Intelligent Clients for Replicated Triple Pattern Fragments

Thomas Minier, Hala Skaf-Molli, Pascal Molli and Maria-Esther Vidal

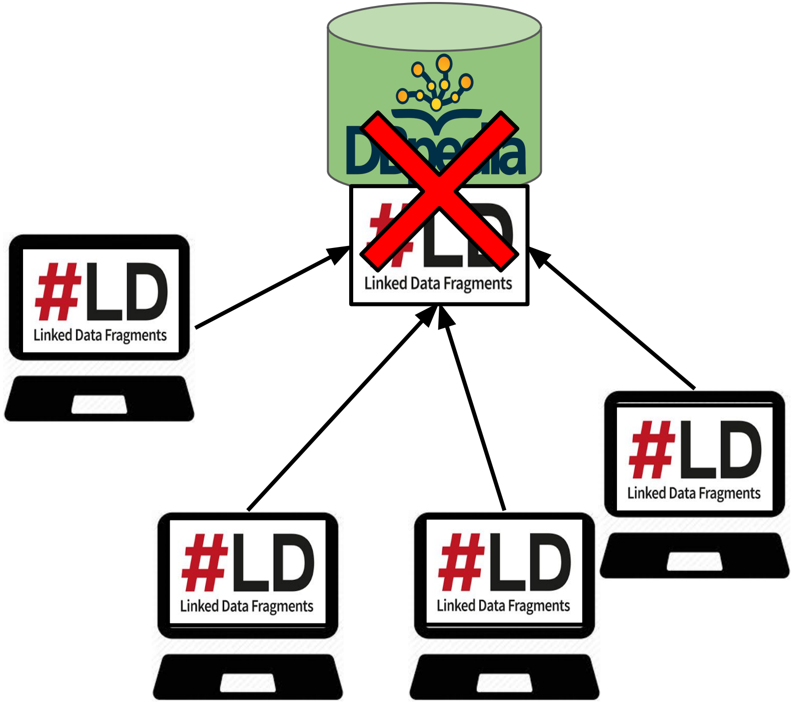
UNIVERSITÉ DE NANTES

TIB LEIBNIZ INFORMATION CENTRE FOR SCIENCE AND TECHNOLOGY UNIVERSITY LIBRARY

ESWC 2018 - Heraklion, Greece
June 6th, 2018

Introduction

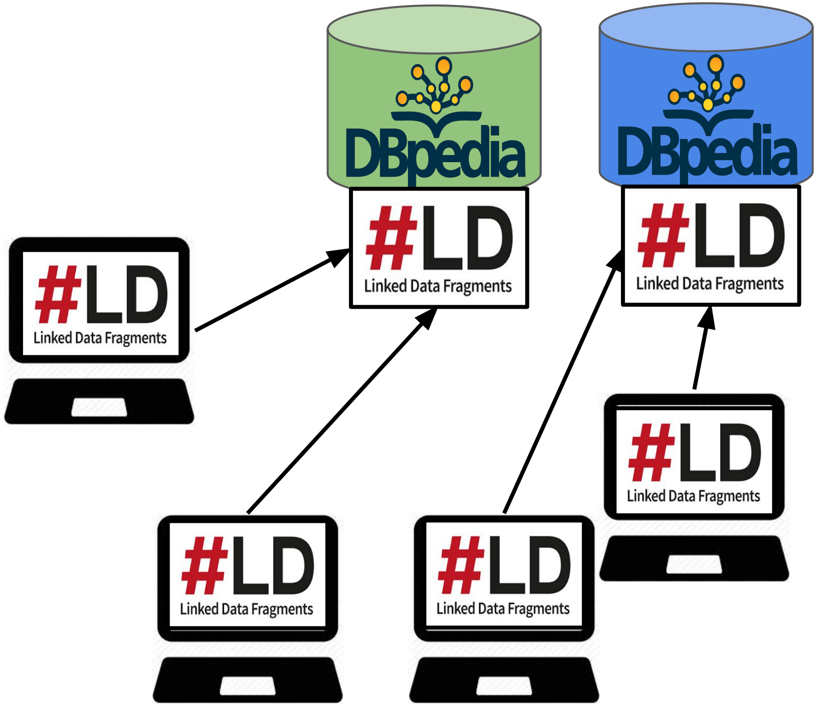
- Following the **Linked Open Data** principles, data providers made available RDF datasets at low-cost using **TPF servers** [1]
- However, **servers availability** remain an issue:
  - Server down
  - Server heavily loaded



[1] Verborgh, Ruben, et al. "Triple Pattern Fragments: A low-cost knowledge graph interface for the Web." *Web Semantics: Science, Services and Agents on the World Wide Web* 37 (2016)

Server Availability

- Data providers **replicate RDF datasets**
  - DBpedia & LANL Linked Data Archive
- **Can we use replicated datasets to improve server availability?**
  - Yes, using **load-balancing**


DBpedia
DBpedia
#LD Linked Data Fragments
#LD Linked Data Fragments
#LD Linked Data Fragments
#LD Linked Data Fragments
#LD Linked Data Fragments
#LD Linked Data Fragments

SPARQL Query load-balancing between Replicated RDF Datasets

- **Good for data providers**
  - Less load -> more available
  - Save €€€ on data hosting
- **Good for data consumers**
  - Tolerance to server failures
  - Tolerance to heavily loaded servers
  - Improve query performance

Problem

How to **balance the load of SPARQL query processing** over replicated **heterogeneous** servers owned by **autonomous** data providers?

Related Work

Triple Pattern Fragments

Existing TPF clients allow to process a federated SPARQL query over a federation of TPF servers [1], *but* they **do not support replication nor client-side load balancing**

Q1: SELECT DISTINCT ?software ?company
WHERE {
    ?software dbo:developer ?company.   *(tp_1)*
    ?company dbo:locationCountry ?country. *(tp_2)*
    ?country rdfs:label "France"@en.   *(tp_3)*
}

DBpedia	11.4s
DBpedia **and** LANL	28.7s

[1] Verborgh, Ruben, et al. "Triple Pattern Fragments: A low-cost knowledge graph interface for the Web." *Web Semantics: Science, Services and Agents on the World Wide Web* 37 (2016)

Linked Data Replication

- Linked Data Replication addressed as a **source-selection** problem [2, 3, 4]
- They **prune** redundant sources != load-balancing

```
Q1: SELECT DISTINCT ?software ?company
WHERE {
        ?software dbo:developer ?company.          (tp_1)
        ?company dbo:locationCountry ?country.     (tp_2)
        ?country rdfs:label "France"@en.           (tp_3)
}
```

DBpedia	11.4s
DBpedia **or** LANL	11.4s **or** 36s

[2] Montoya, G. et al. “Federated Queries Processing with Replicated Fragments.” *ISWC 2015.*

[3] Montoya, G. et al. “Decomposing federated queries in presence of replicated fragments” *Web Semantics: Science, Services and Agents on the World Wide Web (2017)*

[4] Saleem, M. et al. “DAW: duplicate-aware federated query processing over the web of data” *ISWC 2013*

Client-side load-balancing

- **Client-side load-balancing** is well suited for heterogeneous servers [5]
  - **+** Fit well for intelligent TPF clients
  - **+** Respect data providers autonomy
  - **-** Only applied for static files, not for query processing

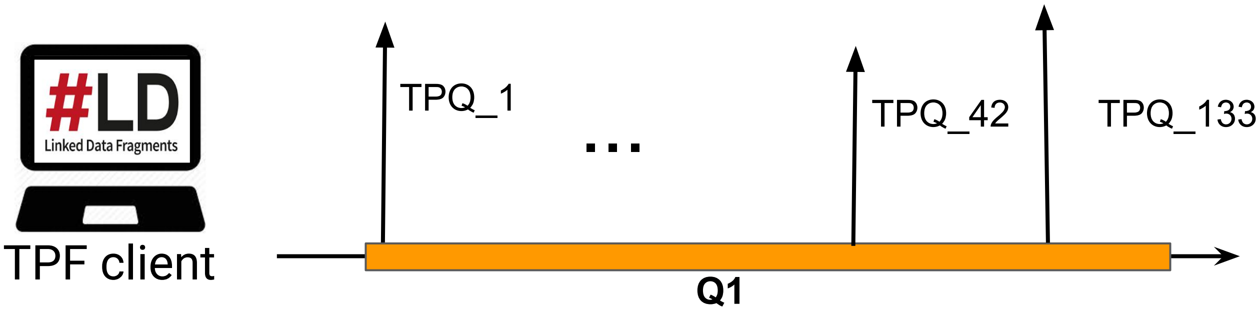
[5] Sandra G Dykes, et al. “*An empirical evaluation of client-side server selection algorithms*”. In INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, Vol. 3. IEEE, 1361–1370.

Ulysses approach

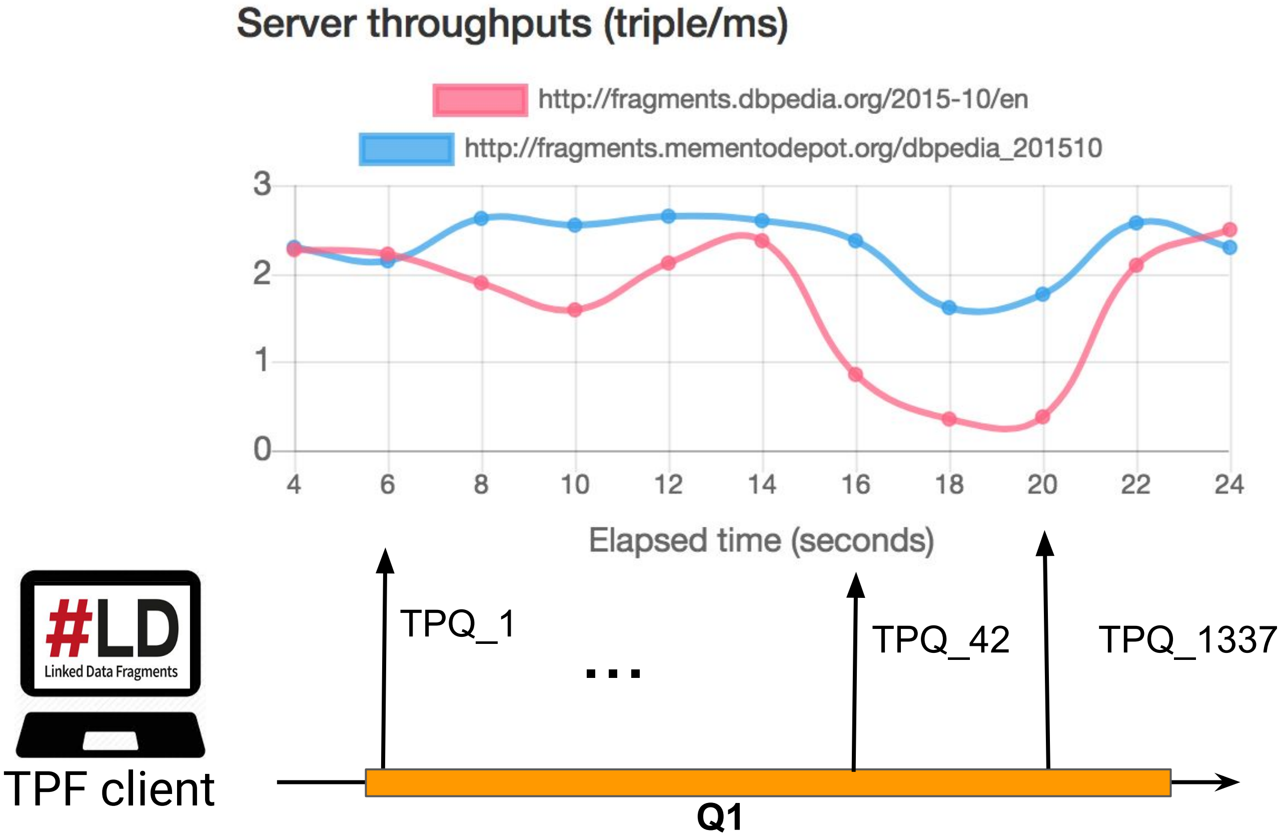
Query evaluation over replicas

Q1: SELECT DISTINCT ?software ?company
WHERE {
    ?software dbo:developer ?company.   *(tp_1)*
    ?company dbo:locationCountry ?country. *(tp_2)*
    ?country rdfs:label "France"@en.   *(tp_3)*
}

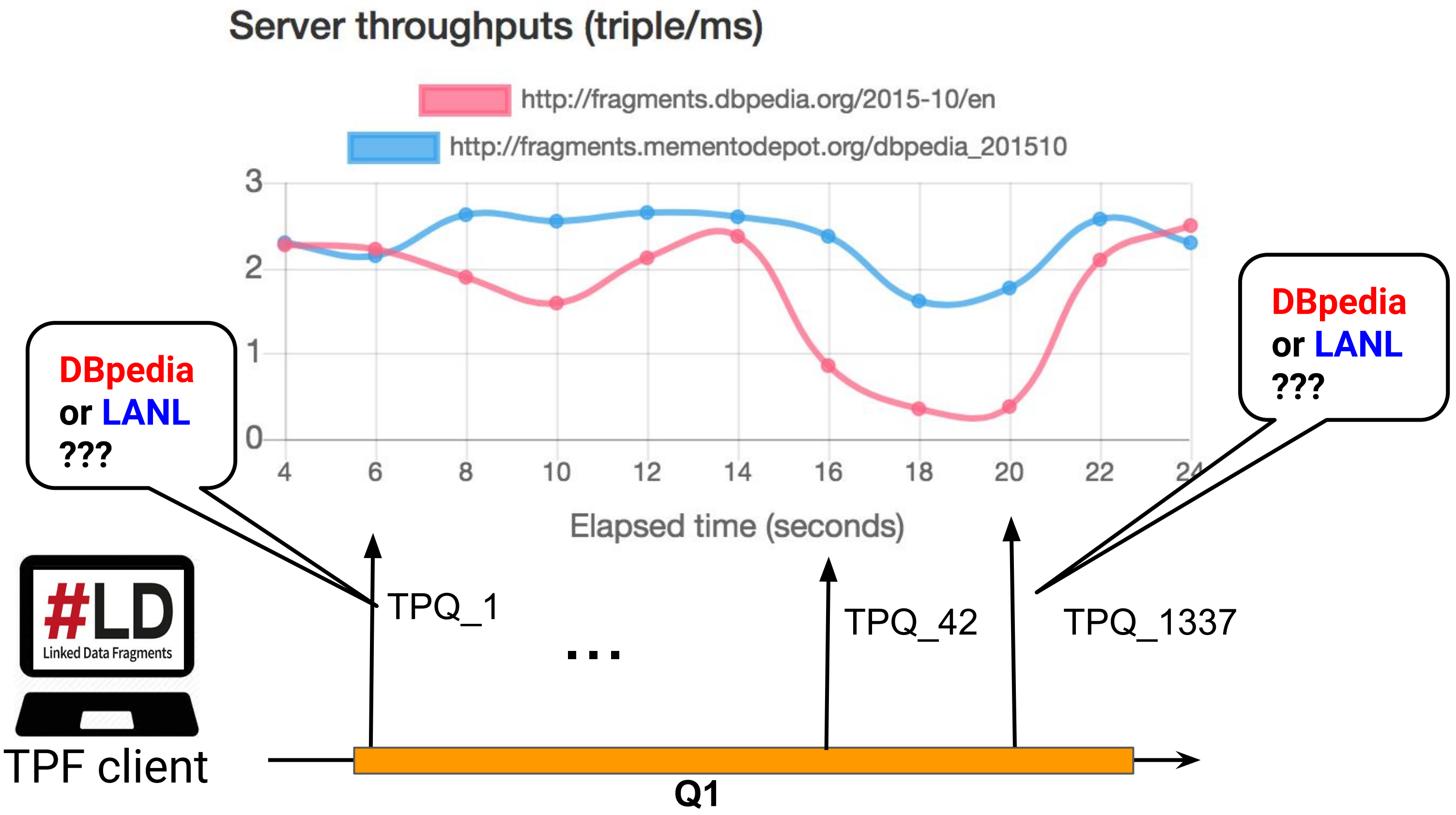
Datasources: DBpedia and a replica from LANL



Servers throughputs change over time



Where to send Triple Pattern Queries?



DBpedia or LANL ???

DBpedia or LANL ???

TPF client

TPQ_1

...

TPQ_42

TPQ_1337

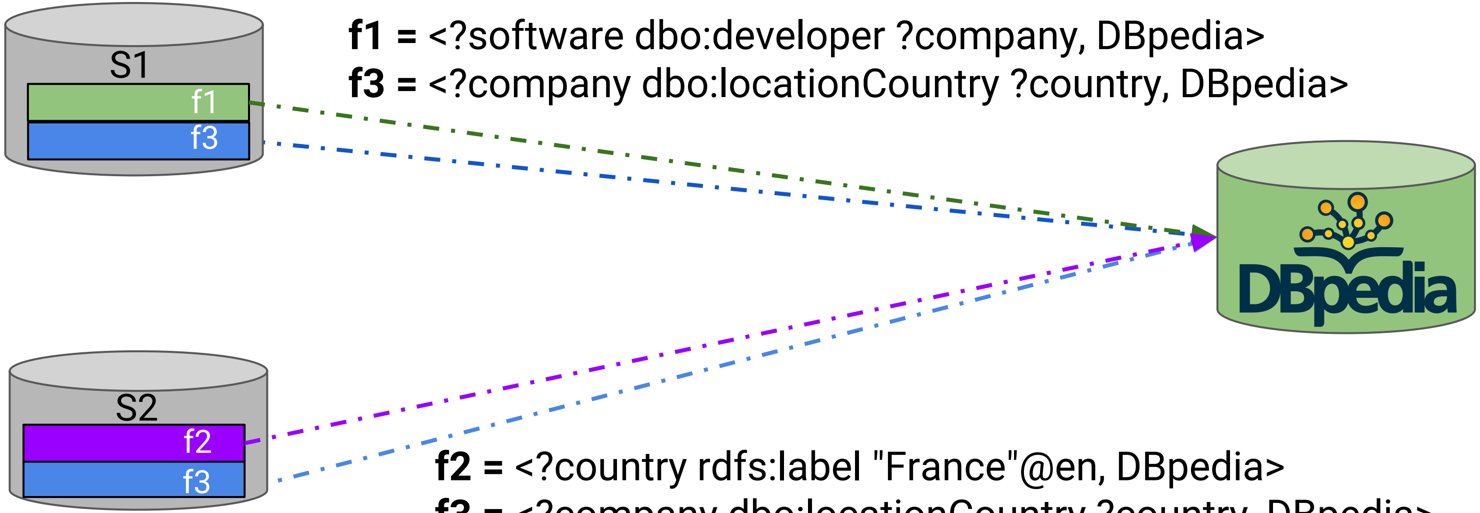
Q1

Ulysses: a replication-aware intelligent TPF client

- A **replication-aware source selection**
  - Total/partial replication
- A **light-weighted cost-model**
  - Heterogeneous TPF servers
- A **client-side load balancer**
  - Distributing SPARQL query evaluation

Partial replication model

Fragments of RDF datasets are replicated [2,6]



f1 = <?software dbo:developer ?company, DBpedia>
f3 = <?company dbo:locationCountry ?country, DBpedia>

f2 = <?country rdfs:label "France"@en, DBpedia>
f3 = <?company dbo:locationCountry ?country, DBpedia>

[2] Montoya, Gabriela, et al. “Federated Queries Processing with Replicated Fragments.” *ISWC 2015*.
[6] Ibáñez, Luis-Daniel, et al. "Col-graph: Towards writable and scalable linked open data." *ISWC* 2014.

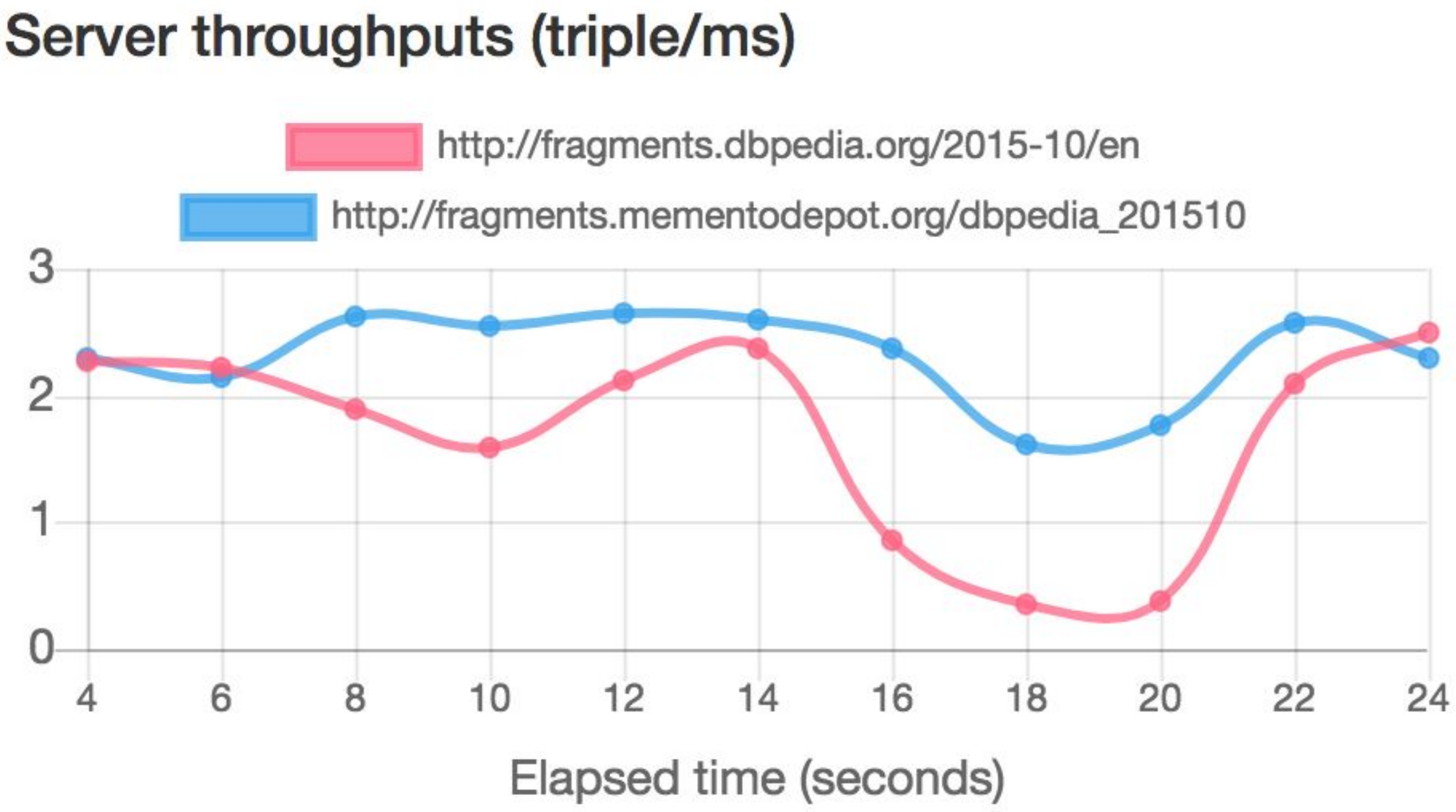
Ulysses replication-aware source selection

- Replicated fragments are defined using a **catalog** [2]
- Describes which fragment is hosted on which server
- Ulysses loads the catalog when starting

Fragment	**Location**
f1 = <?software dbo:developer ?company, DBpedia>	S1
f2 = <?country rdfs:label "France"@en, DBpedia>	S2
f3 = <?company dbo:locationCountry ?country, DBpedia>	S1, S2

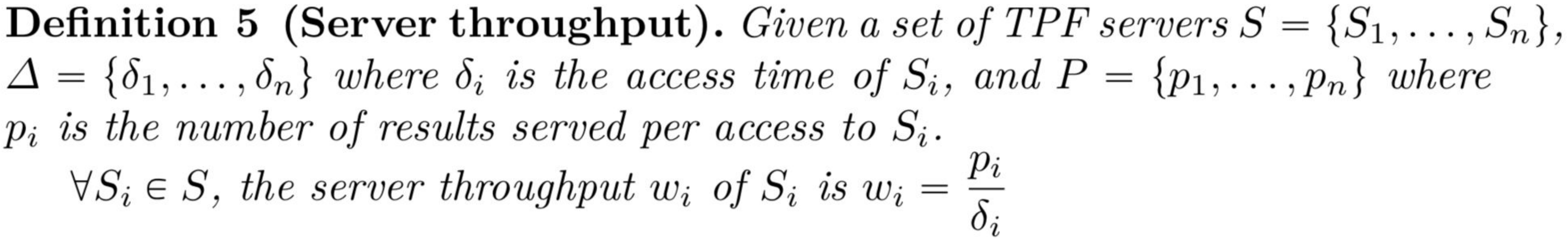
[2] Montoya, Gabriela, et al. “Federated Queries Processing with Replicated Fragments.” *ISWC 2015*.

How to get server throughput?



Computing Server throughput

- A **server throughput** is deduced from its **access time**
  - Triple patterns can be evaluated in approximate **constant time** [7] (with HDT backend)
- During query processing, a TPF client executes many triple pattern queries
  - A lot of free probes!



Definition 5 (Server throughput). *Given a set of TPF servers* $S = \{S_1, \ldots, S_n\}$, $\Delta = \{\delta_1, \ldots, \delta_n\}$ *where* δ_i *is the access time of* S_i, *and* $P = \{p_1, \ldots, p_n\}$ *where* p_i *is the number of results served per access to* S_i.

$$\forall S_i \in S, \text{ the server throughput } w_i \text{ of } S_i \text{ is } w_i = \frac{p_i}{\delta_i}$$

[7] Fernández, J.D. et al. “Binary RDF representation for publication and exchange (HDT)”. *Web Semantics: Science, Services and Agents on the World Wide Web* (2013)

Computing Server throughput

S1

Access time δ_1 = 100ms

Page size p_1 = 100 triples

S2

Access time δ_2 = 100ms

Page size p_2 = 400 triples

S3

Access time δ_3 = 500ms

Page size p_3 = 400 triples

Computing Server throughput

S1

Access time δ_1 = 100ms

Page size p_1 = 100 triples

⇒ **Server throughput ω_1 = 1 triples/ms**

S2

Access time δ_2 = 100ms

Page size p_2 = 400 triples

⇒ **Server throughput ω_2 = 4 triples/ms**

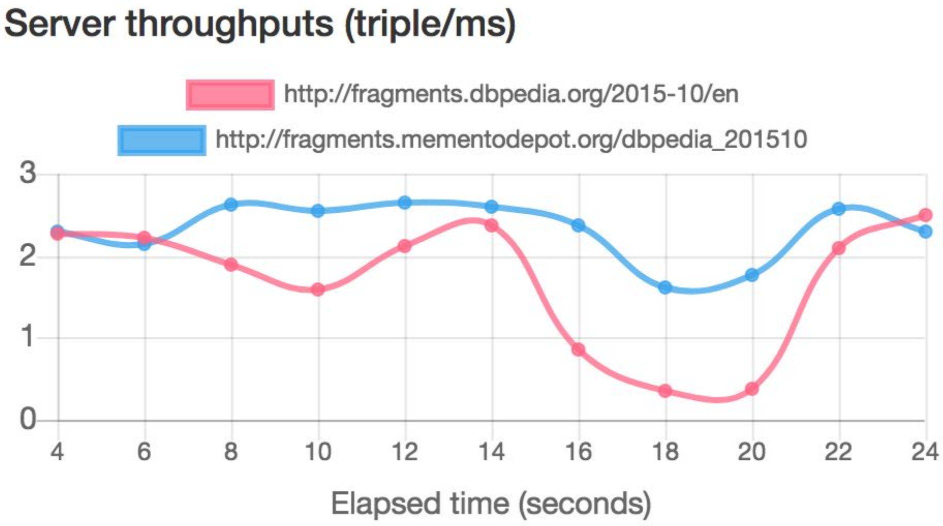
S3

Access time δ_3 = 500ms

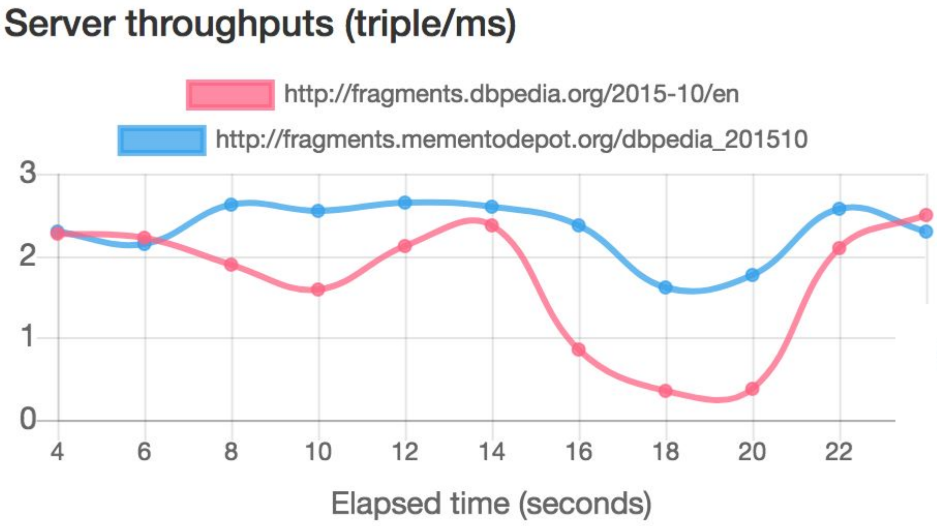
Page size p_3 = 400 triples

⇒ **Server throughput ω_3 = 0.8 triples/ms**

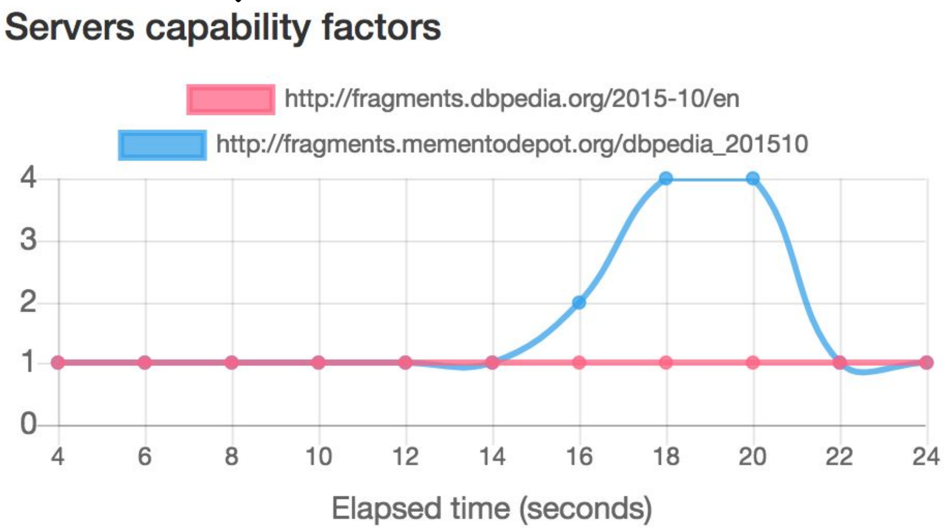
Hard to compare: normalize!



Hard to compare: normalize!

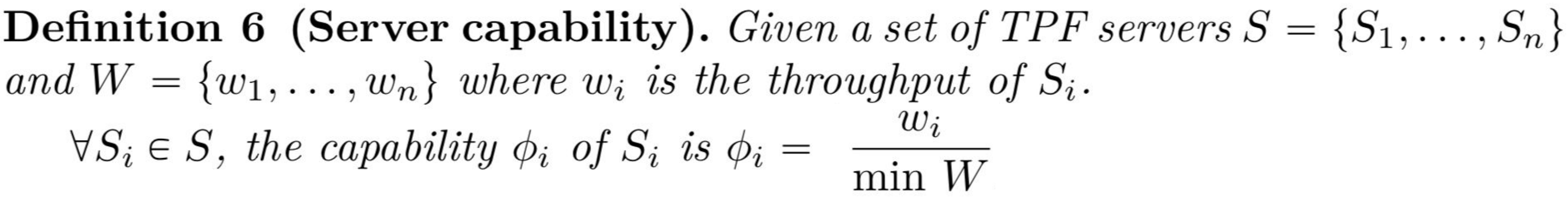


Server throughputs (triple/ms)



Servers capability factors

Computing TPF servers capabilities



Definition 6 (Server capability). *Given a set of TPF servers* $S = \{S_1, \ldots, S_n\}$ *and* $W = \{w_1, \ldots, w_n\}$ *where* w_i *is the throughput of* S_i.

$\forall S_i \in S$, *the capability* ϕ_i *of* S_i *is* $\phi_i = \dfrac{w_i}{\min W}$

Computing TPF servers capabilities

S1

Access time δ_1 = 100ms

Page size p_1 = 100 triples

Server throughput ω_1 = 1 triples/ms

S2

Access time δ_2 = 100ms

Page size p_2 = 400 triples

Server throughput ω_2 = 4 triples/ms

S3

Access time δ_3 = 500ms

Page size p_3 = 400 triples

Server throughput ω_3 = 0.8 triples/ms

Computing TPF servers capabilities

Access time δ_1 = 100ms

Page size p_1 = 100 triples

Server throughput ω_1 = 1 triples/ms

⇩

Capability factor ϕ_1 = 1.25

Access time δ_2 = 100ms

Page size p_2 = 400 triples

Server throughput ω_2 = 4 triples/ms

⇩

Capability factor ϕ_2 = 6.25

Access time δ_3 = 500ms

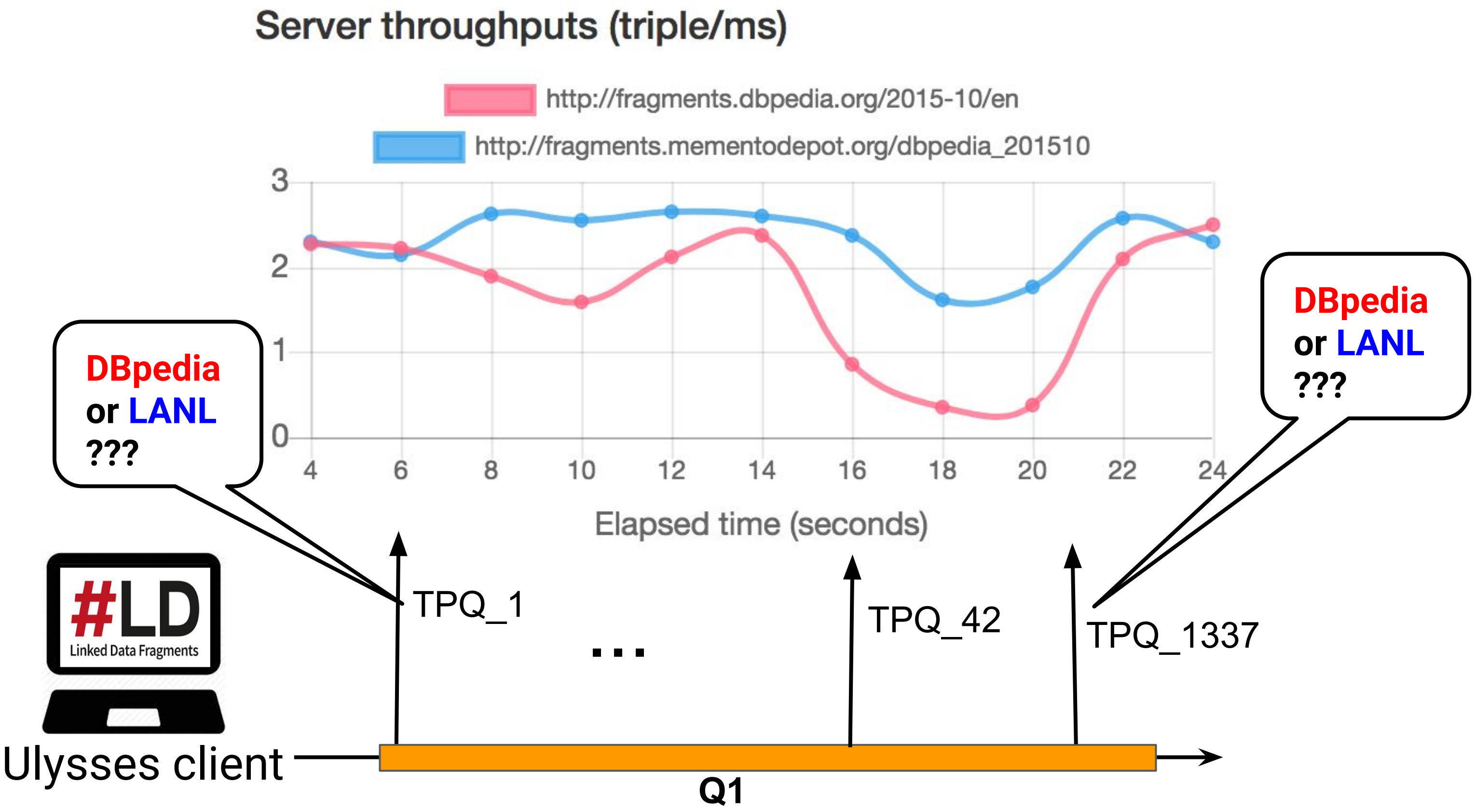
Page size p_3 = 400 triples

Server throughput ω_3 = 0.8 triples/ms

⇩

Capability factor ϕ_3 = 1

Ulysses in action



DBpedia or LANL ???

DBpedia or LANL ???

TPQ_1

...

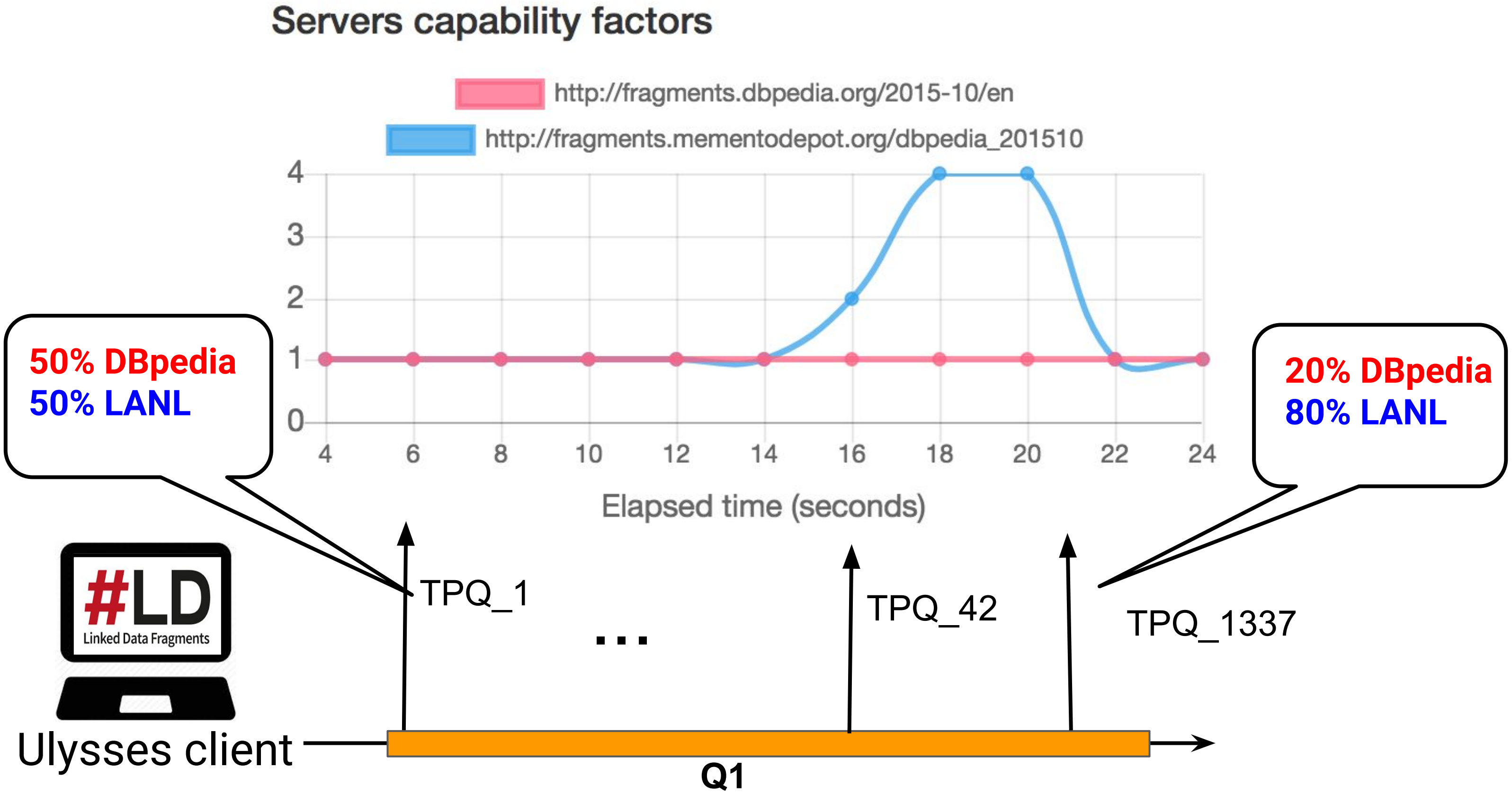
TPQ_42

TPQ_1337

Ulysses client

Q1

Ulysses in action



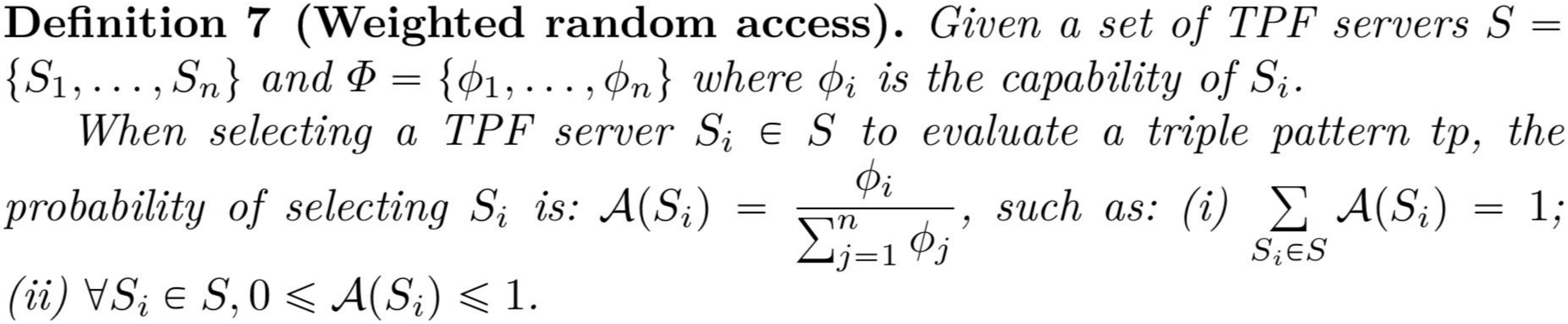
Servers capability factors

50% DBpedia
50% LANL

20% DBpedia
80% LANL

Ulysses client

Weighted random access of TPF servers



Definition 7 (Weighted random access). *Given a set of TPF servers* $S = \{S_1, \ldots, S_n\}$ *and* $\Phi = \{\phi_1, \ldots, \phi_n\}$ *where* ϕ_i *is the capability of* S_i.

When selecting a TPF server $S_i \in S$ *to evaluate a triple pattern tp, the probability of selecting* S_i *is:* $\mathcal{A}(S_i) = \frac{\phi_i}{\sum_{j=1}^{n} \phi_j}$, *such as: (i)* $\sum\limits_{S_i \in S} \mathcal{A}(S_i) = 1$; *(ii)* $\forall S_i \in S, 0 \leqslant \mathcal{A}(S_i) \leqslant 1$.

Weighted random access of TPF servers

S1 Capability factor ϕ_1 = 1.25

S2 Capability factor ϕ_2 = 6.25

S3 Capability factor ϕ_3 = 1

Weighted random access of TPF servers

S1 Capability factor $\phi_1 = 1.25$ $\Longrightarrow$ **Access probability A_1 = 14.7%**

S2 Capability factor $\phi_2 = 6.25$ $\Longrightarrow$ **Access probability A_2 = 73.5%**

S3 Capability factor $\phi_3 = 1$ $\Longrightarrow$ **Access probability A_3 = 11.7%**

Experimental Study

Experimental setup

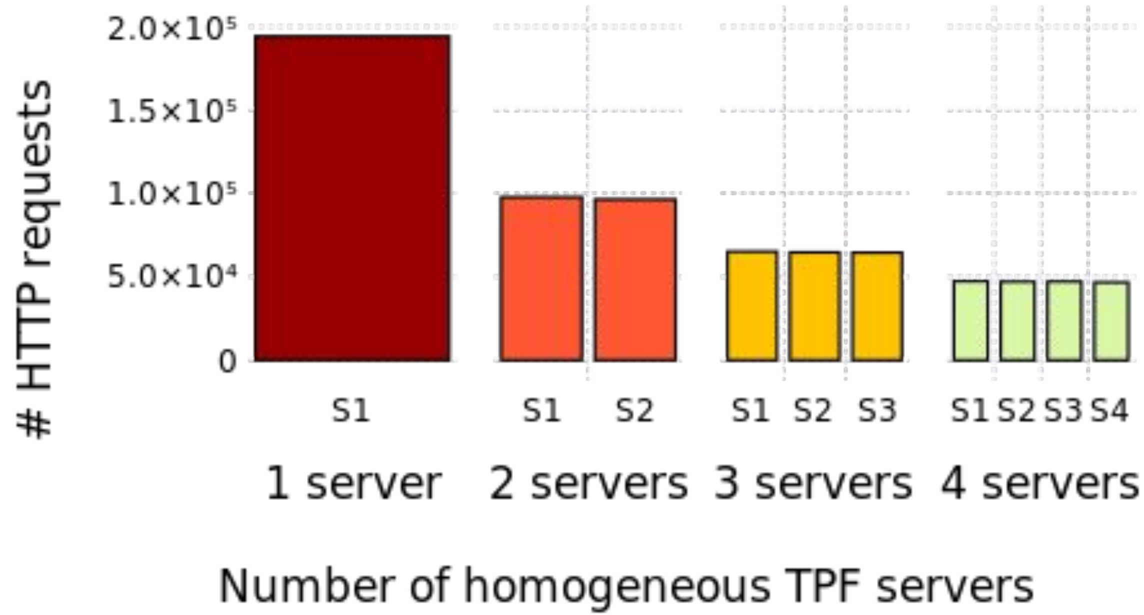
- **Dataset:** Waterloo SPARQL Diversity Test Suite [8] (WatDiv) synthetic dataset with 10^7 triples
- **Queries:** 100 random WatDiv queries (STAR, PATH and SNOWFLAKE shaped SPARQL queries)
- **Replication configurations:**
  - ***Total replication:*** each server replicates the whole dataset
  - ***Partial replication:*** fragments are created from the 100 random queries and are replicated up to two times.

[8] Aluc, G. et al. “Diversified stress testing of RDF data management systems”. In *ISWC 2014*

Experimental setup

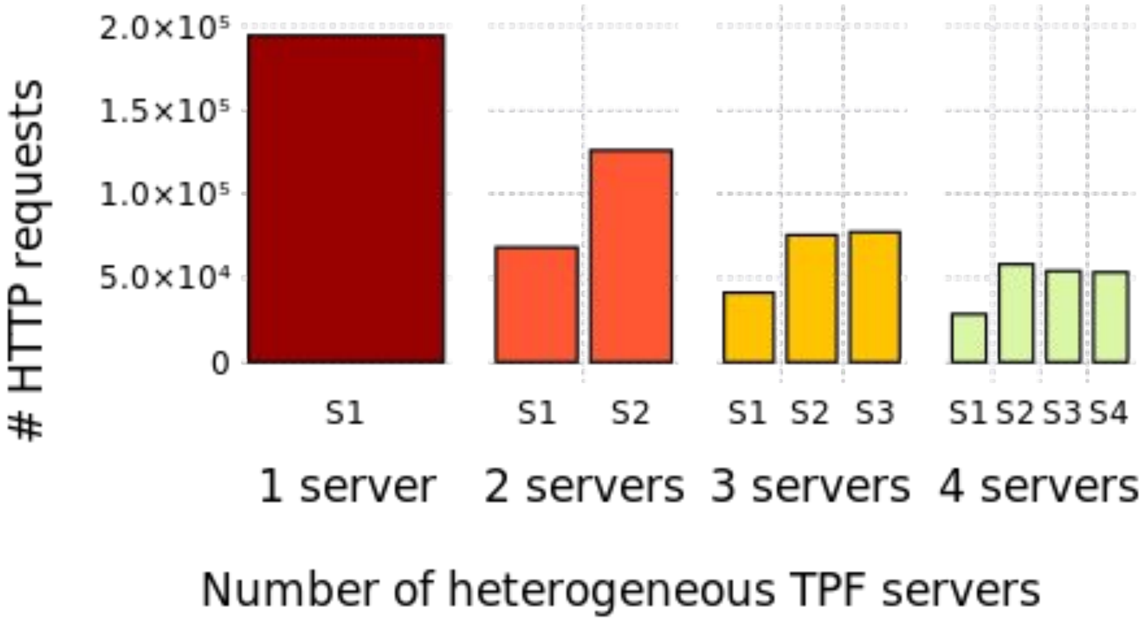
- **Servers:** hosted on Amazon EC2 cloud using *t2.micro* instances
- **Network configurations:**
  - HTTP proxies are used to simulate network latencies and special conditions
  - ***Homogeneous:*** all servers have access latencies of 300ms.
  - ***Heterogeneous:*** The first server has an access latency of 900ms, and other servers have access latencies of 300ms.

Ulysses balance the load according to servers processing capabilities



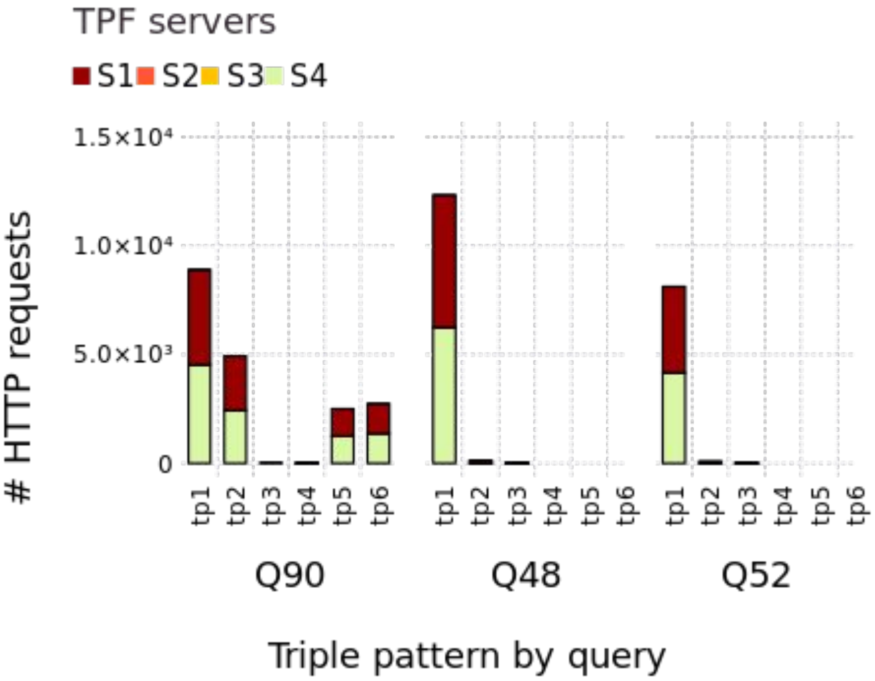
Homogeneous servers and **total replication**

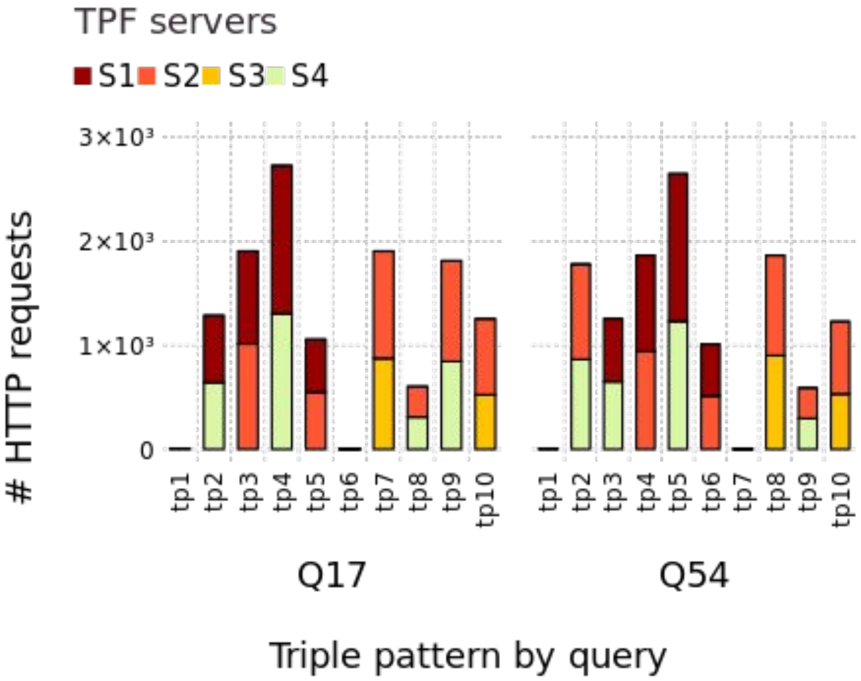
Ulysses balance the load according to servers processing capabilities



Heterogeneous servers and **total replication**

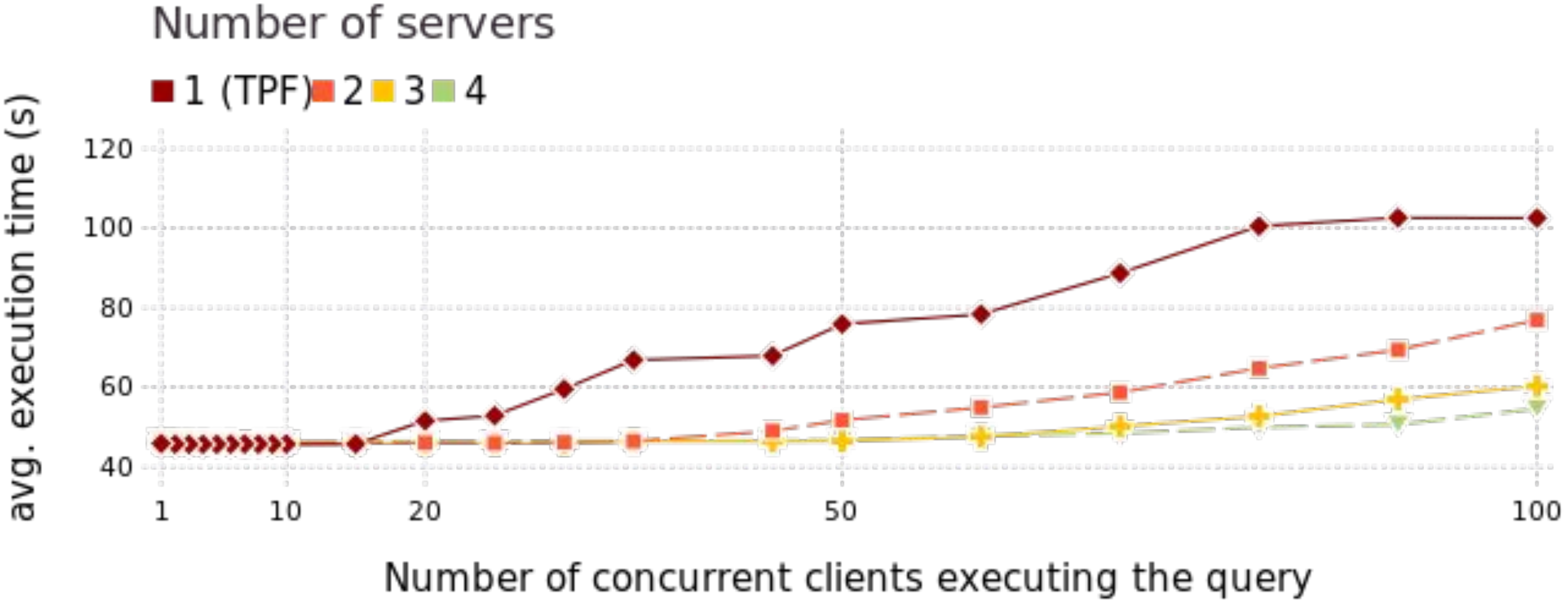
Ulysses balance the load according to servers processing capabilities



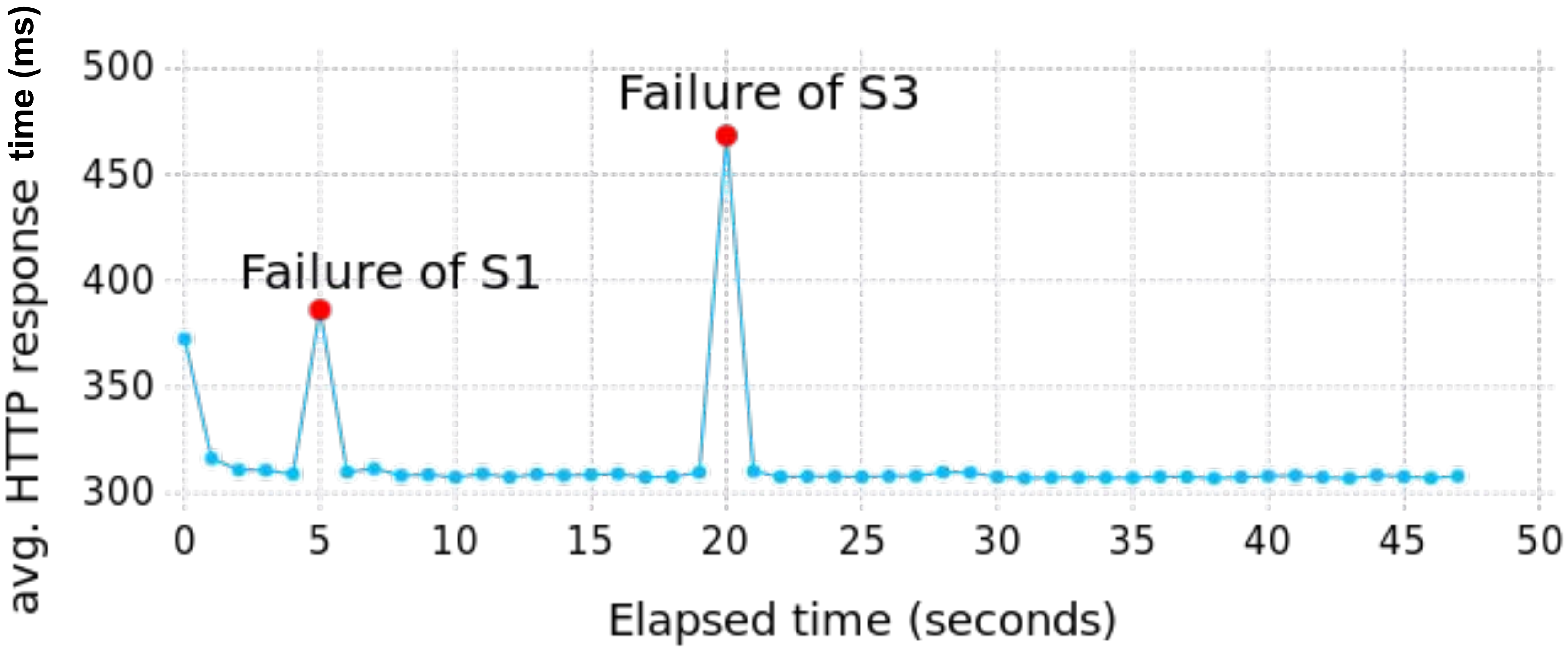


Homogeneous servers and **partial replication**

Ulysses improves query execution time under the load

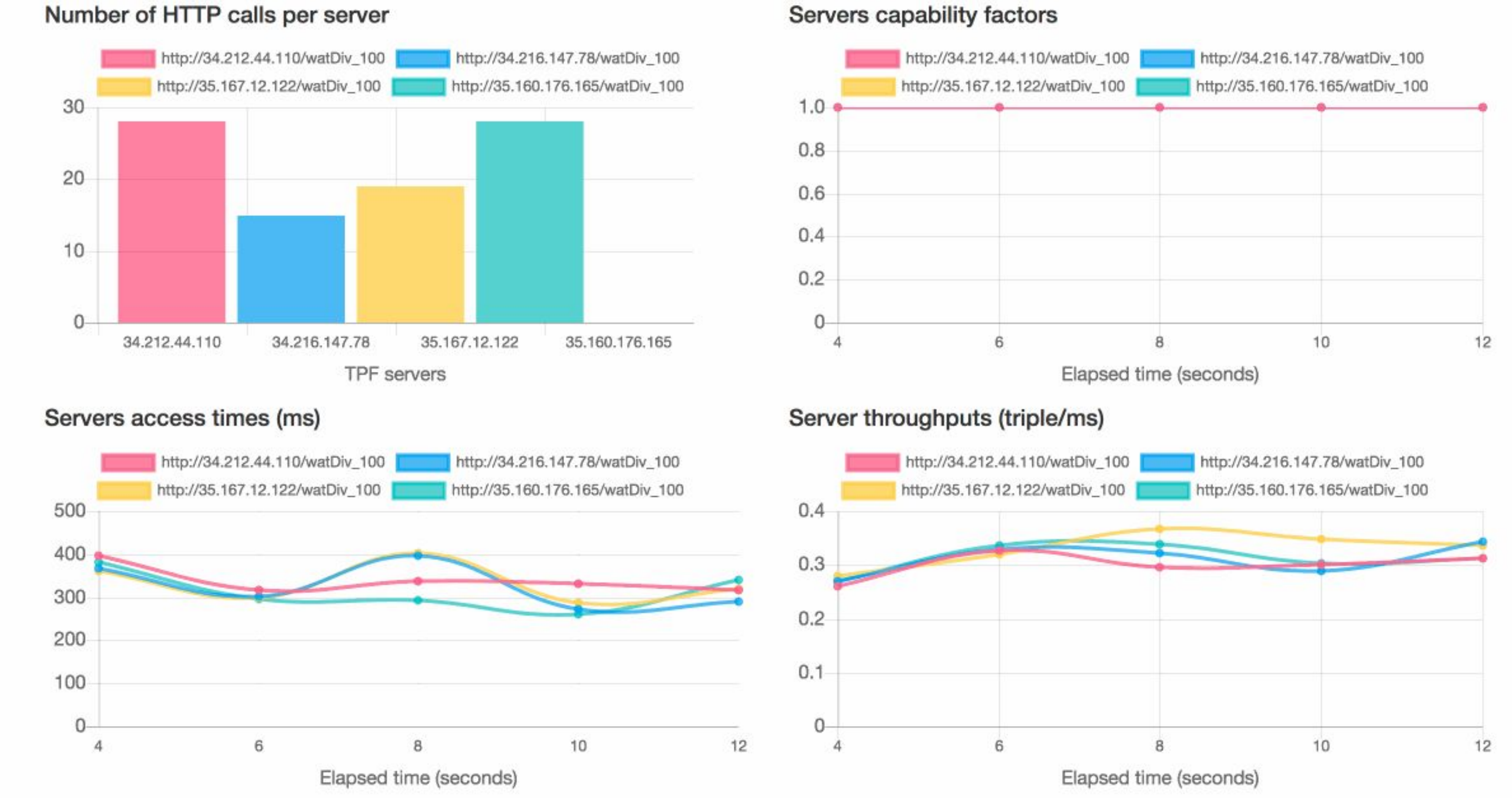


Ulysses tolerates server failures



S1, S2, S3 **homogeneous**: S1 fails at 5s and S3 fails at 20s

Ulysses in real-life
http://ulysses-demo.herokuapp.com



Number of HTTP calls per server

Servers capability factors

Servers access times (ms)

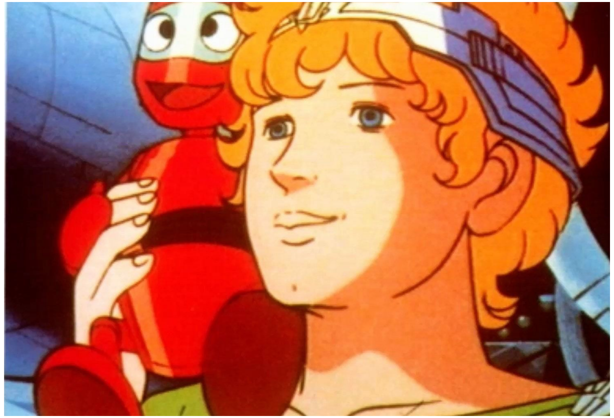
Server throughputs (triple/ms)

Conclusion

- How to **balance the load of SPARQL query processing** over replicated **heterogeneous** servers owned by **autonomous** data providers?
  - Using a **client-side load-balancer** based on Ulysses **cost-model**
  - Require **no changes** from data providers!

Future Works

- **How to build the catalog** of replicated fragments?
  - Provided by data providers as **metadata**
  - A central **index** of replicated RDF datasets
- Consider **divergence** over replicated data
  - Load-balance only if datasets are *k-atomic* [9] or *delta-consistent* [10]



[9] A. Aiyer et al. “*On the availability of non strict quorum systems*”. In Proceedings of the 19th International Symposium on Distributed Computing (DISC) (2005)

[10] Cao, J. et al. “*Data consistency for cooperative caching in mobile environments.*” Computer (2007)

Intelligent Clients for Replicated Triple Pattern Fragments

Come to see the demo tomorrow! (290)
http://ulysses-demo.herokuapp.com

ESWC 2018 - Heraklion, Greece
June 6th, 2018