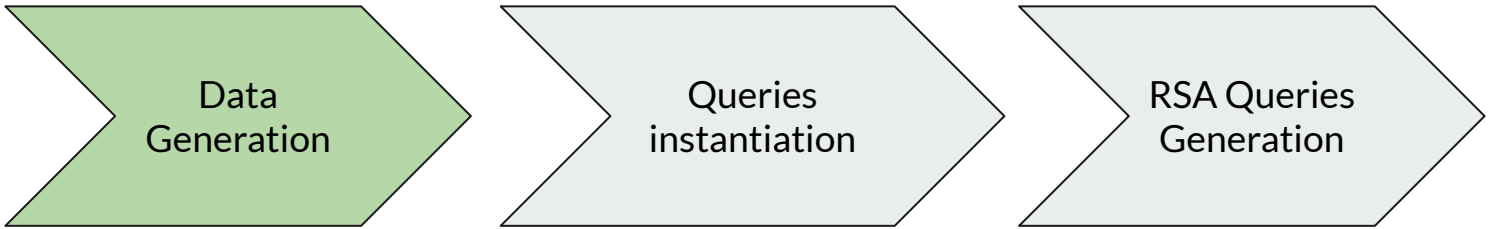
The FedShop Benchmark

Key ideas

- Simulate a federated e-commerce marketplace where users search for products in **a federation of shops** :
  - Find products for a given set of generic features.
  - Retrieve basic information about a specific product.
  - Find products that are similar to a specific product
  - Retrieve in-depth information about a specific product including offers and reviews.
- Similar to Berlin SPARQL Benchmark (BSBM) [1] but with a federation of shops.

[1] Bizer, Christian, and Andreas Schultz. "The berlin sparql benchmark." *International Journal on Semantic Web and Information Systems (IJSWIS)* 5.2 (2009): 1-24.

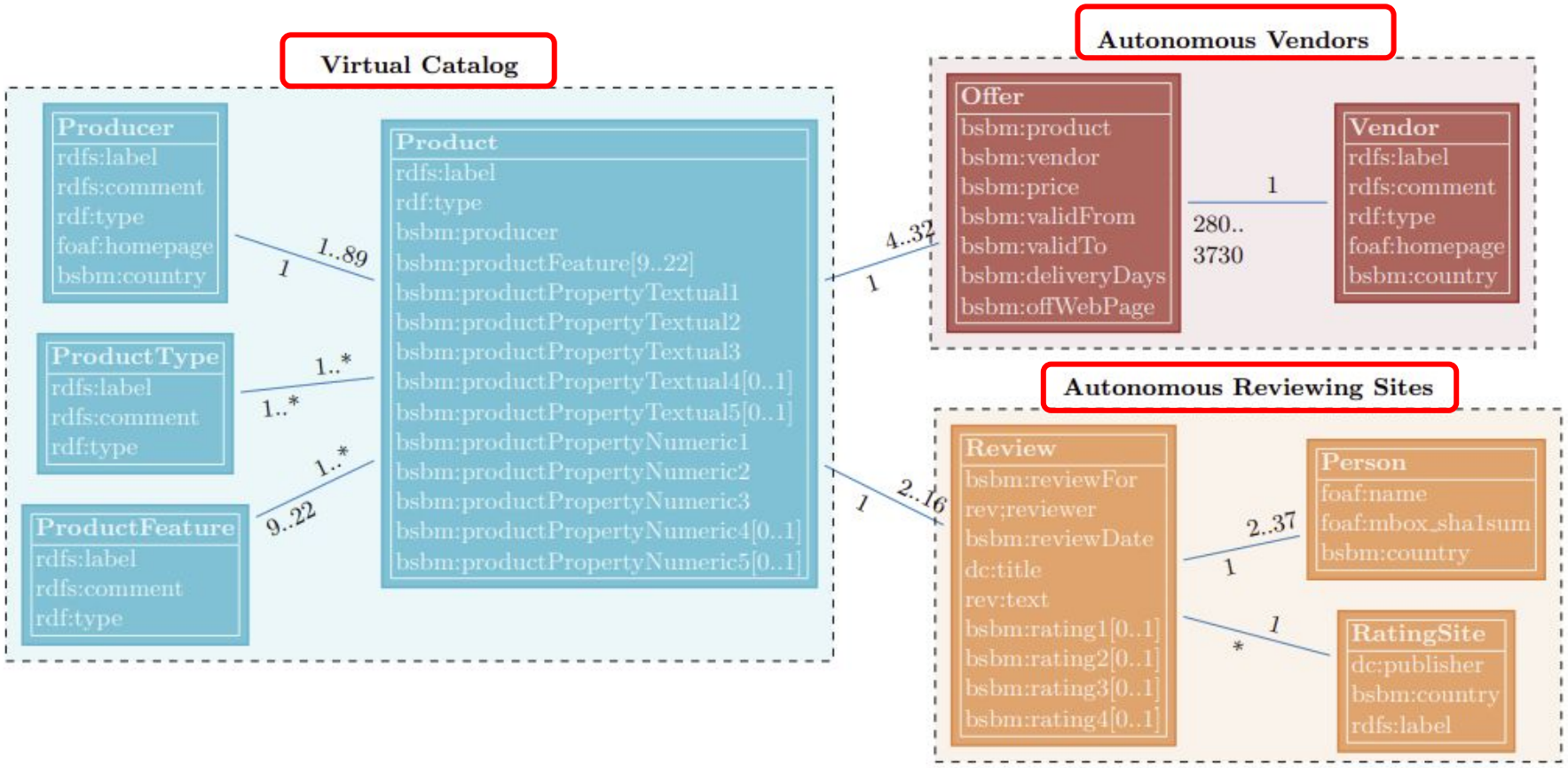
FedShop:
Data Generation



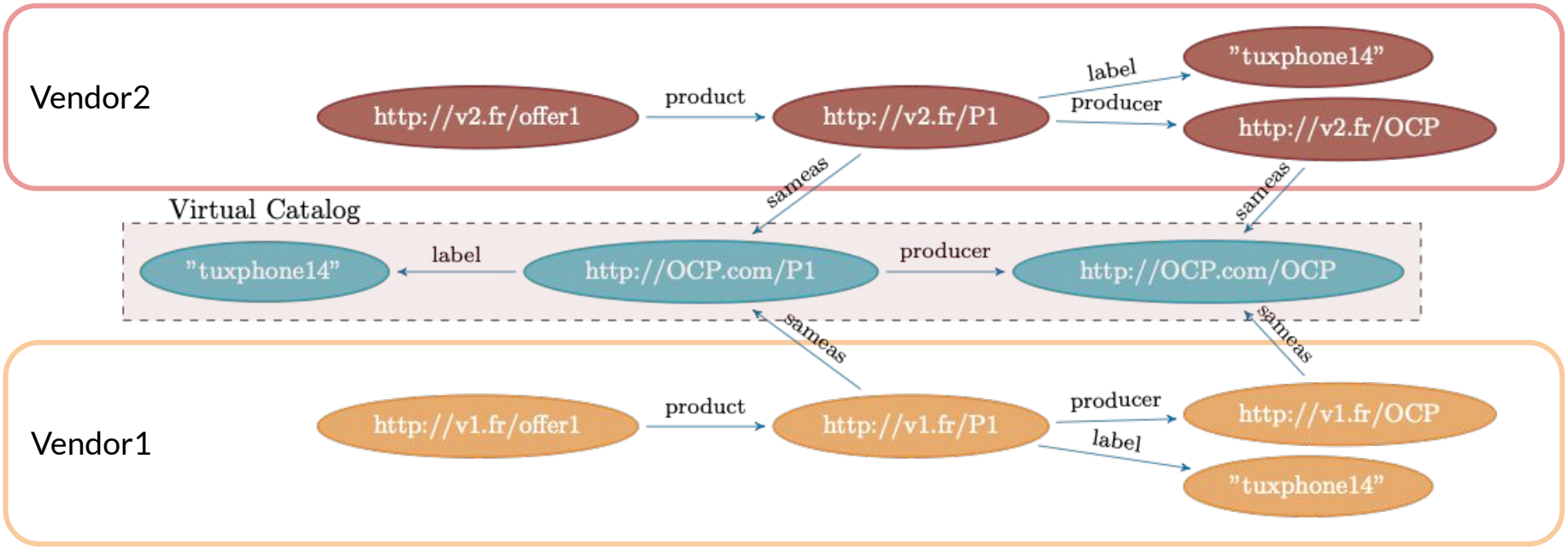
Data generation principles

- We consider a **virtual catalog** of products shared by all vendors.
- Each vendor sells a *subset* of products of the virtual catalog.
- Each vendor is **autonomous**:
  - Capable of operating **independently** from other vendors.
  - If a shop sells a product, it will have a **local URI** for that product and all reachable informations.

FedShop Schema (adapted from BSBM)



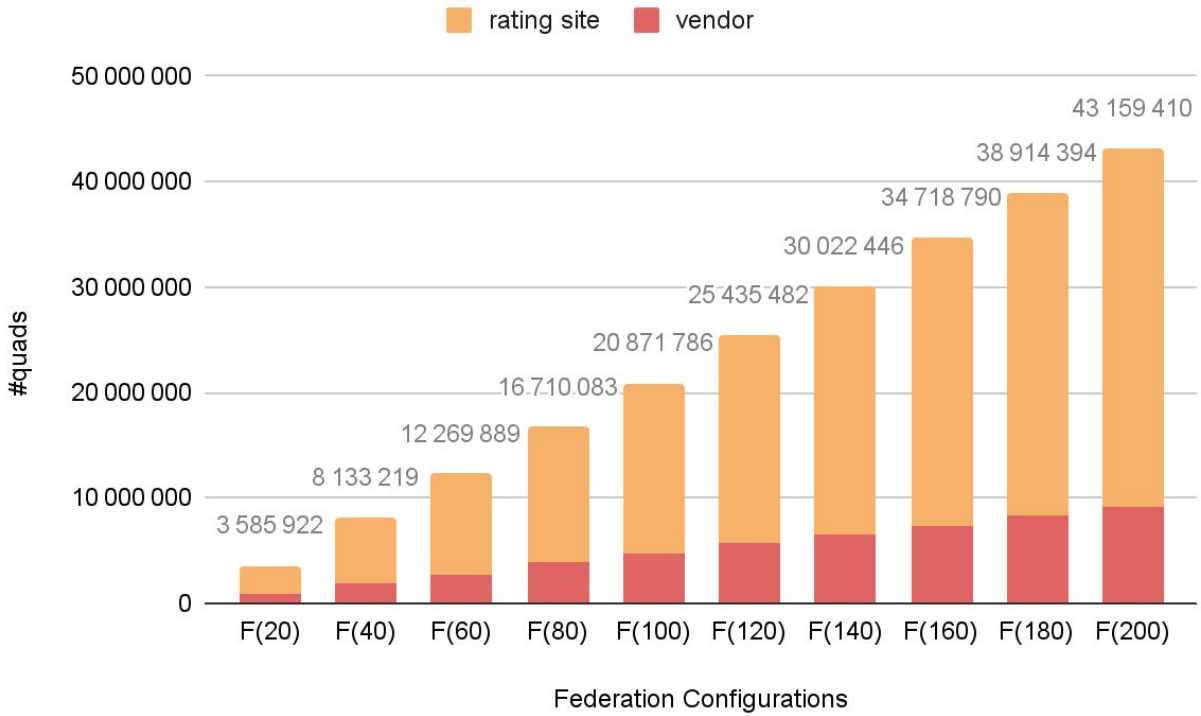
Two shops selling the same product P1



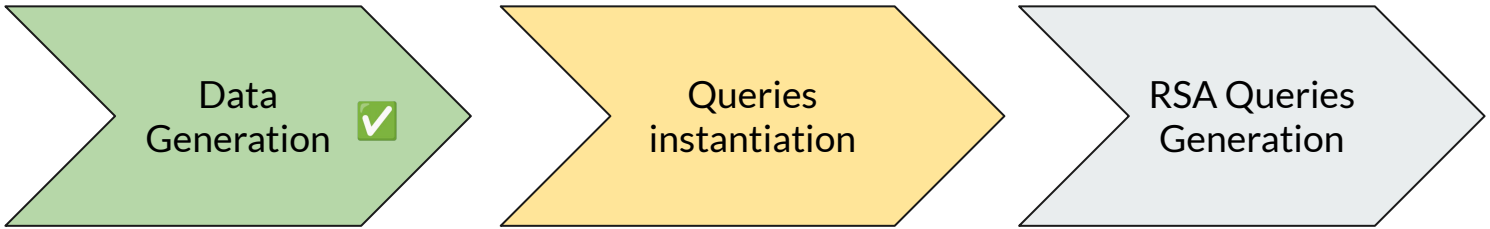
FedShop200 : a configuration of FedShop

- **Virtual catalog** of 200,000 products.
- Each Vendor has ~2,000 offers.
- Each RatingSite has ~10,000 reviews.
- **10 Federations:**
  - F(20) = 10 shops + 10 rating sites
  - F(40) = F(20) + 10 new shops and 10 rating sites
  - F(60) = F(40) + 10 new shops and 10 rating sites
  - …
  - F(200) = F(180) + 10 new shops and 10 rating sites

FedShop200 : https://zenodo.org/records/7919872



FedShop:
The Queries



FedShop Queries

- We adopt the explore use case of BSBM [1] composed of **12 query templates :**
  - Find products for a given set of generic features.
  - Retrieve basic information about a specific product.
  - Find products that are similar to a specific product.
  - Retrieve in-depth information about a specific product including offers and reviews.
- Rewrite BSBM [1] query templates to FedShop query templates.

[1] Bizer, Christian, and Andreas Schultz. "The berlin sparql benchmark." *International Journal on Semantic Web and Information Systems (IJSWIS)* 5.2 (2009): 1-24.

Rewrite BSBM query templates to FedShop query templates

```
SELECT DISTINCT ?product ?label
WHERE {
 ?product rdfs:label ?label .
 ?product a %ProductType% .
 ?product bsbm:productFeature %ProductFeature1% .
 ?product bsbm:productFeature %ProductFeature2% .
 ?product bsbm:productPropertyNumeric1 ?value1 .
 FILTER (?value1 > %x%)
}
ORDER BY ?label
LIMIT 10
```

(a) Q1 BSBM query

```
SELECT DISTINCT ?product ?label
WHERE {
    ?product rdfs:label ?label .

    ?product rdf:type ?localProductType .
    ?localProductType owl:sameAs %ProductType% .

    ?product bsbm:productFeature ?localProductFeature1 .
    ?localProductFeature1 owl:sameAs %ProductFeature1% .

    ?product bsbm:productFeature ?localProductFeature2 .
    ?localProductFeature2 owl:sameAs %ProductFeature2% .
    ?product bsbm:productPropertyNumeric1 ?value1 .
    FILTER (?value1 > %x%)
}
ORDER BY ?label
LIMIT 10
```

(b) Q1 FedShop query

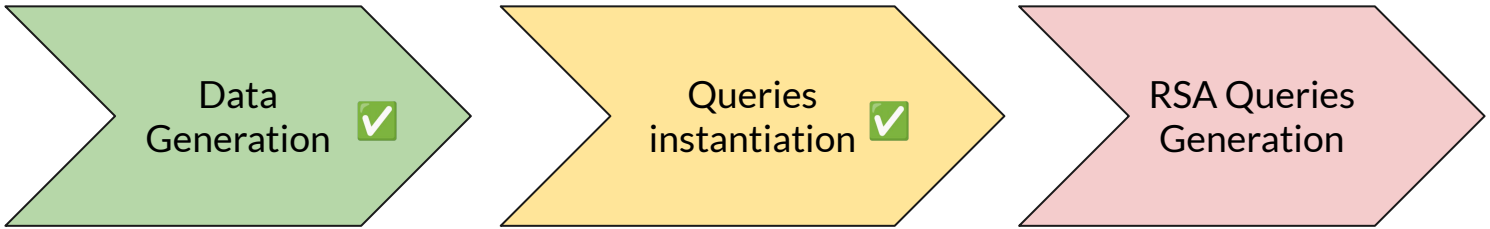
Query 1: Find products for a given set of generic features.

Instantiation of the FedShop Query Templates

- Per template, we instantiate 10 random queries on **F(20) of FedShop200** such that:
  - Instantiated queries return non-empty results.
  - Instantiated queries are unique.
- Total of 120 queries.
- Result size increases monotonically with the size of the federation, F(20) to F(200).

Is the FedShop use-case realizable ?

Is it possible to execute each query of FedShop in a few seconds ?



Reference Source Assignment (RSA) Queries

- Is there a source assignment for each query that can be executed in a few seconds ?
- **Idea:** By relying on FedShop generation rules, we perform query decomposition **manually** and compute minimal source assignment.
  - The output is a SPARQL 1.1 `SERVICE` query that represents a Reference Source Assignment (RSA).
- If RSA queries can be executed in a few seconds, then the FedShop use case is realizable.

RSA Decomposition

- Decompose queries using the **global/local join variables** [1]:
  - A join variable is **global** if it has to be resolved using at least 2 endpoints.
  - A join variable is **local** if it can be resolved on one endpoint.
- As result of the FedShop generation rules:
  - Join variables on **subjects** are **local join variables**.
  - **Objects** of the **sameAs** predicate are **global join variables**.

[1] Abdelaziz, Ibrahim, et al. "Lusail: a system for querying linked data at scale." *VLDB Endowment* 11.4 (2017).

Example of RSA Generation : Step by step

```
SELECT DISTINCT ?product ?localProductLabel
WHERE {
 ?localProduct  rdfs:label  ?localProductLabel  .
 ?localProduct  bsbm:productFeature ?localProdFeature  .
 ?localProduct  bsbm:productPropertyNumeric1 ?simProperty1 .
 ?localProduct  bsbm:productPropertyNumeric2 ?simProperty2 .
 ?localProduct  owl:sameAs ?product .
 ?localProdFeature  owl:sameAs ?prodFeature .
 ?localProductXYZ bsbm:productFeature ?localProdFeatureXYZ .
 ?localProductXYZ bsbm:productPropertyNumeric1 ?origProperty1 .
 ?localProductXYZ bsbm:productPropertyNumeric2 ?origProperty2 .
 ?localProductXYZ owl:sameAs bsbm:Product136030 .
 ?localProdFeatureXYZ owl:sameAs ?prodFeature .
  FILTER (bsbm:Product136030 != ?product)
  FILTER (?simProperty1 < (?origProperty1 + 20) &&
   ?simProperty1 > (?origProperty1 - 20))
  FILTER (?simProperty2 < (?origProperty2 + 70) &&
   ?simProperty2 > (?origProperty2 - 70))}
ORDER BY ?localProductLabel
LIMIT 5
```

Q5: Find products that are similar to a given product.
The consumer has found a product that fulfills her requirements. Then, she wants to find products with similar features.

Example of RSA Generation : Global Join Variables

```
SELECT DISTINCT ?product ?localProductLabel
WHERE {
 ?localProduct  rdfs:label  ?localProductLabel  .
 ?localProduct  bsbm:productFeature ?localProdFeature  .
 ?localProduct  bsbm:productPropertyNumeric1 ?simProperty1 .
 ?localProduct  bsbm:productPropertyNumeric2 ?simProperty2 .
 ?localProduct  owl:sameAs ?product .
 ?localProdFeature  owl:sameAs ?prodFeature .
 ?localProductXYZ bsbm:productFeature ?localProdFeatureXYZ .
 ?localProductXYZ bsbm:productPropertyNumeric1 ?origProperty1 .
 ?localProductXYZ bsbm:productPropertyNumeric2 ?origProperty2 .
 ?localProductXYZ owl:sameAs bsbm:Product136030 .
 ?localProdFeatureXYZ owl:sameAs ?prodFeature .
  FILTER (bsbm:Product136030 != ?product)
  FILTER (?simProperty1 < (?origProperty1 + 20) &&
   ?simProperty1 > (?origProperty1 - 20))
  FILTER (?simProperty2 < (?origProperty2 + 70) &&
   ?simProperty2 > (?origProperty2 - 70))}
ORDER BY ?localProductLabel
LIMIT 5
```

Q5: Find products that are similar to a given product.
The consumer has found a product that fulfills her requirements. Then, she wants to find products with similar features.

Example of RSA Generation : Decomposition

```
SELECT DISTINCT ?product ?localProductLabel
WHERE {
 ?localProduct  rdfs:label  ?localProductLabel  .
 ?localProduct  bsbm:productFeature ?localProdFeature  .
 ?localProduct  bsbm:productPropertyNumeric1 ?simProperty1 .
 ?localProduct  bsbm:productPropertyNumeric2 ?simProperty2 .
 ?localProduct  owl:sameAs ?product .
 ?localProdFeature  owl:sameAs ?prodFeature .
 ?localProductXYZ bsbm:productFeature ?localProdFeatureXYZ .
 ?localProductXYZ bsbm:productPropertyNumeric1 ?origProperty1 .
 ?localProductXYZ bsbm:productPropertyNumeric2 ?origProperty2 .
 ?localProductXYZ owl:sameAs bsbm:Product136030 .
 ?localProdFeatureXYZ owl:sameAs ?prodFeature .
  FILTER (bsbm:Product136030 != ?product)
  FILTER (?simProperty1 < (?origProperty1 + 20) &&
   ?simProperty1 > (?origProperty1 − 20))
  FILTER (?simProperty2 < (?origProperty2 + 70) &&
   ?simProperty2 > (?origProperty2 − 70))}
ORDER BY ?localProductLabel
LIMIT 5
```

RSA Generation

```
SERVICE ?bgp1 {
 ?localProductXYZ owl:sameAs bsbm:Product136030 .
 ?localProductXYZ bsbm:productFeature ?localProdFeatureXYZ .
 ?localProdFeatureXYZ owl:sameAs ?prodFeature .
 ?localProductXYZ bsbm:productPropertyNumeric1 ?origProperty1 .
 ?localProductXYZ bsbm:productPropertyNumeric2 ?origProperty2} .
SERVICE ?bgp2 {
 ?localProduct  owl:sameAs ?product  .
  FILTER (bsbm:Product136030 != ?product)
 ?localProduct  rdfs:label  ?localProductLabel  .
 ?localProduct  bsbm:productFeature ?localProdFeature  .
 ?localProdFeature  owl:sameAs ?prodFeature .
 ?localProduct  bsbm:productPropertyNumeric1 ?simProperty1 .
 ?localProduct  bsbm:productPropertyNumeric2 ?simProperty2} .
FILTER(?simProperty1 < (?origProperty1 + 20) &&
  ?simProperty1 > (?origProperty1 − 20))
FILTER(?simProperty2 < (?origProperty2 + 70) &&
  ?simProperty2 > (?origProperty2 − 70))}
ORDER BY ?product ?localProductLabel
LIMIT 5
```

Q5: Find products that are similar to a given product.
The consumer has found a product that fulfills her requirements. Then, she wants to find products with similar features.

Example of RSA Generation : Source Selection

```
SELECT DISTINCT ?product ?localProductLabel
WHERE {
 ?localProduct  rdfs:label  ?localProductLabel  .
 ?localProduct  bsbm:productFeature ?localProdFeature  .
 ?localProduct  bsbm:productPropertyNumeric1 ?simProperty1 .
 ?localProduct  bsbm:productPropertyNumeric2 ?simProperty2 .
 ?localProduct  owl:sameAs ?product .
 ?localProdFeature  owl:sameAs ?prodFeature .
 ?localProductXYZ bsbm:productFeature ?localProdFeatureXYZ .
 ?localProductXYZ bsbm:productPropertyNumeric1 ?origProperty1 .
 ?localProductXYZ bsbm:productPropertyNumeric2 ?origProperty2 .
 ?localProductXYZ owl:sameAs bsbm:Product136030 .
 ?localProdFeatureXYZ owl:sameAs ?prodFeature .
  FILTER (bsbm:Product136030 != ?product)
  FILTER (?simProperty1 < (?origProperty1 + 20) &&
   ?simProperty1 > (?origProperty1 - 20))
  FILTER (?simProperty2 < (?origProperty2 + 70) &&
   ?simProperty2 > (?origProperty2 - 70))}
ORDER BY ?localProductLabel
LIMIT 5
```

RSA Generation

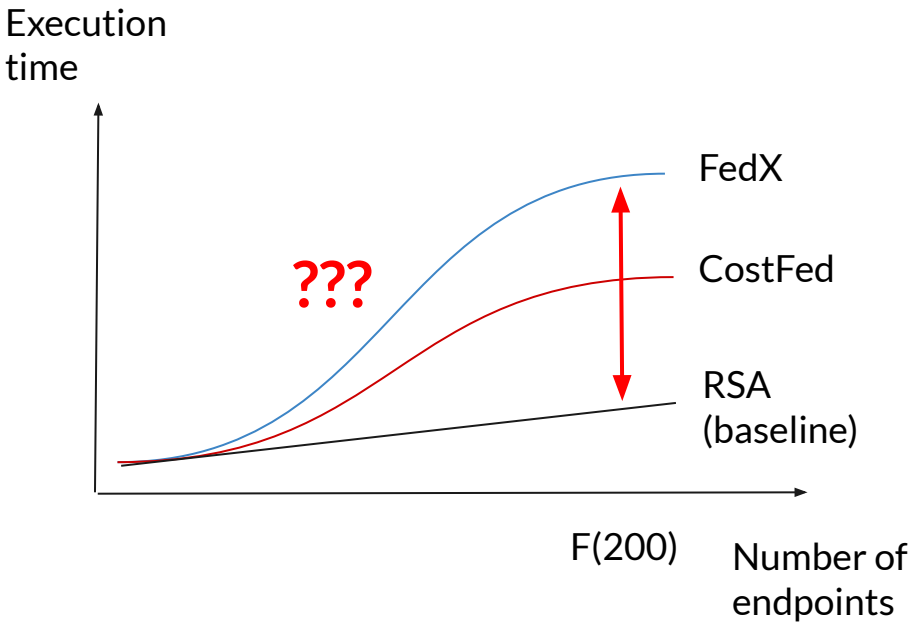
```
SELECT DISTINCT ?product ?localProductLabel WHERE {
 VALUES ( ?bgp1 ?bgp2 ) {
 ( <http://www.vendor1.fr/> <http://www.vendor1.fr/> )
 ( <http://www.vendor1.fr/> <http://www.vendor2.fr/> )
 # ... ) }
SERVICE ?bgp1 {
 ?localProductXYZ owl:sameAs bsbm:Product136030 .
 ?localProductXYZ bsbm:productFeature ?localProdFeatureXYZ .
 ?localProdFeatureXYZ owl:sameAs ?prodFeature .
 ?localProductXYZ bsbm:productPropertyNumeric1 ?origProperty1 .
 ?localProductXYZ bsbm:productPropertyNumeric2 ?origProperty2} .
SERVICE ?bgp2 {
 ?localProduct owl:sameAs ?product  .
  FILTER (bsbm:Product136030 != ?product)
 ?localProduct  rdfs:label  ?localProductLabel  .
 ?localProduct  bsbm:productFeature ?localProdFeature  .
 ?localProdFeature  owl:sameAs ?prodFeature .
 ?localProduct  bsbm:productPropertyNumeric1 ?simProperty1 .
 ?localProduct  bsbm:productPropertyNumeric2 ?simProperty2} .
FILTER(?simProperty1 < (?origProperty1 + 20) &&
  ?simProperty1 > (?origProperty1 - 20))
FILTER(?simProperty2 < (?origProperty2 + 70) &&
  ?simProperty2 > (?origProperty2 - 70))}
ORDER BY ?product ?localProductLabel
LIMIT 5
```

Q5: Find products that are similar to a given product.
The consumer has found a product that fulfills her requirements. Then, she wants to find products with similar features.

FedShop200 : Experimental Study

Experimental Study

1. Is it possible to execute RSA queries in a few seconds ? Is FedShop Realizable ?
2. Can existing engines execute the FedShop use case in a few seconds ?
3. How do federation engines perform compared to RSA ?



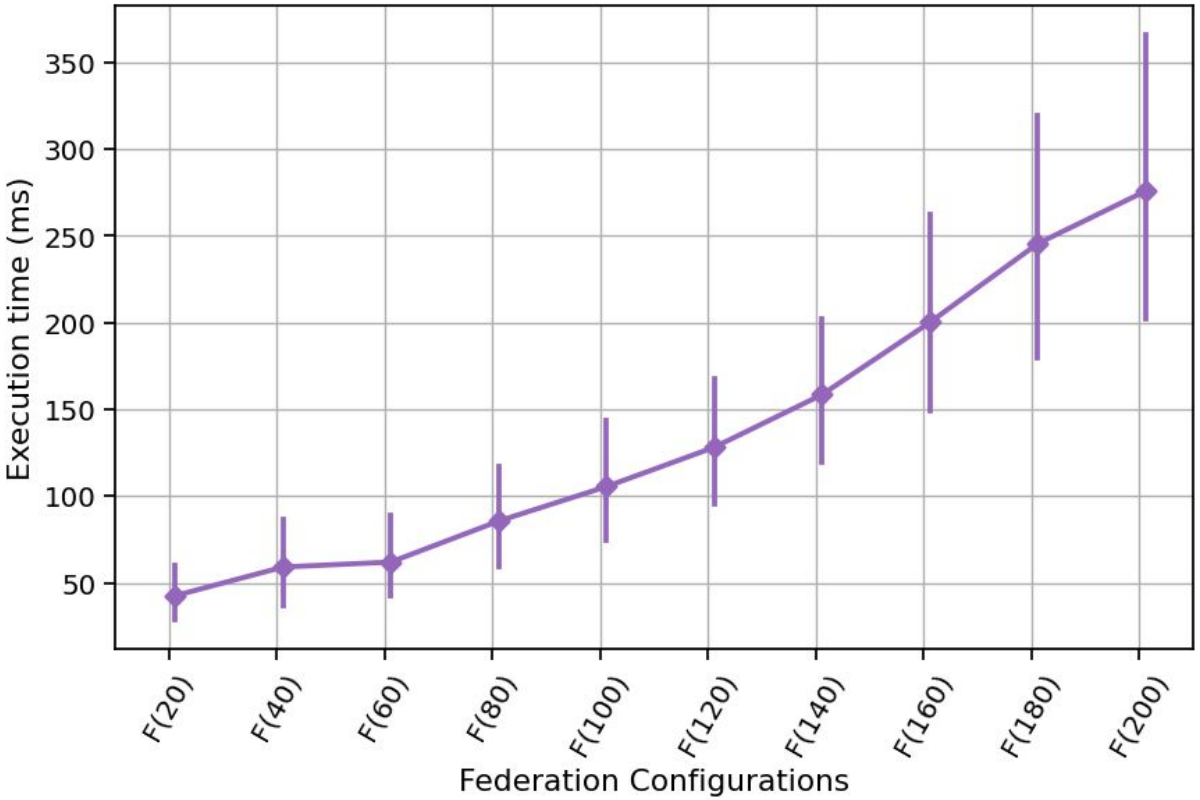
Experimental setup

- All FedShop200 data is stored in a single Virtuoso Server with 20 to 200 virtual SPARQL endpoints.
- 120 queries executed over each of the 10 federations (FedShop20 to FedShop200), using:
  - **FedX , CostFed, ANAPSID, and SemaGrow**.
- We generated all 10x120 = **1200 RSA queries** and executed them with Apache Jena.

Metrics

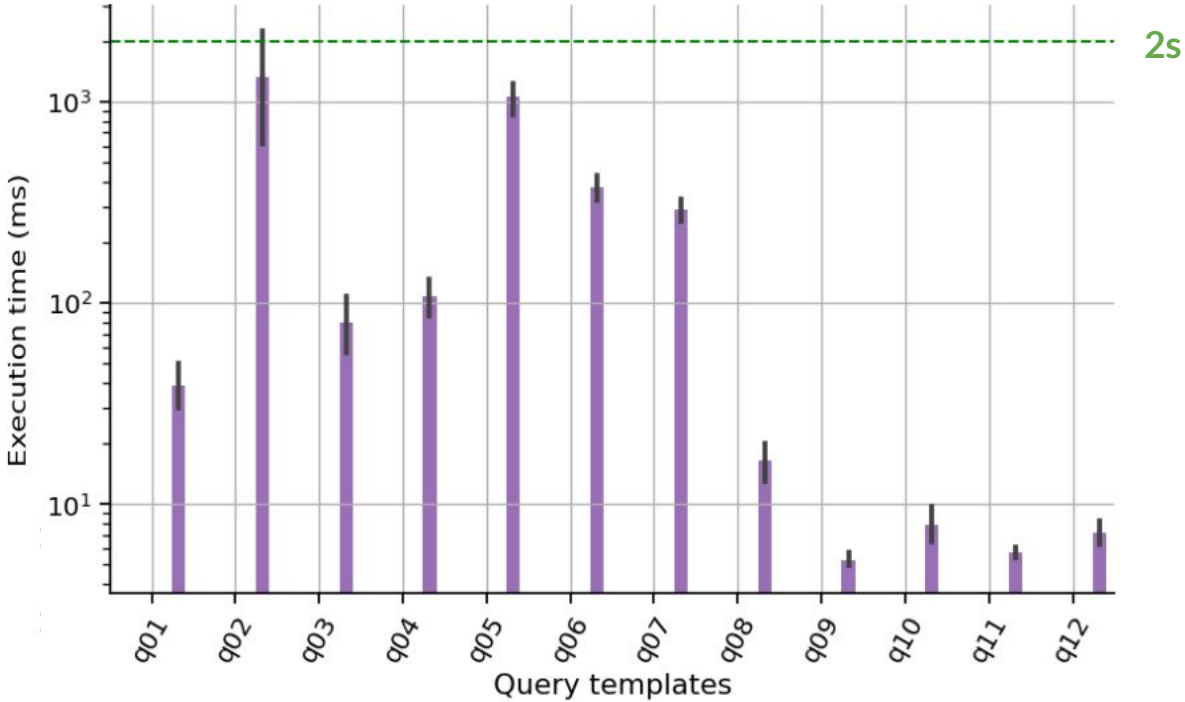
- **Execution Time** : the total time spent by the SPARQL federation engine to produce the query results.
  - The reported times are averages of 4 runs.
- **Timeout is set to 120 seconds** per query to match interactive use-case.
  - If a timeout or an error occurs, the execution time is assigned a value of 120 seconds.

Is it possible to execute RSA queries in a few seconds?



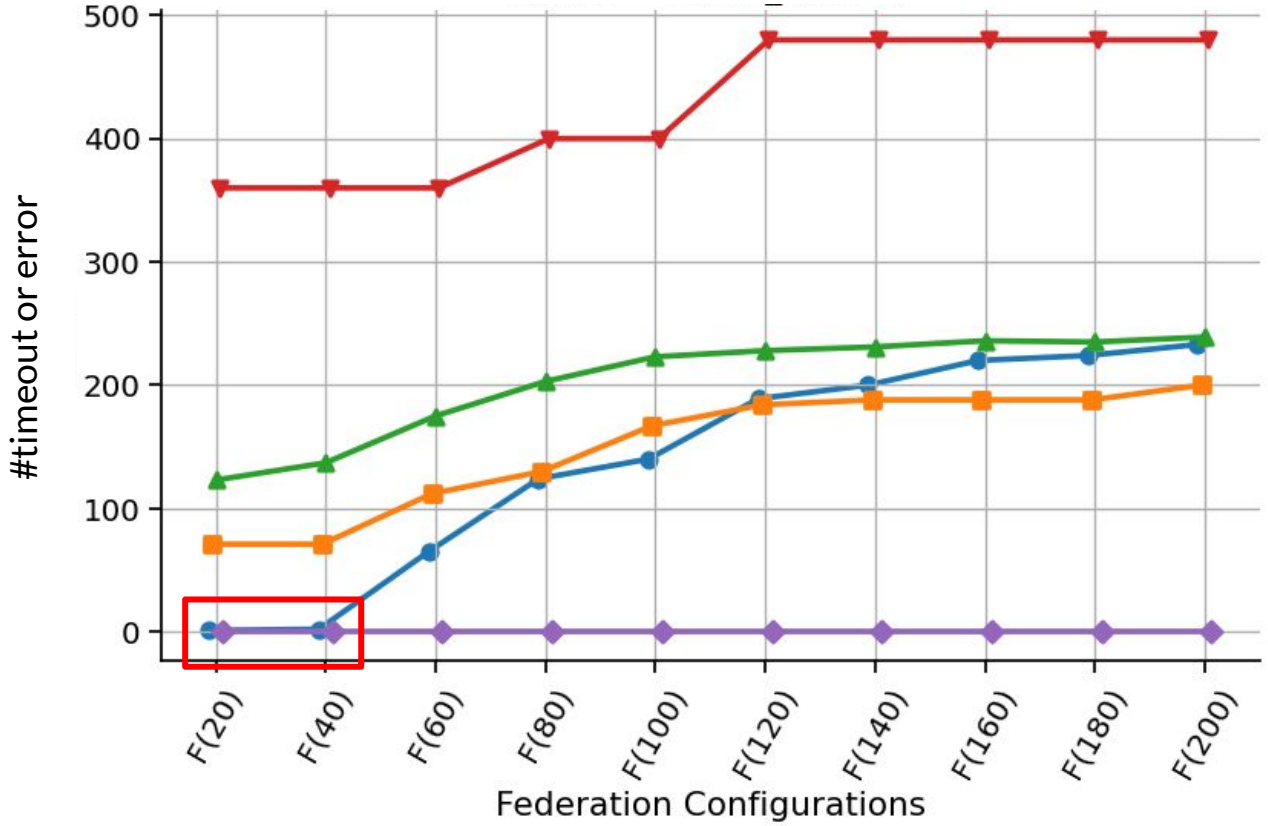
Average execution time per query across federations < 400ms.

Is it possible to execute *every* RSA query in a few seconds ?



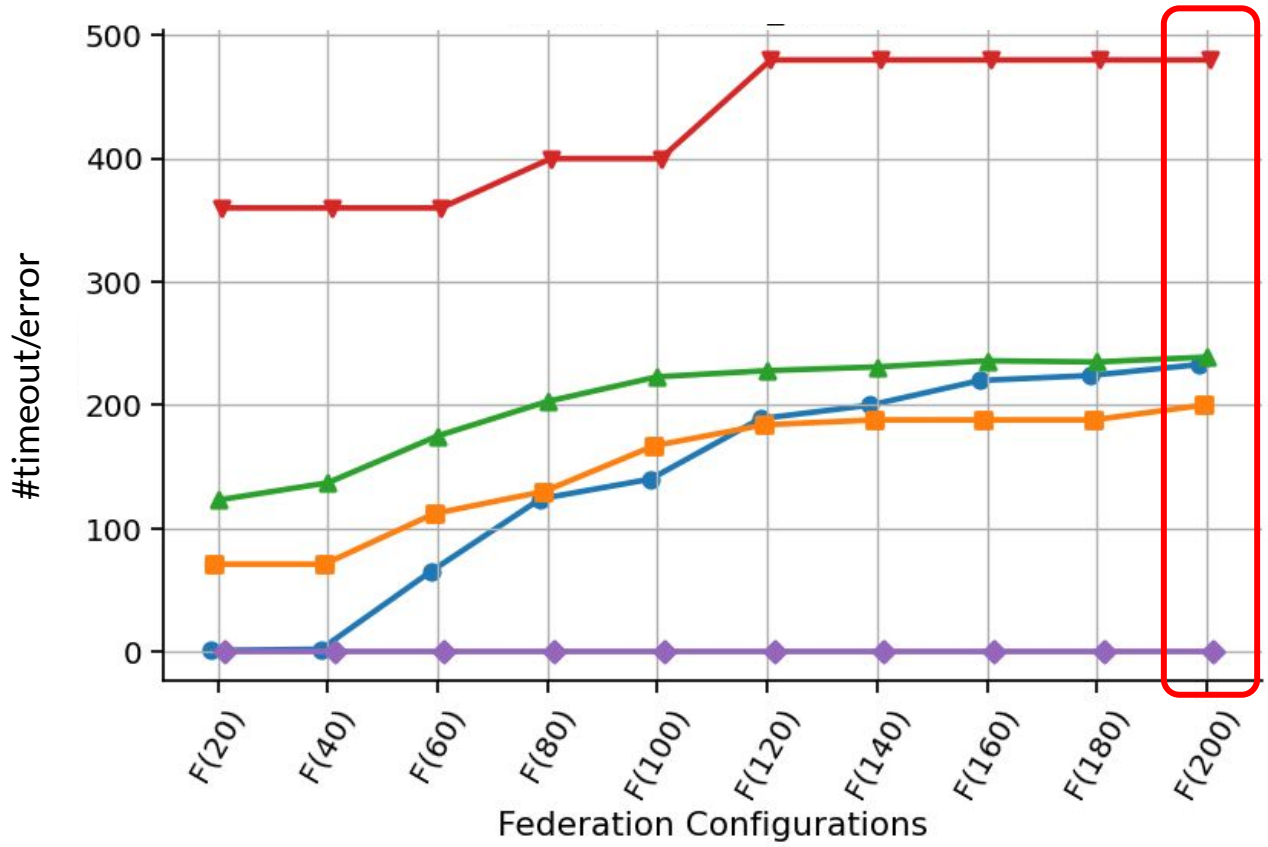
F(200)

Can existing engines execute the FedShop use case in a few seconds ?



Number of queries that fail with timeout or error per federation (max: 120x4=480).

Can existing engines execute the FedShop use case in a few seconds ?

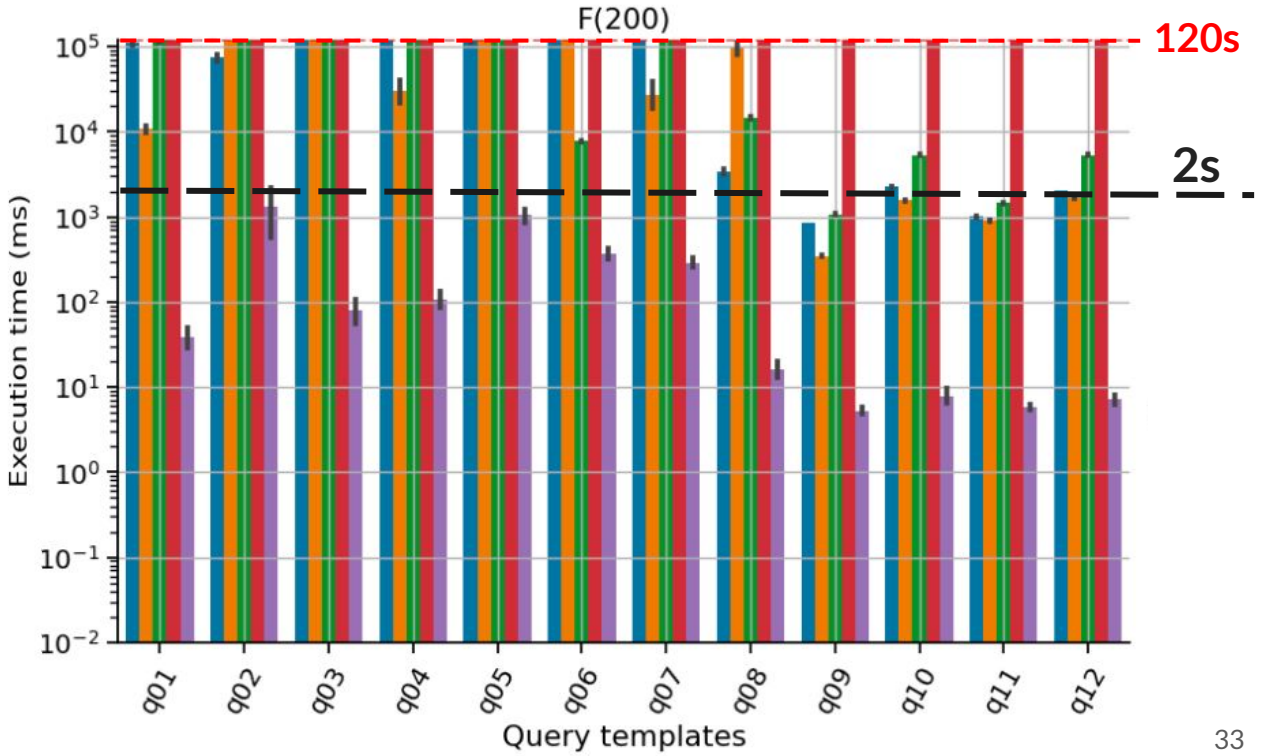


Number of queries that fail with timeout or error per federation (max: 120x4=480).

- fedx
- costfed
- semagrow
- anapsid
- rsa

Which query template fails ? What is the gap Engine/RSA ?

There is a large gap of performance between RSA and current engines **for all query templates**



Conclusion

- FedShop is the first benchmark designed for studying the scalability of SPARQL federation engines.
- **The good news :**
  - RSA shows that FedShop use case is realizable.
- **The bad news :**
  - Existing engines cannot run the Fedshop use case.
- **The message:**
  - Further research on federation engines needed to fill the gap.

Perspectives

- Evaluate FedShop on more SPARQL federation engines.
- Include more metrics.
- Produce new configurations for FedShop different from FedShop200.
- Better automation of RSA queries generation.

FedShop

A Benchmark for Testing the Scalability of SPARQL Federation Engines

https://github.com/GDD-Nantes/FedShop

Minh-Hoang Dang , Julien Aimonier-Davat , Pascal Molli , **Olaf Hartig** , Hala Skaf-Molli and Yotlan Le Crom

Nantes Université

LINKÖPING UNIVERSITY

Existing benchmarks

- FedBench [1]:
  - 10 datasets (4 Life Science + 6 Cross domain) + 14 real queries
  - 16 synthetic datasets + 11 real queries from SP2Bench.
  - ~178M triples
- LargeRDFBench [2] is an extension of FedBench:
  - 13 datasets and 40 queries
  - ~1 billion triples

Increasing the triple/query number does not reveal how query federation engines **behave** when **the number of source increases**.

[1] Schmidt, Michael, et al. "Fedbench: A benchmark suite for federated semantic data query processing." The Semantic Web–ISWC 2011: 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011,
[2] Saleem, Muhammad, Ali Hasnain, and Axel-Cyrille Ngonga Ngomo. "LargeRDFBench: A billion triples benchmark for SPARQL endpoint federation." Journal of Web Semantics 48 (2018): 85-125.

Partitioning existing datasets to create more sources

- In the experimental study of Lusail [1], DARQ [2], LHD [3], LILAC [4], different partitioning techniques are applied to non-federated benchmark (SP2Bench, LUBM):
  - horizontal, vertical, and hybrid partitioning with or without replication.
- Partitioning based on classes are limited by the number of classes [Fedbench, DARQ]
  - Cannot scale
- Simple horizontal partitioning techniques:
  - As pointed out in Splendid, it is "**difficult to create partitions which resemble the characteristics of real world datasets"**
- Lusail partitioned LUBM by universities to create a setup up to 256 endpoints.
  - LUBM queries are designed for evaluating performance of reasoning, not with a real use case in mind.

[1] Abdelaziz, Ibrahim, et al. "Lusail: a system for querying linked data at scale." Proceedings of the VLDB Endowment 11.4 (2017): 485-498.
[2] Quilitz, Bastian, and Ulf Leser. "Querying distributed RDF data sources with SPARQL." Lecture Notes in Computer Science 5021 (2008): 524.
[3] Wang, Xin, Thanassis Tiropanis, and Hugh C. Davis. "Lhd: Optimising linked data query processing using parallelisation." (2013).
[4] Montoya, Gabriela, et al. "Decomposing federated queries in presence of replicated fragments." Journal of Web Semantics 42 (2017): 1-18.

Selecting Sources from Provenance Queries

```
SELECT DISTINCT ?product ?localProductLabel WHERE {
 VALUES ( ?bgp1 ?bgp2 ) {
 ( <http://www.vendor1.fr/> <http://www.vendor1.fr/> )
 ( <http://www.vendor1.fr/> <http://www.vendor2.fr/> )
 # ... ) }
SERVICE ?bgp1 {
 ?localProductXYZ owl:sameAs bsbm:Product136030 .
 ?localProductXYZ bsbm:productFeature ?localProdFeatureXYZ .
 ?localProdFeatureXYZ owl:sameAs ?prodFeature .
 ?localProductXYZ bsbm:productPropertyNumeric1 ?origProperty1 .
 ?localProductXYZ bsbm:productPropertyNumeric2 ?origProperty2} .
SERVICE ?bgp2 {
 ?localProduct owl:sameAs ?product .
  FILTER (bsbm:Product136030 != ?product)
 ?localProduct rdfs:label ?localProductLabel .
 ?localProduct bsbm:productFeature ?localProdFeature .
 ?localProdFeature owl:sameAs ?prodFeature .
 ?localProduct bsbm:productPropertyNumeric1 ?simProperty1 .
 ?localProduct bsbm:productPropertyNumeric2 ?simProperty2} .
FILTER(?simProperty1 < (?origProperty1 + 20) &&
  ?simProperty1 > (?origProperty1 − 20))
FILTER(?simProperty2 < (?origProperty2 + 70) &&
  ?simProperty2 > (?origProperty2 − 70))}
ORDER BY ?product ?localProductLabel
LIMIT 5
```

(a) Cross-Domain service query $Q5$

2

1

```
SELECT DISTINCT ?bgp1 ?bgp2 WHERE {
graph ?bgp1 {
 ?localProductXYZ owl:sameAs bsbm:Product136030 .
 ?localProductXYZ bsbm:productFeature ?localProdFeatureXYZ .
 ?localProdFeatureXYZ owl:sameAs ?prodFeature .
 ?localProductXYZ bsbm:productPropertyNumeric1 ?origProperty1 .
 ?localProductXYZ bsbm:productPropertyNumeric2 ?origProperty2} .
graph ?bgp2 {
 ?localProduct owl:sameAs ?product .
  FILTER (bsbm:Product136030 != ?product)
 ?localProduct rdfs:label ?localProductLabel .
 ?localProduct bsbm:productFeature ?localProdFeature .
 ?localProdFeature owl:sameAs ?prodFeature .
 ?localProduct bsbm:productPropertyNumeric1 ?simProperty1 .
 ?localProduct bsbm:productPropertyNumeric2 ?simProperty2} .
FILTER(?simProperty1 < (?origProperty1 + 20) &&
  ?simProperty1 > (?origProperty1 − 20))
FILTER(?simProperty2 < (?origProperty2 + 70) &&
  ?simProperty2 > (?origProperty2 − 70))}
ORDER BY ?product ?localProductLabel
LIMIT 5
```

(b) Provenance query for $Q5_{prov}$

1: SERVICE query to prov. query
2: Update VALUES with Result of evaluation of prov. query on union of graph on federation.

RSA Decomposition categories

- **Single Domain** (Q9, Q11, Q12) **:**
  - The whole query is executed on only 1 endpoint.
- **Multiple Domain** (Q1, Q2, Q3, Q4, Q6, Q8, Q10):
  - The whole query is executed on different endpoints
- **Cross Domain** (Q5,Q7) **:**
  - The query is decomposed on several subqueries.

BSBM Queries:

Query 1: Find products for a given set of generic features.

Query 2: Retrieve basic information about a specific product for display purposes.

Query 3: Find products for a given more specific set of features.

Query 4: Find products matching two different sets of features.

Query 5: Find products that are similar to a given product.

Query 6: Find products having a label that contains a specific string.

Query 7: Retrieve in-depth information about a specific product including offers and reviews.

Query 8: Give me recent German reviews for a specific product.

Query 9: Get information about a reviewer.

Query 10: Get offers for a given product which fulfill specific requirements.

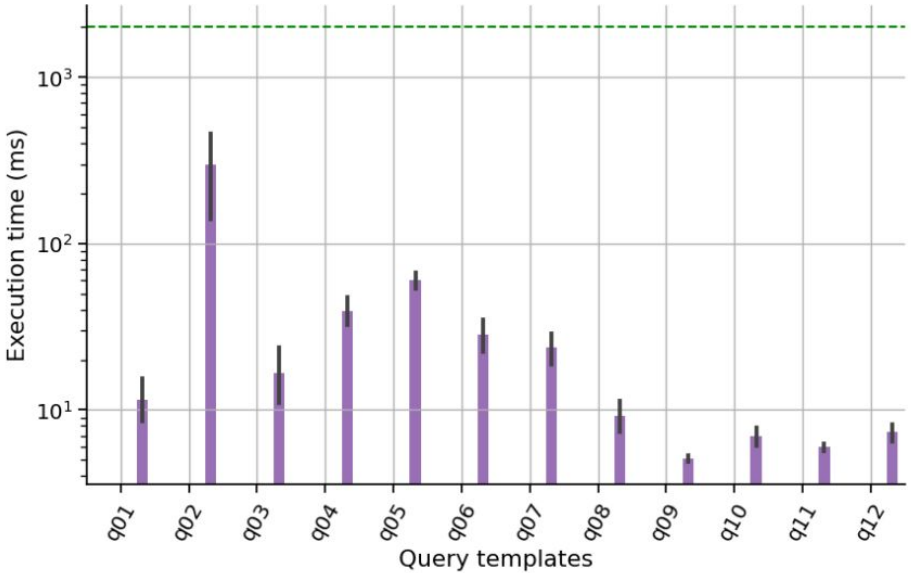
Query 11: Get all information about an offer.

Query 12: Export information about an offer into another schemata.

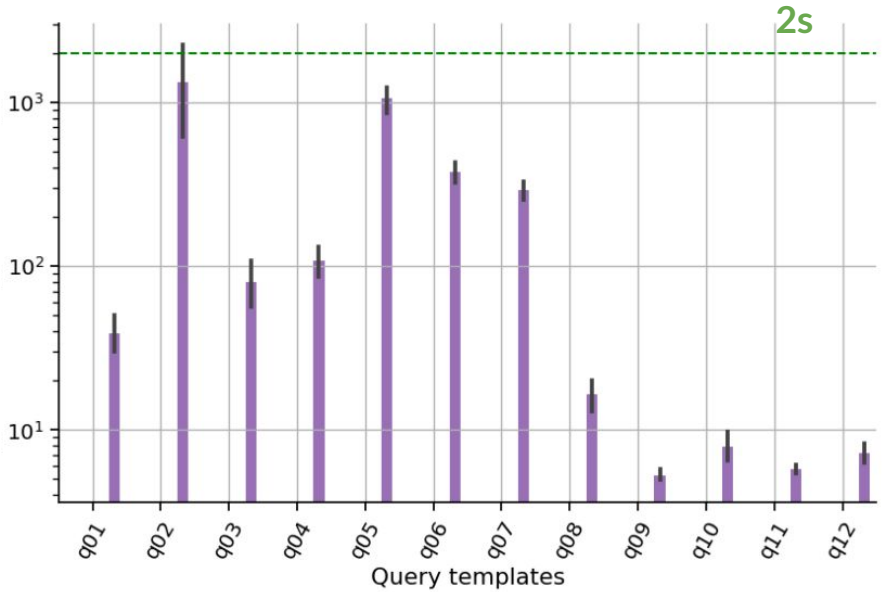
Example of RSA queries

```
SELECT DISTINCT ?product ?localProductLabel WHERE {
  VALUES (?bgp1 ?bgp2) {
    (
      <http://virtuoso:5555/sparql/?default-graph-uri=http://www.ratingsite6.fr/>
      <http://virtuoso:5555/sparql/?default-graph-uri=http://www.vendor3.fr/>
    )
    (
      <http://virtuoso:5555/sparql/?default-graph-uri=http://www.ratingsite0.fr/>
      <http://virtuoso:5555/sparql/?default-graph-uri=http://www.vendor3.fr/>
    )
  }
  SERVICE ?bgp2 {
    ?localProductXYZ owl:sameAs bsbm:Product136030;
      bsbm:productFeature ?localProdFeatureXYZ.
    ?localProdFeatureXYZ owl:sameAs ?prodFeature.
    ?localProductXYZ bsbm:productPropertyNumeric1 ?origProperty1;
      bsbm:productPropertyNumeric2 ?origProperty2.
  }
  SERVICE ?bgp1 {
    ?localProduct owl:sameAs ?product.
    FILTER(bsbm:Product136030 != ?product)
    ?localProduct rdfs:label ?localProductLabel;
      bsbm:productFeature ?localProdFeature.
    ?localProdFeature owl:sameAs ?prodFeature.
    ?localProduct bsbm:productPropertyNumeric1 ?simProperty1;
      bsbm:productPropertyNumeric2 ?simProperty2.
  }
  FILTER((?simProperty1 < (?origProperty1 + 20 )) && (?simProperty1 > (?origProperty1 - 20 )))
  FILTER((?simProperty2 < (?origProperty2 + 70 )) && (?simProperty2 > (?origProperty2 - 70 )))
}
ORDER BY (?product) (?localProductLabel)
LIMIT 5
```

Is it possible to execute RSA queries in a few seconds ?

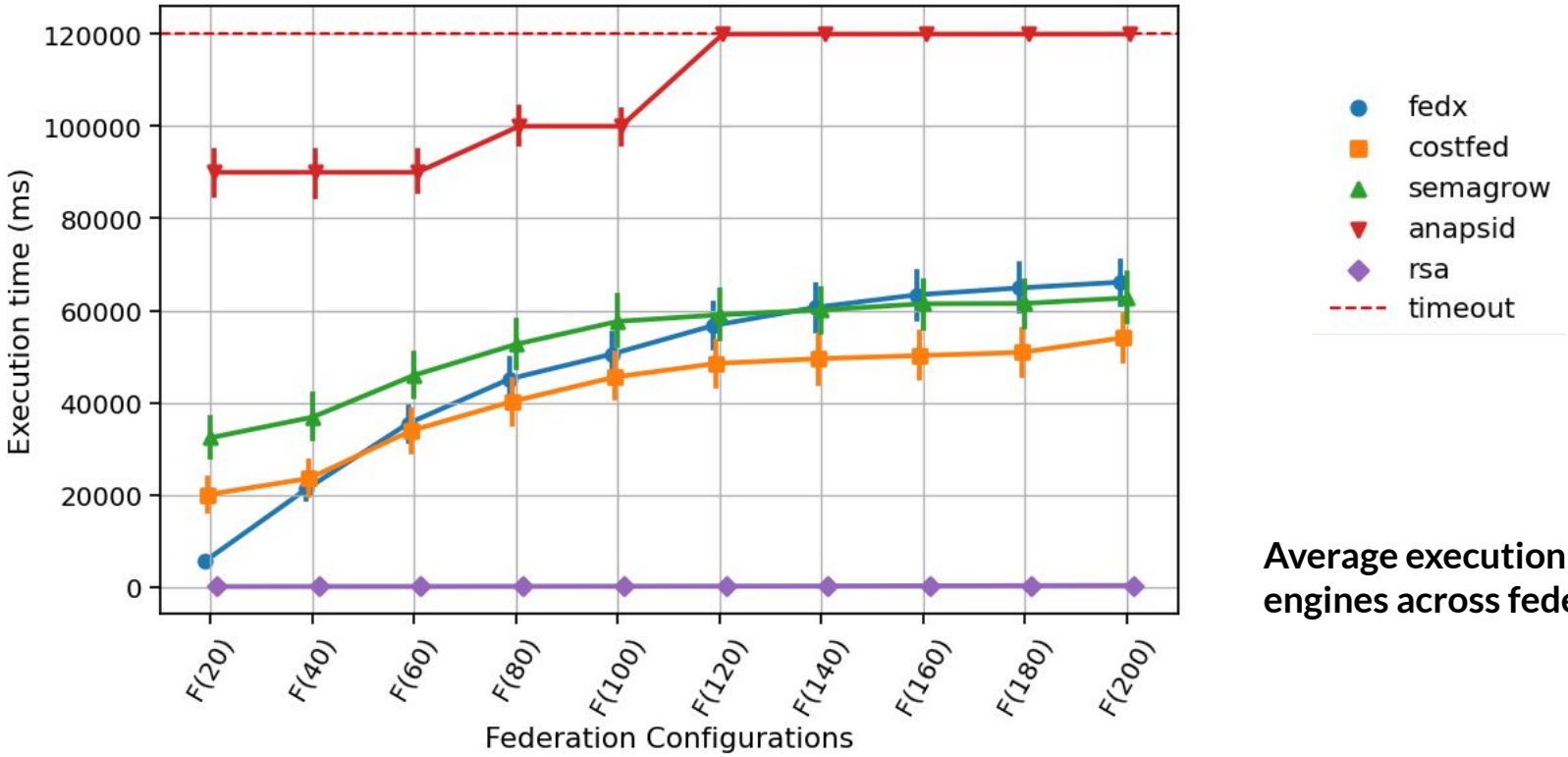


F(20)



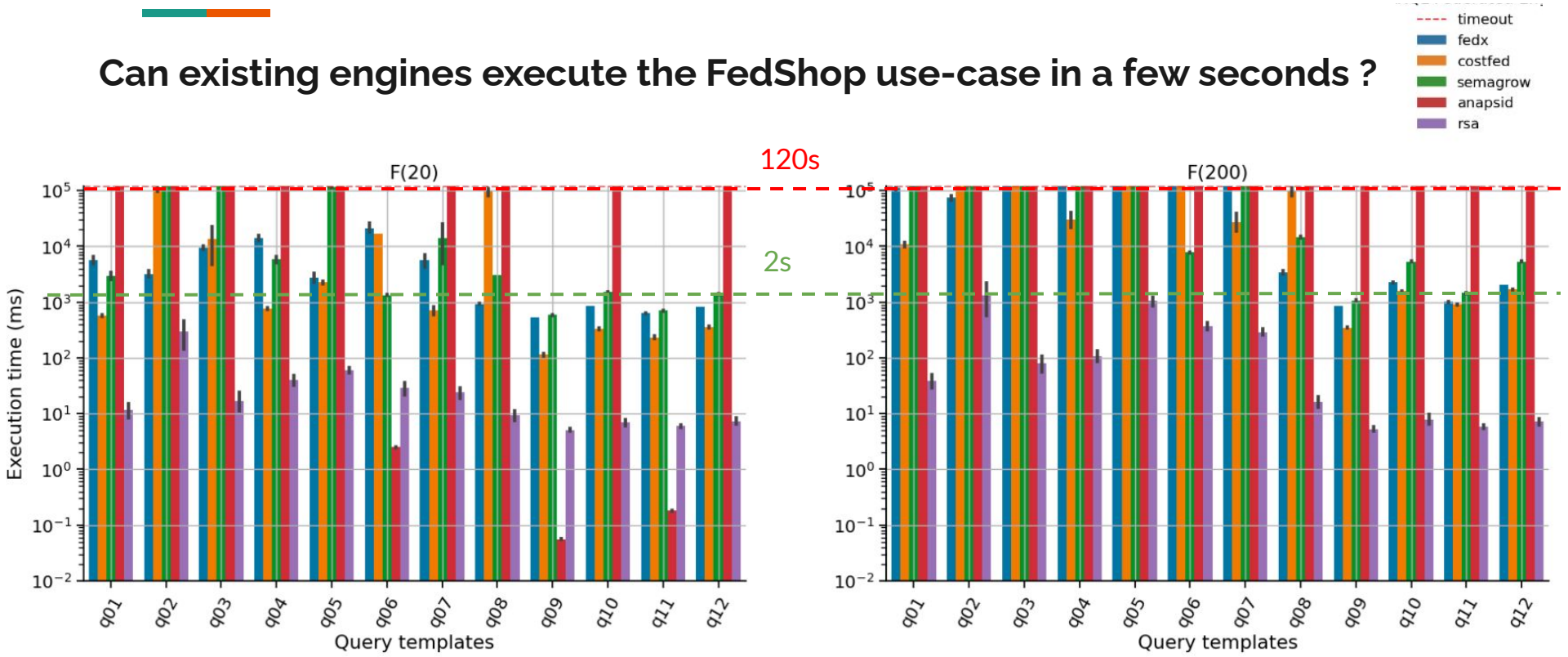
F(200)

The bad news: Existing engines cannot run FedShop200



Average execution time of engines across federation.

Can existing engines execute the FedShop use-case in a few seconds ?



There is a large gap of performance between RSA and current engines **for all query templates when federation size grows**!