Federated Query Processing over RDF Graphs

Maria-Esther Vidal
Fraunhofer IAIS, Germany
Universidad Simón Bolívar, Venezuela
Motivating Example

Vessels that have arrived at port of Funchal on 2017-06-21
Motivating Example

Interrelated databases distributed over a computer network. Components are designed and tuned to work together.
Dimensions of Distributed Database Systems

**Distribution**: Physical distribution of the data over multiple sites.

- **Vessels**: Canada
- **Ports**: World-Port Index USA
- **Vessel Observations**: Greece
Dimensions of Distributed Database Systems

**Autonomy**: Distribution of the control; degree to which individual systems can operate independently.

- **Vessels**: Evergreen Marine
- **Ports**: World-Port Index
- **Vessel Observations**: Maritime Traffic
Dimensions of Distributed Database Systems

Heterogeneity: Different forms ranging from hardware, differences in network protocols, data models, query languages.
Dimensions of Distributed Database Systems [1]

Agenda

1) Distributed Data Management Systems
   a) Client-Server Systems
   b) Peer-to-Peer Systems
   c) Federated Data Management Systems
2) Challenges in Federated Data Management
3) Distributed SPARQL Systems
4) Conclusions
Distributed Data Management Systems
Client-Server Systems:

- **Clients** run user applications and interfaces.
- **Servers** run data management tasks, e.g., query processing and storage.
Client-Server Systems

Client

Queries

Server Engine

Query Answers

Vessels

Ports

Vessel Observations
Peer-to-Peer Systems:

- **Massive** distribution
- May used **different** data models.
- Each system **manages** a different dataset.
- Peers can **communicate**.
Peer-to-Peer Systems

Peers communicate.
Federated Data Management (FDM) Systems

FDM Systems:
- **Fully** autonomous and have no concept of cooperation.
- May use **different** data models.
- Each system manages a **different** database.
Federated Data Management (FDM) Systems

- Vessels
  - Evergreen Marine
- Vessel Observations
  - Maritime Traffic
- Ports
  - World-Port Index

- CSV
- RDF
- JSON

Federated Engine
Challenges in Federated Data Management
Challenges for Query Processing in FDM

Given a query Q in a formal language, i.e., SPARQL

- **Identify** the relevant data sources for Q (**Source Selection**)
- **Decompose** Q into subqueries on relevant data sources (**Query Decomposition**)
- **Plan** evaluation of subqueries against relevant data sources (**Query Planning**)
- **Merge** data collected from relevant data sources (**Query Execution**)

17
Challenges for Query Processing in FDM

Vessels that have arrived at port of Funchal on 2017-06-21

```
SELECT ?mmsi ?name ?flag
WHERE {
  ?vessel bdo:name ?name.
  ?observ bdo:lat ?lat.
  ?observ bdo:date "2017-06-21" .
  ?port bdo:lat ?lat.
  ?port bdo:name "Funchal" }
```
Example

```
SELECT ?mmsi ?name ?flag
WHERE {
  ?vessel bdo:name ?name.
  ?observ bdo:lat ?lat.
  ?observ bdo:date "2017-06-21" .
  ?port bdo:lat ?lat.
  ?port bdo:name "Funchal" }
```

```
SELECT ?mmsi ?name ?flag
WHERE {
  SERVICE C1 {
    ?vessel bdo:name ?name.
    ?vessel bdo:flag ?flag.}
  SERVICE C2 {
    ?observ bdo:lat ?lat.
    ?observ bdo:date "2017-06-21" .
    ?observ bdo:mmsi ?mmsi. }
  SERVICE C3 {
    ?port bdo:lat ?lat.
    ?port bdo:name "Funchal" }}
```
Challenges for Query Processing in FDM

Vessels that have arrived at port of **Funchal** on **2017-06-21**

```
SELECT ?mmsi ?name ?flag
WHERE {
  ?vessel bdo:name ?name.
  ?observ bdo:lat ?lat.
  ?observ bdo:date "2017-06-21" .
  ?port bdo:lat ?lat.
  ?port bdo:name "Funchal" }
```
Challenges for Query Processing in FDM

Vessels that have arrived at port of Funchal on 2017-06-21

```
SELECT ?mmsi ?name ?flag
WHERE {
  ?vessel bdo:name ?name.
  ?observ bdo:lat ?lat.
  ?observ bdo:date "2017-06-21" .
  ?port  bdo:lat ?lat.
  ?port  bdo:name "Funchal" }
```
Challenges for Query Processing in FDM

Given a query Q in a formal language, i.e., SPARQL

- **Identify** the **relevant** data sources for Q (**Source Selection**)
- **Decompose** Q into subqueries on **relevant** data sources (**Query Decomposition**)
- **Plan** evaluation of **subqueries** against **relevant** data sources (**Query Planning**)
- **Merge** data collected from **relevant** data sources (**Query Execution**)

---

22
Challenges for Query Processing in FDM

Vessels that have arrived at port of Funchal on 2017-06-21

SELECT ?mmsi ?name ?flag
WHERE {
  ?vessel bdo:name ?name.
  ?observ bdo:lat ?lat.
  ?observ bdo:date "2017-06-21".
  ?port bdo:lat ?lat.
  ?port bdo:name “Funchal”
}
Challenges for Query Processing in FDM

Given a query Q in a formal language, i.e., SPARQL

- **Identify** the relevant data sources for Q (**Source Selection**)
- **Decompose** Q into subqueries on relevant data sources (**Query Decomposition**)
- **Plan** evaluation of subqueries against relevant data sources (**Query Planning**)
- **Merge** data collected from relevant data sources (**Query Execution**)

---

24
Challenges for Query Processing in FDM

?vessel bdo:name ?name.

?observ bdo:lat ?lat.
?observ bdo:date "2017-06-21".

?port bdo:lat ?lat.
?port bdo:name “Funchal”

Vessels

Observations

Ports
Challenges for Query Processing in FDM

Given a query Q in a formal language, i.e., SPARQL

- **Identify** the relevant data sources for Q (Source Selection)
- **Decompose** Q into subqueries on relevant data sources (Query Decomposition)
- **Plan** evaluation of subqueries against relevant data sources (Query Planning)
- **Merge** data collected from relevant data sources (Query Execution)
Challenges for Query Processing in FDM

SELECT name, mmsi, flag FROM VESSELS

SELECT lat, long, mmsi FROM OBSERVATIONS WHERE date="2017-06-21"

SELECT ?lat, ?long, ?mmsi WHERE {
  ?port bdo:lat ?lat.
  ?port bdo:name "Funchal"
}
Challenges for Query Processing in FDM

Operators **consume** tuples from **children** and **pass** output tuples to their **parents**.

- `?vessel bdo:name ?name.`

- `?observ bdo:date "2017-06-21" .`

- `?port bdo:name “Funchal”`
Complexity of FDM Query Processing Tasks

**NP-Hard** Finding Optimal Plan for Distributed Data Sources [1]

Evaluation of SPARQL Queries is **NP-Complete** [2]

**NP-Hard** Finding Optimal Query Decomposition over Distributed RDF Data Sources [3]


Traditional Federated Query Processing

Query

Source Selection and Query Decomposition

Query Optimizer

Query Execution Engine

Catalog

Statistics Generator

Traditional Optimize-Then-Execute Architecture
Ideal Federated Management Systems

- Systems able to change their behavior by learning behavior of data providers.
- Receive information from the environment.
- Use up-to-date information to change their behavior.
- Keep iterating over time to adapt their behavior based on the environment conditions.
Adaptive Query Processing [4]

Adaptivity is needed:
- Misestimated or missing statistics.
- Unexpected correlations.
- Unpredictable costs.
- Dynamically changing data, workload, and source availability.
- Changes at rates at which tuples arrive from sources
  - Initial Delays.
  - Slow Delivery.
  - Bursty Arrivals.
Adaptive Federated Query Processing

Re-optimize the original plan on-the-fly according to the source conditions.
Levels of Adaptation

Adaptation

Adaptation Levels

Source Selection

Query Execution
Adaptation Levels

**Source Selection**: searching strategies to select the sources for answering a query according to the real-time source conditions:

- **Schema** changes
- **Source** availability
- **Data distribution** changes
Adaptation Levels

Query Execution: strategies to adapt query processing to the environment conditions. Adaptation can be at different granularities:

- **Fine-grained** adaptation of small processes, e.g.,
  - Per tuple
- **Coarse-grained** is attempted for large processes, e.g.,
  - Several queries are executed under certain assumptions about the conditions of the environment
Adaptive Query Processing Strategies

Adaptive Join Processing (Intra-Operator):
● Adaptivity performed at tuple level.
● Query operators are able to adapt to the environment changes, even in the context of a fixed query plan.

Non-Pipelined Query Execution (Inter-Operator):
● Sub-queries are re-scheduled based on uncertainties:
  ○ Execution cost of remote access,
  ○ Size of intermediate results, and
  ○ Unexpected delays.
Adaptive Query Processing Strategies

Adaptive Join Processing (Intra-Operator):
- Adaptivity performed at tuple level.
- Query operators are able to adapt to the environment changes, even in the context of a fixed query plan.

Non-Pipelined Query Execution (Inter-Operator):
- Sub-queries are re-scheduled based on uncertainties:
  - Execution cost of remote access,
  - Size of intermediate results, and
  - Unexpected delays.
Distributed SPARQL Systems
SPARQL Web-based Interfaces

**SPARQL Endpoints**: RESTful services that accept queries over HTTP written in the SPARQL query language and respecting the SPARQL protocol.

```sparql
SELECT ?long
WHERE
```

http://bigdataocean.eu/sparql?query=
SELECT%20%3Flong%20
WHERE
%20%7B%20%3Fport%20%3Chttp%3A%2F%2Fbigdataocean.eu%2Fproperty%2Fname%3E%20%3Fport%20%3Chttp%3A%2F%2Fbigdataocean.eu%2Fresource%2FFunchal%3E%20%3Flong%20%7D
The result of evaluating Q0 corresponds to the result of evaluating Q1 in the RDF dataset accessible through the endpoint <http://bigdataocean.de/sparql>.
SPARQL 1.1 Federation Extension

(1) If $P$ is a triple pattern $t$, then $[P]^{DS}_G = \{\mu \mid \text{dom}(\mu) = \text{var}(t) \text{ and } \mu(t) \in G\}$.

(2) If $P$ is $(P_1 \text{ AND } P_2)$, then $[P]^{DS}_G = [P_1]^{DS}_G \Join [P_2]^{DS}_G$.

(3) If $P$ is $(P_1 \text{ OPT } P_2)$, then $[P]^{DS}_G = [P_1]^{DS}_G \Join [P_2]^{DS}_G$.

(4) If $P$ is $(P_1 \text{ UNION } P_2)$, then $[P]^{DS}_G = [P_1]^{DS}_G \cup [P_2]^{DS}_G$.

(5) If $P$ is (GRAPH $c P_1$) with $c \in I \cup V$, then

$$[P]^{DS}_G = \left\{ \begin{array}{ll}
\{\mu\} & \text{if } c \in \text{names}(DS) \\
\{\mu \cup \mu_c \mid \exists g \in \text{names}(DS) : \mu_c = [c \rightarrow g], \mu \in [P_1]^{DS}_{\text{graph}(g,DS)} \text{ and } \mu_c \sim \mu\} & \text{if } c \in I \setminus \text{names}(DS) \\
\{\} & \text{if } c \in V
\end{array} \right.$$ 

(6) If $P$ is (SERVICE $c P_1$) with $c \in I \cup V$, then

$$[P]^{DS}_G = \left\{ \begin{array}{ll}
\{\mu_0\} & \text{if } c \in \text{dom}(ep) \\
\{\mu \cup \mu_c \mid \exists s \in \text{dom}(ep) : \mu_c = [c \rightarrow s], \mu \in [P_1]^{ep(s)}_{\text{graph}(def,ep(s))} \text{ and } \mu_c \sim \mu\} & \text{if } c \in I \setminus \text{dom}(ep) \\
\{\} & \text{if } c \in V
\end{array} \right.$$ 

(7) If $P$ is $(P_1 \text{ FILTER } R)$, then $[P]^{DS}_G = \{\mu \in [P_1]^{DS}_G \mid \mu \models R\}$.
SPARQL Web-based Interfaces

Triple Pattern Fragment:

Web-based access interface that allows for evaluating a triple pattern over an RDF dataset.

“Accessing a TPF interface is equivalent to accessing a SPARQL endpoint with queries that contain a single triple pattern only” [5]

Client-Server SPARQL Query Engines [5]

Server of Triple Pattern Fragments (TPFs)

TPF: Web-based interface to execute

SPARQL triple patterns

qvessel bdo:mmsi ?mmsi

TPF Clients: Execute

SPARQL queries over a TPF Server

Clients of Triple Pattern Fragments (TPFs)

SPARQL Query Q

Adaptive Client for Triple Pattern Fragments

SPARQL Query $Q$

Input

Query Optimizer

Optimized plan

Routing Policies

Adaptive Engine

Output

Results for $Q$

nLDE [6]:
- Optimization techniques tailored for TPFs
- Autonomous Eddies
- Routing policies for SPARQL

Peer-to-Peer SPARQL Query Engines

Avalanche [7]:
- **Endpoints** may dereference data from other endpoints.
- **On-line** statistical information about relevant data sources.
- **Queries** are executed in a **concurrent** and **distributed** fashion.

Federated SPARQL Query Engines

**SPARQL endpoints:** Web-access interfaces that allow for querying RDF data following SPARQL protocol.
Federated SPARQL Query Engines

- **Heterogeneity**
- **Distribution**
- **Autonomy**
Federated SPARQL Query Engines
Federated SPARQL Query Engines

Distribution

Heterogeneity

Autonomy

Extensions

Fed-DESATUR[3]
HIBISCUS[15]

ANAPSID[8]

FedX[10]

SPLENDID [9]

Federated SPARQL Query Engine

SemaGrow[16]
Adaptivity During Source Selection

No Adaptivity

Coarse-Grained Adaptivity

Fine-Grained Adaptivity

LILAC  FEDRA  Fed-DESATUR
DAW  MULDER  HIBISCUS

ANAPSID  SPLENDID

FedX

SemaGrow
Adaptivity During Source Selection

No Adaptivity

Coarse-Grained Adaptivity

Fine-Grained Adaptivity

MULDER
Adaptivity During Source Selection-FedX

```
SELECT ?mmsi ?name ?flag
WHERE {
  ?vessel bdo:name ?name.
  ?observ bdo:lat ?lat.
  ?observ bdo:date "2017-06-21" .
  ?port bdo:lat ?lat.
  ?port bdo:name "Funchal" }
```

For triple pattern, (s p o):
A query is sent to the **SPARQL endpoints** for identifying **relevant** sources
Adaptivity During Source Selection-FedX

SELECT ?mmsi ?name ?flag
WHERE {
  sq1
  { ?vessel bdo:name ?name. }
  sq2
  sq3
  sq4
  { ?observ bdo:mmsi ?mmsi. }
  sq5
}

Exclusive groups:
Subqueries are executed by only one source
Adaptivity During Source Selection-FedX

Exclusive groups are executed first in the plan
Sources are described in terms of Classes and properties.
Sources are described in terms of Classes and properties.

Source descriptions are exploited during query decomposition.

```
SELECT ?mmsi ?name ?flag
WHERE {
  ?vessel bdo:name ?name.
  ?observ bdo:lat ?lat.
  ?observ bdo:date "2017-06-21".
  ?port bdo:lat ?lat.
  ?port bdo:name "Funchal".
}
```
Query Planning-Mulder

?port bdo:lat ?lat .
?port bdo:name "Funchal"

?observation bdo:lat ?lat .
?observation bdo:date "2017-06-21".

?vessel bdo:name ?name .
Empirical Evaluation

FedBench Benchmark:
- Ten RDF datasets (314K to 44M)

25 Queries:
- Cross Domain (CD)
- Linked Data (LD)
- Life Science (LS)

Ten Complex (C) queries

Completeness versus Execution Time
Empirical Evaluation

FedBench Benchmark:
- Ten RDF datasets (314K to 44M)

25 Queries:
- Cross Domain (CD)
- Linked Data (LD)
- Life Science (LS)

Ten Complex (C) queries

Completeness versus Execution Time
Empirical Evaluation

FedBench Benchmark:
- Ten RDF datasets (314K to 44M)

25 Queries:
- Cross Domain (CD)
- Linked Data (LD)
- Life Science (LS)

Ten Complex (C) queries

Completeness versus Execution Time

Adaptivity is a costly task
Adaptivity During Query Execution

No Adaptivity

FedX
Linked Data in a Federation

SPLENDID

ANAPSID

Fine-Grained Adaptivity
Adaptivity During Query Execution

No Adaptivity

Fine-Grained Adaptivity

Fed-DESATUR

FedX

SPLENDID

DAW HIBISCUS

LILAC FEDRA

ANAPSID MULDER

Fed-DESATUR
Intra-Operator Adaptivity

Intra-Operator Strategies:

- SPARQL operators able to detect when sources become blocked or data traffic is bursty
- Opportunistically produce results as quickly as data arrives from the sources
- Results are produced incrementally
Inter-Operator Adaptivity

Inter-Operator Strategies:

- Produce an **answer** as **soon** as it is computed
- Can **keep** producing **intermediate** results **even** when data a source becomes **blocked**
Inter-Operator Adaptivity

Inter-Operator Strategies:

- Produce an **answer** as **soon** as it is computed
- Can **keep** producing **intermediate** results **even** when data a source becomes **blocked**
Empirical Evaluation

FedBench Benchmark:
- Ten RDF datasets (314K to 44M)

25 Queries:
- Cross Domain (CD)
- Linked Data (LD)
- Life Science (LS)

Ten Complex (C) queries

Completeness versus Execution Time
Empirical Evaluation

FedBench Benchmark:
- Ten RDF datasets (314K to 44M)

25 Queries:
- Cross Domain (CD)
- Linked Data (LD)
- Life Science (LS)

Ten Complex (C) queries

Completeness versus Execution Time
Empirical Evaluation

FedBench Benchmark:
- Ten RDF datasets (314K to 44M)

25 Queries:
- Cross Domain (CD)
- Linked Data (LD)
- Life Science (LS)

Ten Complex (C) queries

Completeness versus Execution Time
Empirical Evaluation

FedBench Benchmark:
- Ten RDF datasets (314K to 44M)
- 25 Queries:
  - Cross Domain (CD)
  - Linked Data (LD)
  - Life Science (LS)
- Ten Complex (C) queries

Completeness versus Execution Time
Empirical Evaluation

FedBench Benchmark:
- Ten RDF datasets (314K to 44M)
- 25 Queries:
  - Cross Domain (CD)
  - Linked Data (LD)
  - Life Science (LS)
- Ten Complex (C) queries

Completeness versus Execution Time

Adaptivity is a costly task
Adaptive Query Processing

Delays simulated with a Gamma distribution ($\alpha=1$, $\beta=0.3$)

Twenty-five queries against DBpedia; basic graph patterns with up to 15 triple patterns. Adaptive query processing strategies are able to speed up query execution in presence of delays.

---

Data Replication-Aware

No Replication

Replication-Aware

Fed-DESATUR

ANAPSID

MULDER

HIBISCUS

Fed-DESATUR

SPLENDID

DAW

LILAC

FEDRA
Each dataset is accessible through **10 endpoints**. Each **endpoint** has data for **100 random queries**. The **same** fragment is replicated in at least **3 endpoints**. **WatDiv datasets**: **50,000 queries**. Rest of the datasets: **10,000 queries**.

---

**Diseasome**: 72,445  
**SWDF**: 198,797  
**DBpedia Geo**: 1,900,004  
**LinkedMDB**: 3,579,610  
**WatDiv1**: 104,532  
**WatDiv100**: 10,934,518

---

Each dataset is accessible through **10 endpoints**. Each endpoint has data for **100 random queries**.
The same fragment is replicated in **at least 3 endpoints**
WatDiv datasets: **50,000 queries**
Rest of the datasets: **10,000 queries**
Experimental Study

Consider replication allows for **producing complete answers** while the **execution time** is reduced.
Open Issues

Diverse types of heterogeneity: Data models, query capabilities, availability.
Open Issues

- Formal Definition
- Access Control
- Security
- Privacy

- Data Evolution
- Synchronization
- Big Data
Conclusions
References

References

References