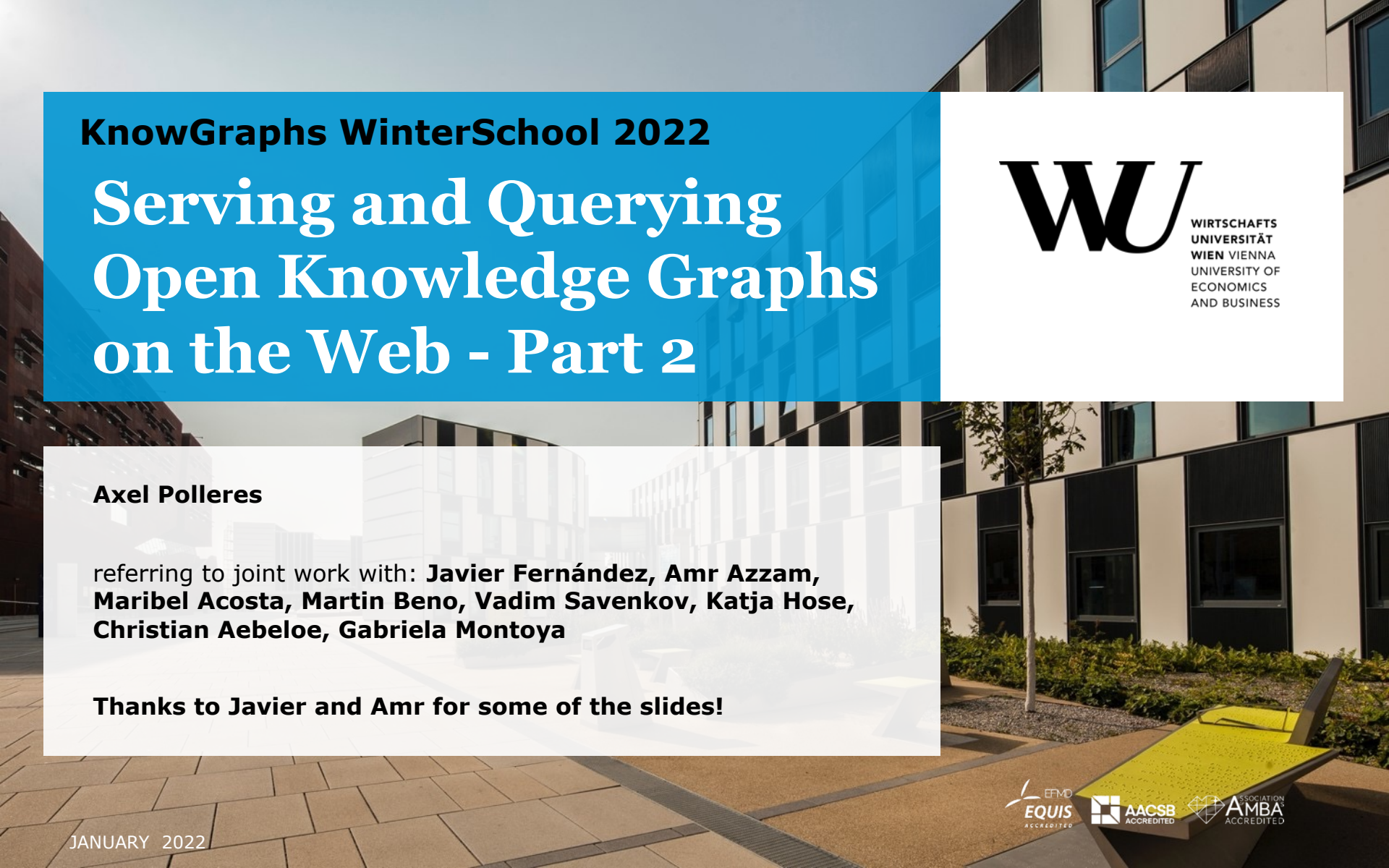


KnowGraphs WinterSchool 2022

Serving and Querying Open Knowledge Graphs on the Web - Part 2

WU WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

Axel Polleres

referring to joint work with: **Javier Fernández, Amr Azzam, Maribel Acosta, Martin Beno, Vadim Savenkov, Katja Hose, Christian Aebeloe, Gabriela Montoya**

Thanks to Javier and Amr for some of the slides!

JANUARY 2022

What I've planned for today:

- **Part 1:**
  - Interlude
  - Practical Tutorial on querying Open KGs with SPARQL
  - Challenges/limitations of SPARQL over public endpoints
- **Part 2:**
  - Serve and query KGs for local processing – HDT
  - Addressing the SPARQL endpoint bottleneck – where are we?
    - Linked Data Fragments
    - Smart-KG
    - Wise-KG

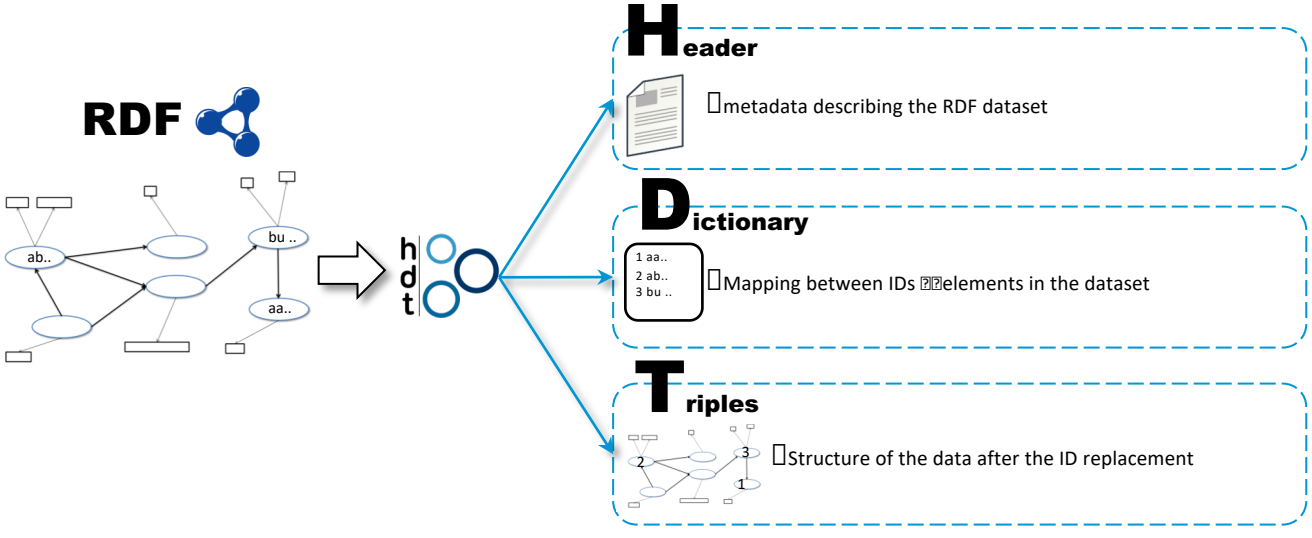
HDT - a Linked Data hacker toolkit

- Highly compact serialization of RDF
- W3C member submission 2011: https://www.w3.org/Submission/HDT/
- Allows fast RDF retrieval in compressed space (without prior decompression)
  - Includes internal indexes to solve basic queries with small memory footprint.
    - Very fast on basic queries (triple patterns), x 1.5 faster than Virtuoso, Jena, RDF3X.
    - Supports FULL SPARQL as the compressed backend store of **Jena**, with an efficiency on the same scale as current more optimized solutions

▷Slightly more but you can query!

- Challenges:
  - Publisher has to pay a bit of overhead to convert the RDF dataset to HDT (but then it is ready to consume efficiently!)
  - Inefficient for live updates

HDT (Header-Dictionary-Triples) Overview

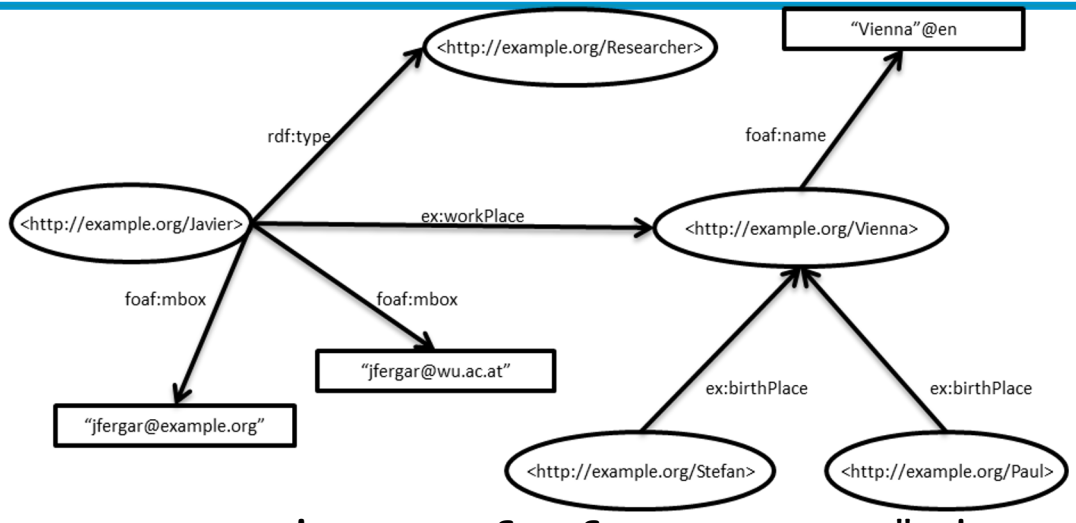


HDT – Header information:

```
$ hdtInfo wikidata20210305.hdt
```

```
<file://[latest-all.ttl.gz]> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://purl.org/HDT/hdt#Dataset> .
<file://[latest-all.ttl.gz]> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://rdfs.org/ns/void#Dataset> .
<file://[latest-all.ttl.gz]> <http://rdfs.org/ns/void#triples> "14578569927" .
<file://[latest-all.ttl.gz]> <http://rdfs.org/ns/void#properties> "38867" .
<file://[latest-all.ttl.gz]> <http://rdfs.org/ns/void#distinctSubjects> "1625057179" .
<file://[latest-all.ttl.gz]> <http://rdfs.org/ns/void#distinctObjects> "2538585808" .
<file://[latest-all.ttl.gz]> <http://purl.org/HDT/hdt#formatInformation> "_:format" .
_:format <http://purl.org/HDT/hdt#dictionary> "_:dictionary" .
_:format <http://purl.org/HDT/hdt#triples> "_:triples" .
<file://[latest-all.ttl.gz]> <http://purl.org/HDT/hdt#statisticalInformation> "_:statistics" .
<file://[latest-all.ttl.gz]> <http://purl.org/HDT/hdt#publicationInformation> "_:publicationInformation" .
_:publicationInformation <http://purl.org/dc/terms/issued> "2021-04-24T12:42Z" .
_:dictionary <http://purl.org/dc/terms/format> <http://purl.org/HDT/hdt#dictionaryFour> .
_:dictionary <http://purl.org/HDT/hdt#dictionarynumSharedSubjectObject> "1451915667" .
_:triples <http://purl.org/dc/terms/format> <http://purl.org/HDT/hdt#triplesBitmap> .
_:triples <http://purl.org/HDT/hdt#triplesnumTriples> "14578569927" .
_:triples <http://purl.org/HDT/hdt#triplesOrder> "SPO" .
_:statistics <http://purl.org/HDT/hdt#hdtSize> "159085366343" .
```

Dictionary+Triples partition

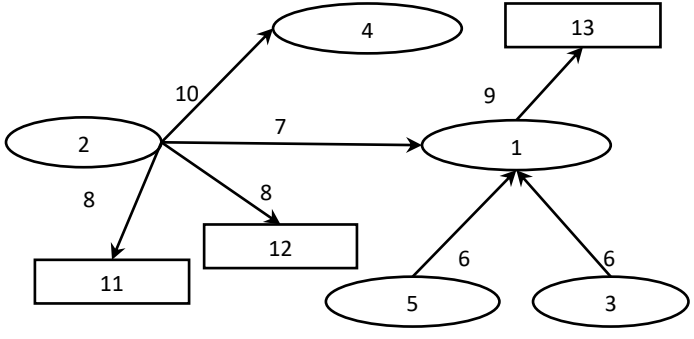


Hint: imagine HDT for now as "SPO-sorted Turtle"

```
ex:Vienna   foaf:name       "Vienna"@en.
ex:Javier   ex:workPlace    ex:Vienna;
            foaf:mbox       "jfergar@example.org",
                            "jfergar@wu.ac.at";
            rdf:type        ex:Researcher .
ex:Paul     ex:birthplace   ex:Vienna.
ex:Stefan   ex:birthplace   ex:Vienna.
```

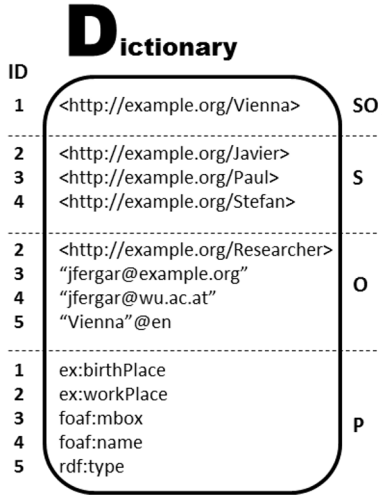
Dictionary+Triples partition

1	ex:Vienna
2	ex:Javier>
3	ex:Paul>
4	ex:Researcher
5	ex:Stefan
6	ex:birthPlace
7	ex:workPlace
8	foaf:mbox
9	foaf:name
10	rdf:type
11	“jfergar@example.org”
12	“jfergar@wu.ac.at”
13	“Vienna”@en



Dictionary (in practice)

1	ex:Vienna
2	ex:Javier
3	ex:Paul
4	ex:Researcher
5	ex:Stefan
6	ex:birthPlace
7	ex:workPlace
8	foaf:mbox
9	foaf:name
10	rdf:type
11	“jfergar@example.org”
12	“jfergar@wu.ac.at”
13	“Vienna”@en



i.e.,dictionary is not exactly "SPO-sorted" but "SO-S-O-P"-sorted

- Split by role
- Prefix-Based compression for each role
  - Efficient ID+String operations

Dictionary compression: Plain Front Coding (PFC)

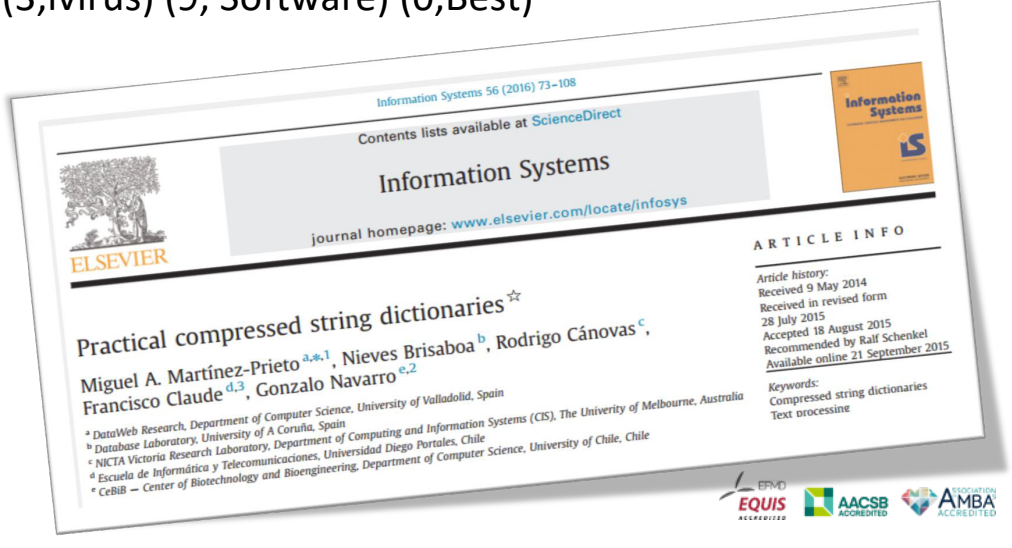
relies on sophisticated prefix-based compression

Each string is encoded with two values

- An integer representing the number of characters shared with the previous string
- A sequence of characters representing the suffix that is not shared with the previous string

A
An
Ant
Antivirus
Antivirus Software
Best

➡ (0,a) (1,n) (2,t) (3,ivirus) (9, Software) (0,Best)


Information Systems 56 (2016) 73–108

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

ELSEVIER

Practical compressed string dictionaries☆

Miguel A. Martínez-Prieto[a,*,1], Nieves Brisaboa[b], Rodrigo Cánovas[c], Francisco Claude[d,3], Gonzalo Navarro[e,2]

[a] DataWeb Research, Department of Computer Science, University of Valladolid, Spain
[b] Database Laboratory, University of A Coruña, Spain
[c] NICTA Victoria Research Laboratory, Department of Computing and Information Systems (CIS), The University of Melbourne, Australia
[d] Escuela de Informática y Telecomunicaciones, Universidad Diego Portales, Chile
[e] CeBiB — Center of Biotechnology and Bioengineering, Department of Computer Science, University of Chile, Chile

ARTICLE INFO

Article history:
Received 9 May 2014
Received in revised form
28 July 2015
Accepted 18 August 2015
Recommended by Ralf Schenkel
Available online 21 September 2015

Keywords:
Compressed string dictionaries
Text processing

Bitmap Triples Encoding

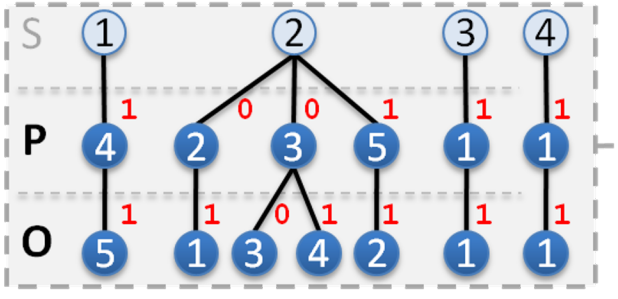
- Bitmap Triples, idea a bit similar to "sorted Turtle":

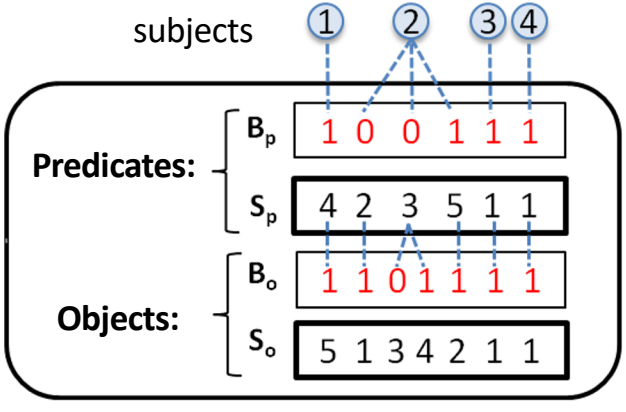
```
ex:Vienna   foaf:name      "Vienna"@en.
ex:Javier   ex:workPlace   ex:Vienna;
            foaf:mbox      "jfergar@example.org",
                           "jfergar@wu.ac.at";
            rdf:type       ex:Researcher .
ex:Paul     ex:birthplace  ex:Vienna.
ex:Stefan   ex:birthplace  ex:Vienna.
```

```
1 4 5.
2 2 1;
  3 3,
    4;
  5 2.
3 1 1.
4 1 1.
```

Dictionary

ID		
1	<http://example.org/Vienna>	SO
2	<http://example.org/Javier>	S
3	<http://example.org/Paul>	
4	<http://example.org/Stefan>	
2	<http://example.org/Researcher>	O
3	"jfergar@example.org"	
4	"jfergar@wu.ac.at"	
5	"Vienna"@en	
1	ex:birthPlace	P
2	ex:workPlace	
3	foaf:mbox	
4	foaf:name	
5	rdf:type	

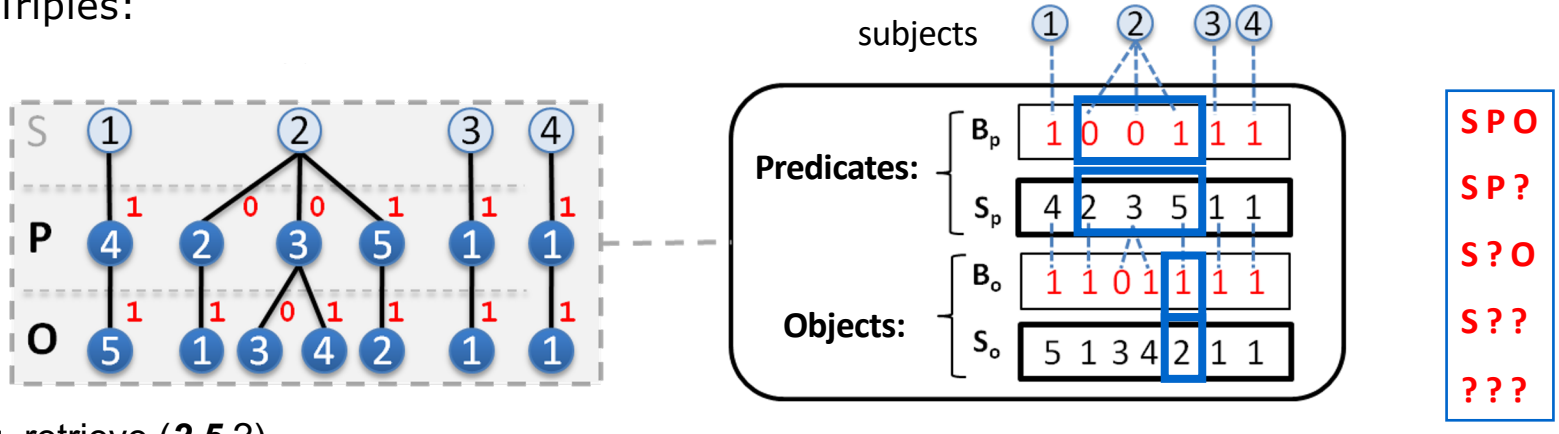




We index the bitsequences to provide a SPO index

Bitmap Triples Encoding

- Bitmap Triples:



E.g. retrieve (***2***,***5***,?)

Find the position of the first and **second** '1'-bits in B_p (`select`)

Binary search on the list of predicates S_p in this range, looking for 5

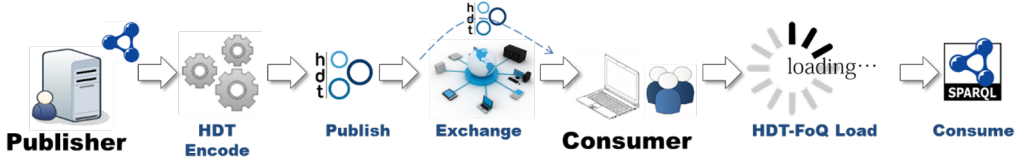
Note that such predicate 5 is in position 4 of S_p

Find the position of the **fourth** '1'-bit in B_o (`select`) → 5

i.e. retrieve 5th value of S_o → **2**

On-the-fly indexes: HDT-FoQ (Focus-on-Querying indexes)

- From the exchanged HDT to the functional HDT-FoQ:
  - Publish and Exchange HDT (i.e., B_p,S_p,B_o,S_o from last slide) and
  - At the consumer:

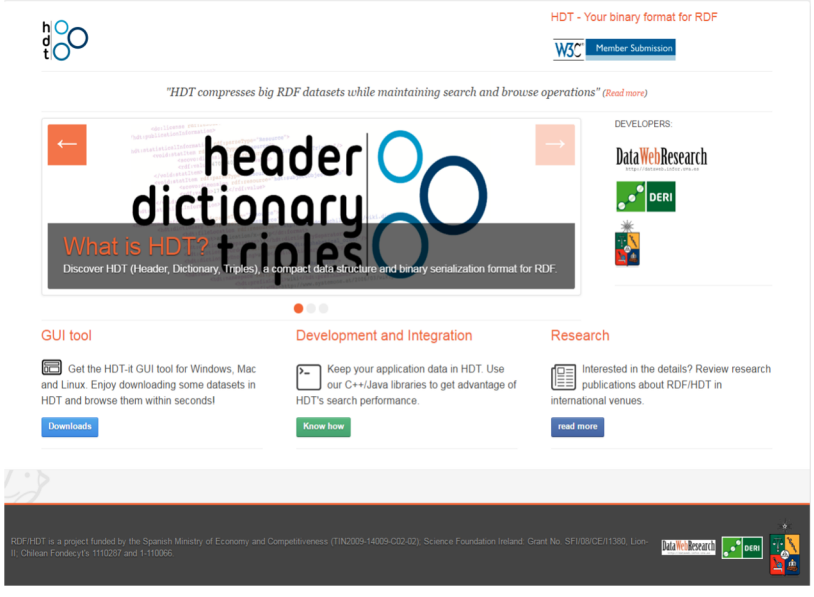


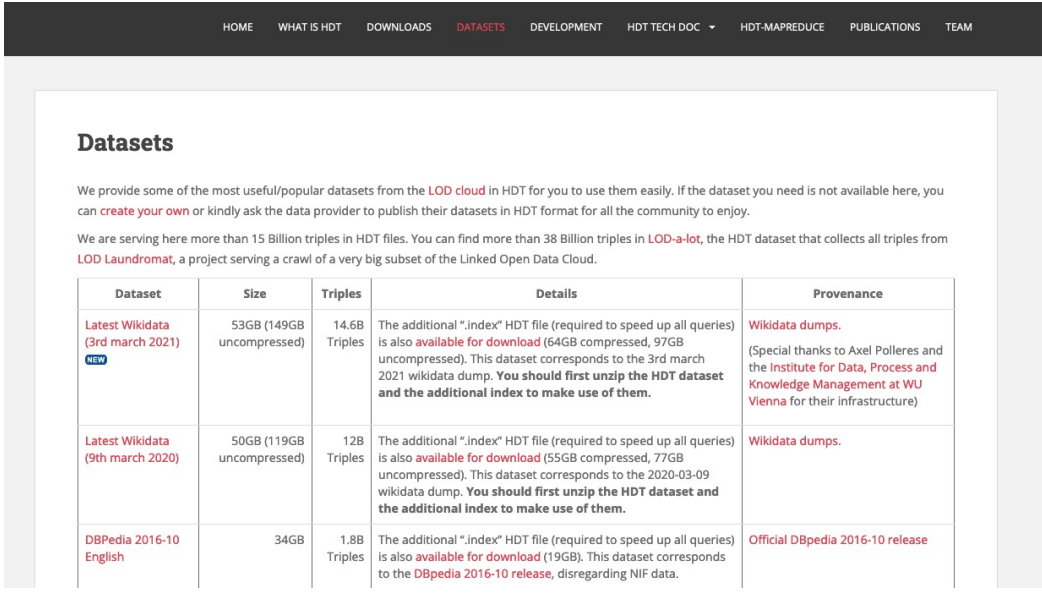
	Process	Type of Index	Patterns
1	index the bitsequences	Subject SPO	SPO, SP?, S??, S?O, ???
2	index the position of each predicate (just a position list)	Predicate PSO	?P?, ?PO
3	index the position of each object	Object OPS	??O

separate index file , created by consumer client (or published as as well)

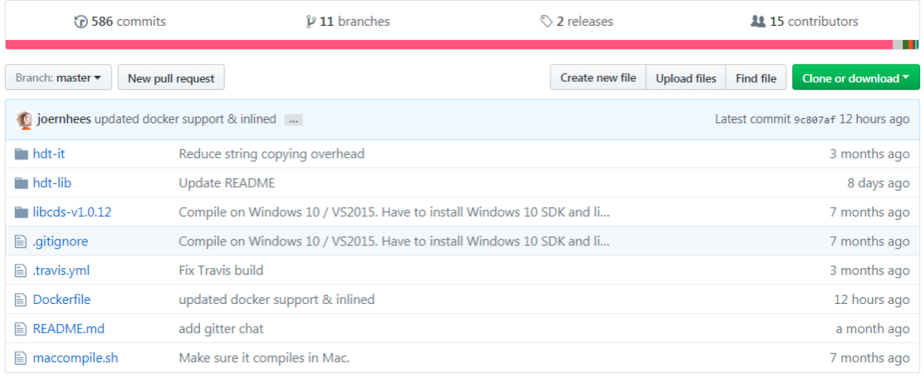
Martínez-Prieto, M., M. Arias, and J. Fernández (2012). Exchange and Consumption of Huge RDF Data. In Proc. of the 9th Extended Semantic Web Conference (ESWC), pp. 437-452.

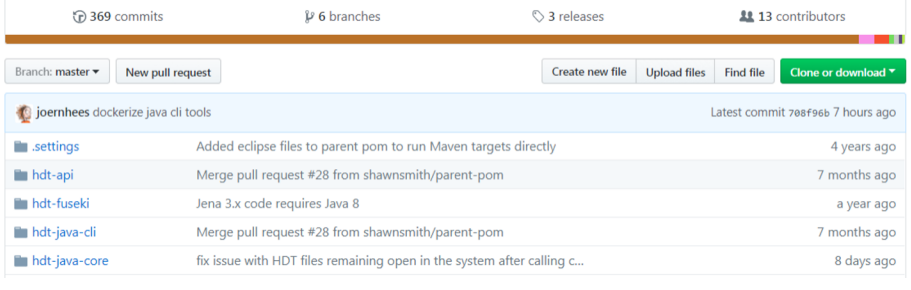
rdfhdt.org community



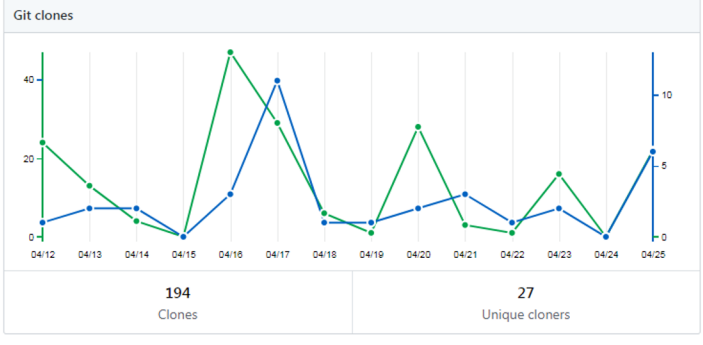


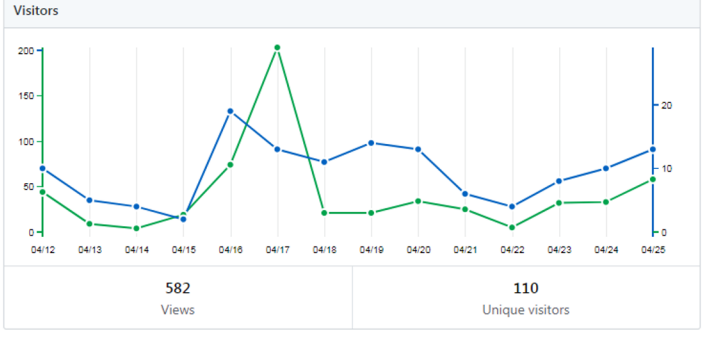
https://github.com/rdfhdt
C++ and Java tools





HDT-cpp





(+python!) https://github.com/Callidon/pyHDT/

Results

- Data is ready to be consumed 10-15x faster than loading in an RDF triple store
  - HDT << any other RDF format || RDF engine
- Competitive query performance.
  - Very fast on triple patterns, x 1.5 faster (Virtuoso, RDF3x).
- Integration with Jena
  - Joins on the same scale of existing solutions (Virtuoso, RDF3x).

Hands-on example: trying out the query that timed out on https://w.wiki/4mTj

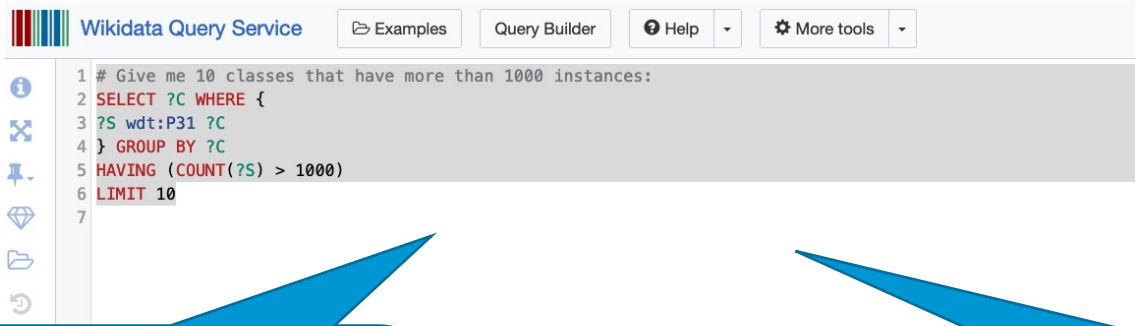
```
$ls
wikidata20210305.hdt
wikidata20210305.hdt.index.v1-1

$ time hdtsparql.sh wikidata20210305.hdt "$(<large_classes.rq)"

C
http://www.wikidata.org/entity/Q846110
http://www.wikidata.org/entity/Q69529214
http://www.wikidata.org/entity/Q1195942
http://www.wikidata.org/entity/Q55488
http://www.wikidata.org/entity/Q18691601
http://www.wikidata.org/entity/Q372363
http://www.wikidata.org/entity/Q174782
http://www.wikidata.org/entity/Q513550
http://www.wikidata.org/entity/Q88865432
http://www.wikidata.org/entity/Q22652
```

Took some ~10min on my VM (92GB RAM)

Challenge 3: Scalability of SPARQL endpoints? - It's often too expensive to host Open KGs



Challenge 3.1: serve complex/long running queries to single users

Challenge 3.2: serve many queries to many users ***concurrently***

Querying KGs with SPARQL "abstractly": What happens if you have many clients?

KG = RDF Graph

- A set of RDF triples (<Subject, Predicate, Object>)
- can be interpreted as edge-labeled directed graph.

SPARQL

- Standard query language for RDF (W3C recommendation).
- Subgraph *pattern* matching

→ **returns variable bindings (table)**

?v1	?v2	?v3
v1	v2	v4
v3	v2	v5
v3	v2	v4
v3	v2	v5

Server Solution: SPARQL Endpoint

Client:

Network:

Server:

```
SELECT ?v1 ?v2 ?v3
WHERE {  ?v1    owl:starring "Brad Pitt" .
         ?v1    rdfs:label ?v2;
         ?v1    dbo:director ?v3. }
```

SPARQL Endpoint

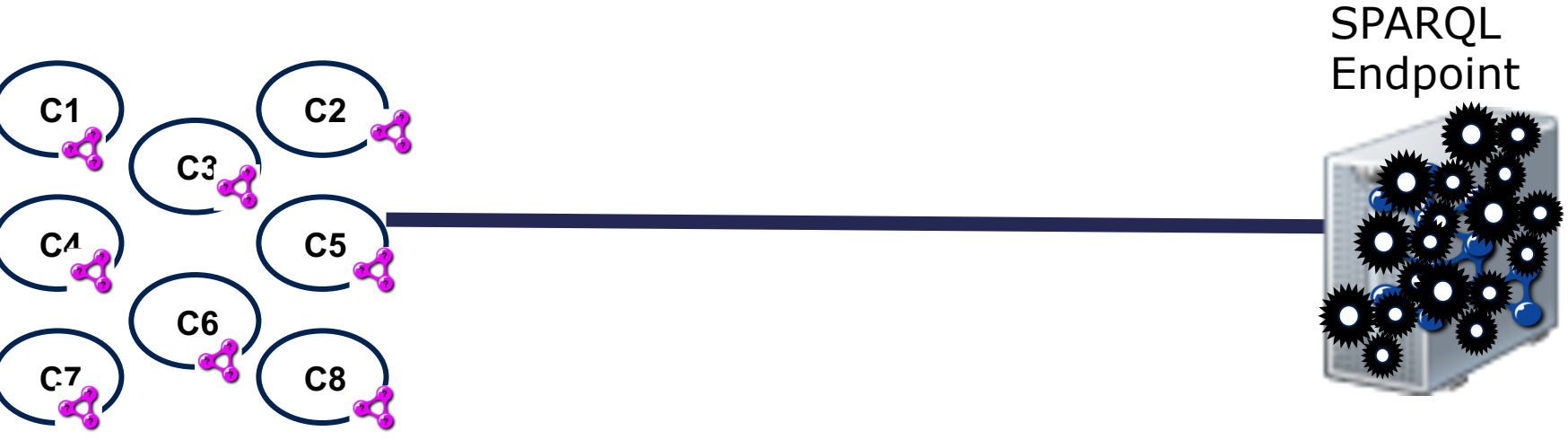
"Query Shipping"

Server Solution: SPARQL Endpoint

Client:

Network:

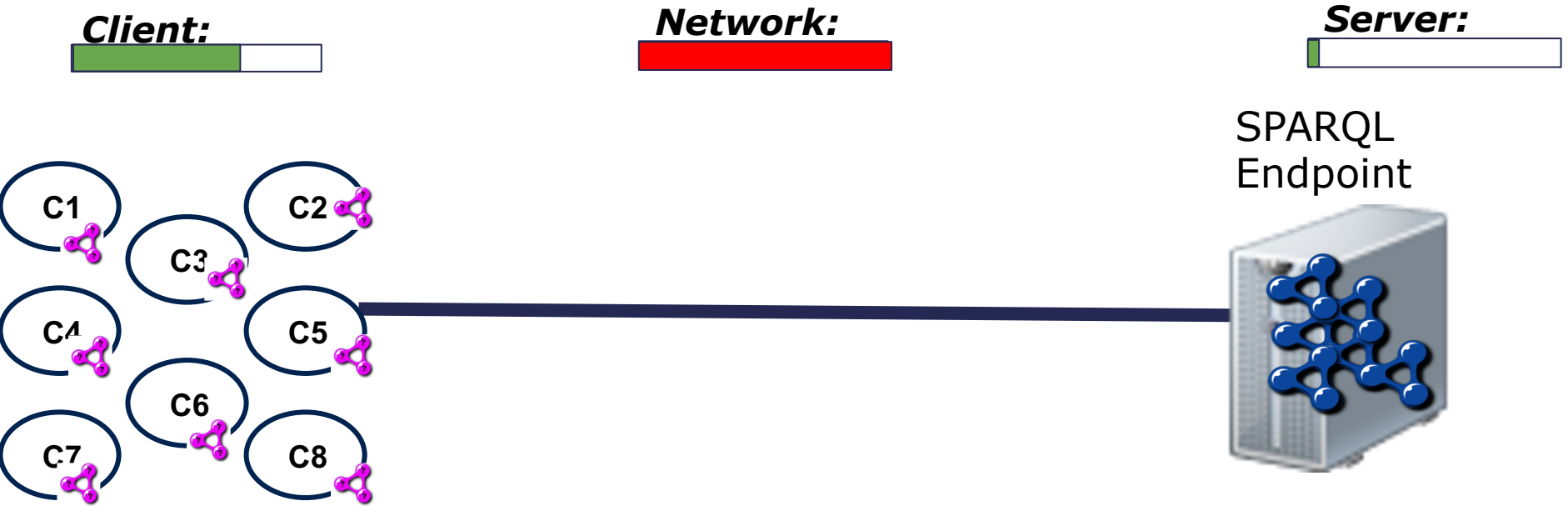
Server:



"Query Shipping"

fails under concurrency

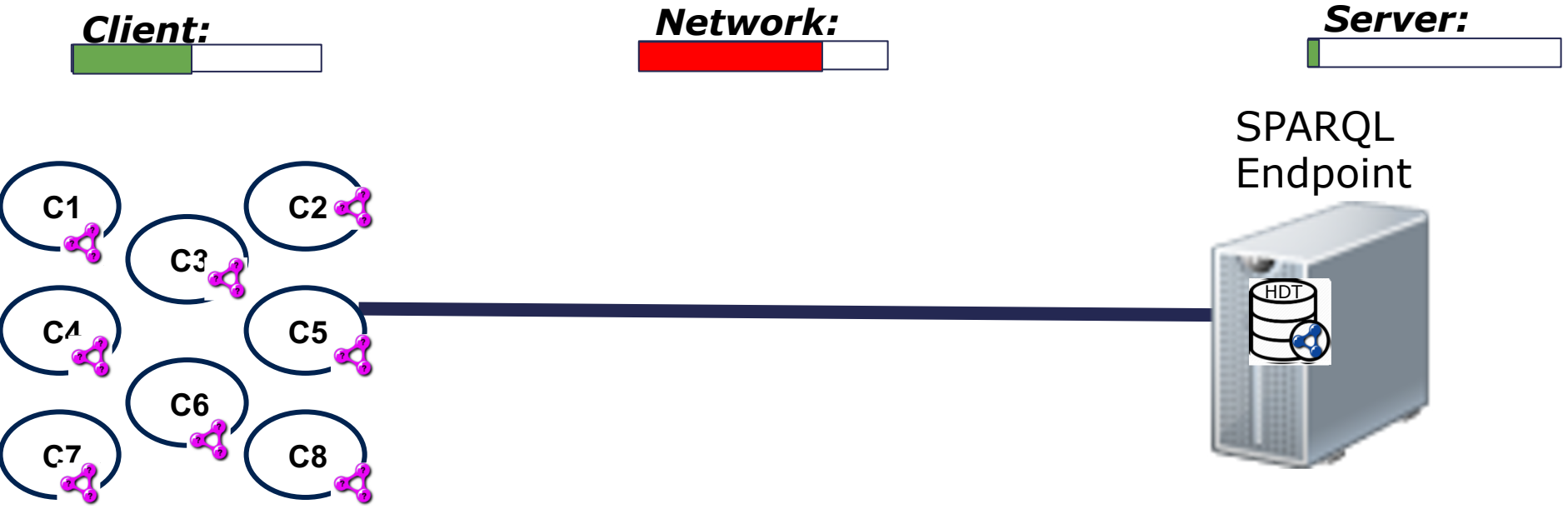
Alternative Client Solution: Data Dump



Data Shipping

might add prohibitive load on the network

Alternative Client Solution: Data Dump



Data Shipping – using HDT

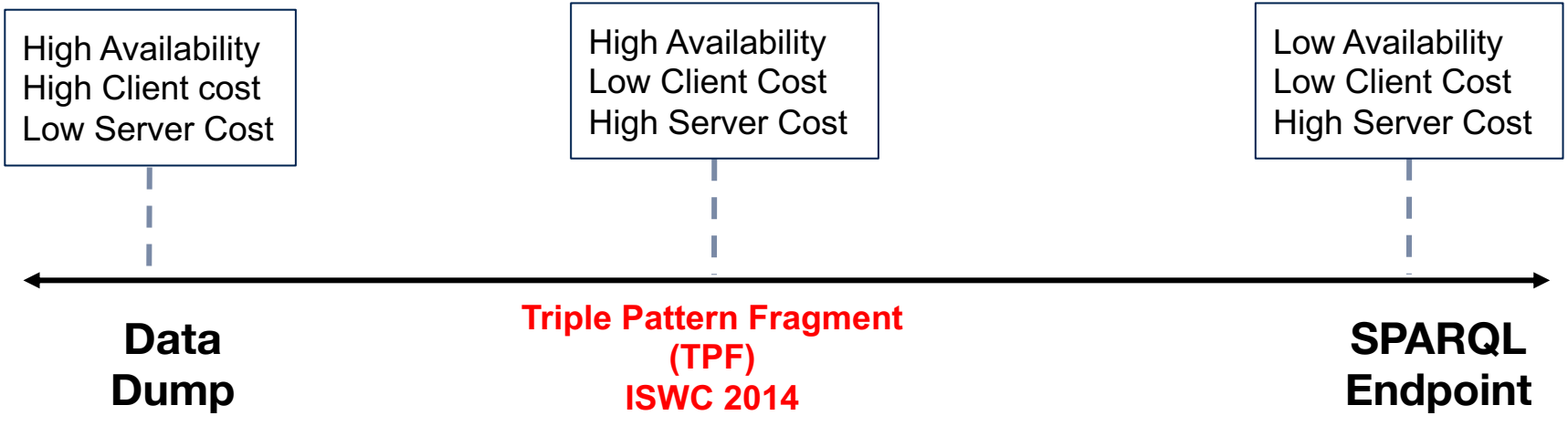
might add prohibitive load on the network

Variants in between:

What are the current mitigations to the availability problem of the open RDF KGs?

Linked Data Fragment Framework(LDF)

Proposed to design new mixes of trade-offs.



Hybrid Shipping

Triple Pattern Fragments :

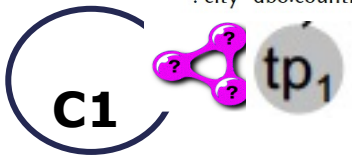
Idea:

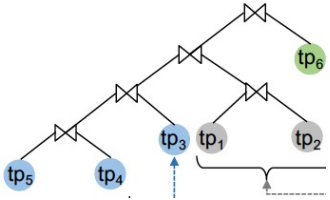
- Execute triple patterns on the server
- Let the clients do JOINs etc. by themselves.

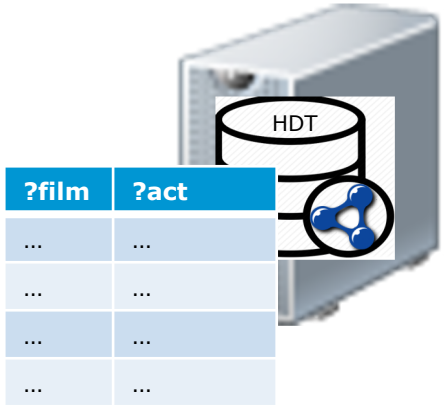
→ less footprint on the server, only triple patterns and intermediate results communicated.

→ can still have significant overhead by large intermediate results

```
SELECT * WHERE {
 ?film  dbo: starring  ? actress  .                  # tp1
 ?film  foaf : name ?filmName .                      # tp2
 ? actress  dbo:wikiPageExternalLink  ? link  .  # tp3
 ? actress  dbo: birthPlace  ? city  .               # tp4
 ? actress  foaf : gender "female"@en .              # tp5
 ? city  dbo:country  ?country  . }                  # tp6
```







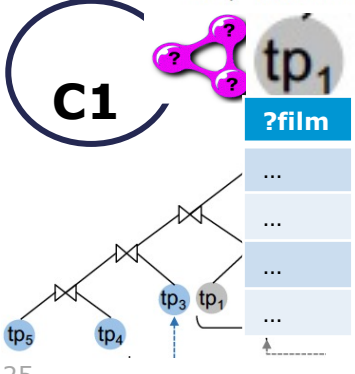
R. Verborgh, M. V. Sande, O. Hartig, J. Van Herwegen, L. De Vocht, B. De Meester, G. Haesendonck, and P. Colpaert. 2016. Triple Pattern Fragments: A low-cost knowledge graph interface for the Web. J. Web Semant. 37-38 (2016), 184–206.

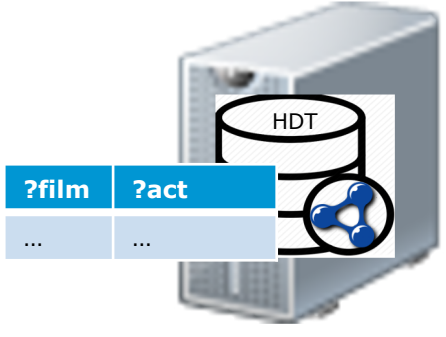
Binding-restricted Triple Pattern Fragments :

Idea:

- ship intermediate bindings with TP and let server only return results matching results
- → smaller intermediate results, "join work" distributed between client and server

```
SELECT * WHERE {
 ?film  dbo: starring  ?actress  .                  # tp1
 ?film  foaf : name ?filmName .                     # tp2
 ?actress  dbo:wikiPageExternalLink  ?link  .  # tp3
 ?actress  dbo: birthPlace  ?city  .                # tp4
 ?actress  foaf : gender "female"@en .              # tp5
 ?city  dbo:country  ?country  . }                  # tp6
```





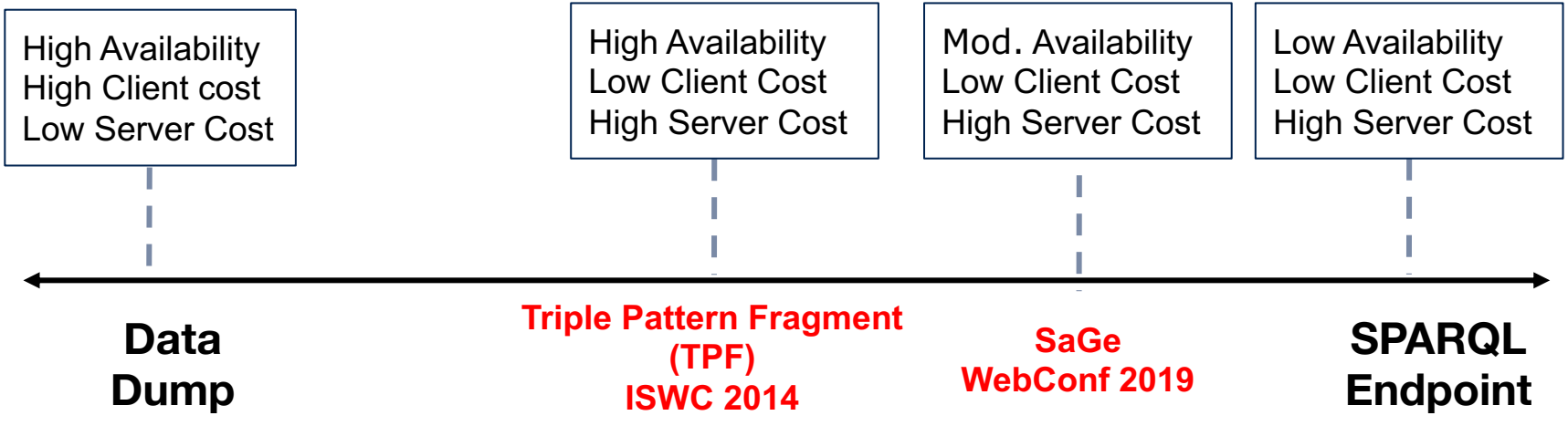
O. Hartig and C. B. Aranda. 2016. Bindings-Restricted Triple Pattern Fragments. In ODBASE 2016. 762–779

Variants in between:

What are the current mitigations to the availability problem of the open RDF KGs?

Linked Data Fragment Framework(LDF)

Proposed to design new mixes of trade-offs.



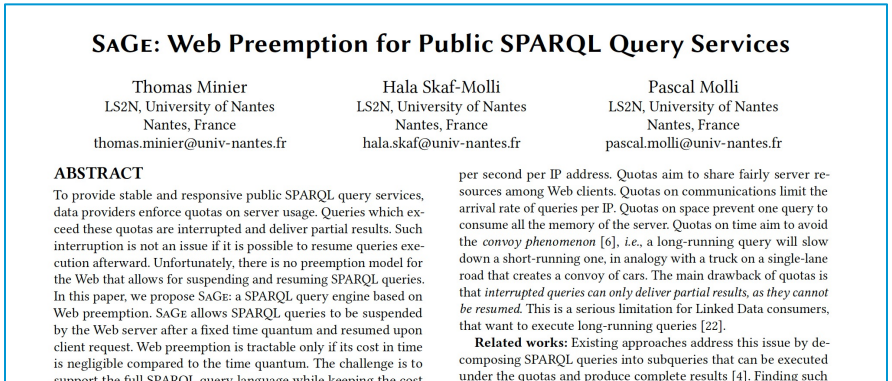
Hybrid Shipping

Slightly different approach: SaGe

Idea: keep working on the server, but improve fair allocation of resources, i.e. interrupt resource-intensive queries …

- BGP, UNION, FILTER are executed fully at the server as "interuptable iterators"
  - … can be stopped and sent back to clients with results "so far"
  - … while giving clients the possibility to resume them later on
  - server suspends running query after a fixed quantum of time and resume the next waiting query

- more complex operations are done on the client (e.g. OPTIONAL, SERVICE, ORDER BY, GROUP BY, DISTINCT, MINUS, FILTER EXIST and aggregations)



SaGe: Web Preemption for Public SPARQL Query Services

Thomas Minier
LS2N, University of Nantes
Nantes, France
thomas.minier@univ-nantes.fr

Hala Skaf-Molli
LS2N, University of Nantes
Nantes, France
hala.skaf@univ-nantes.fr

Pascal Molli
LS2N, University of Nantes
Nantes, France
pascal.molli@univ-nantes.fr

ABSTRACT

To provide stable and responsive public SPARQL query services, data providers enforce quotas on server usage. Queries which exceed these quotas are interrupted and deliver partial results. Such interruption is not an issue if it is possible to resume queries execution afterward. Unfortunately, there is no preemption model for the Web that allows for suspending and resuming SPARQL queries. In this paper, we propose SaGe: a SPARQL query engine based on Web preemption. SaGe allows SPARQL queries to be suspended by the Web server after a fixed time quantum and resumed upon client request. Web preemption is tractable only if its cost in time is negligible compared to the time quantum. The challenge is to

per second per IP address. Quotas aim to share fairly server resources among Web clients. Quotas on communications limit the arrival rate of queries per IP. Quotas on space prevent one query to consume all the memory of the server. Quotas on time aim to avoid the *convoy phenomenon* [6], *i.e.*, a long-running query will slow down a short-running one, in analogy with a truck on a single-lane road that creates a convoy of cars. The main drawback of quotas is that *interrupted queries can only deliver partial results, as they cannot be resumed.* This is a serious limitation for Linked Data consumers, that want to execute long-running queries [22].

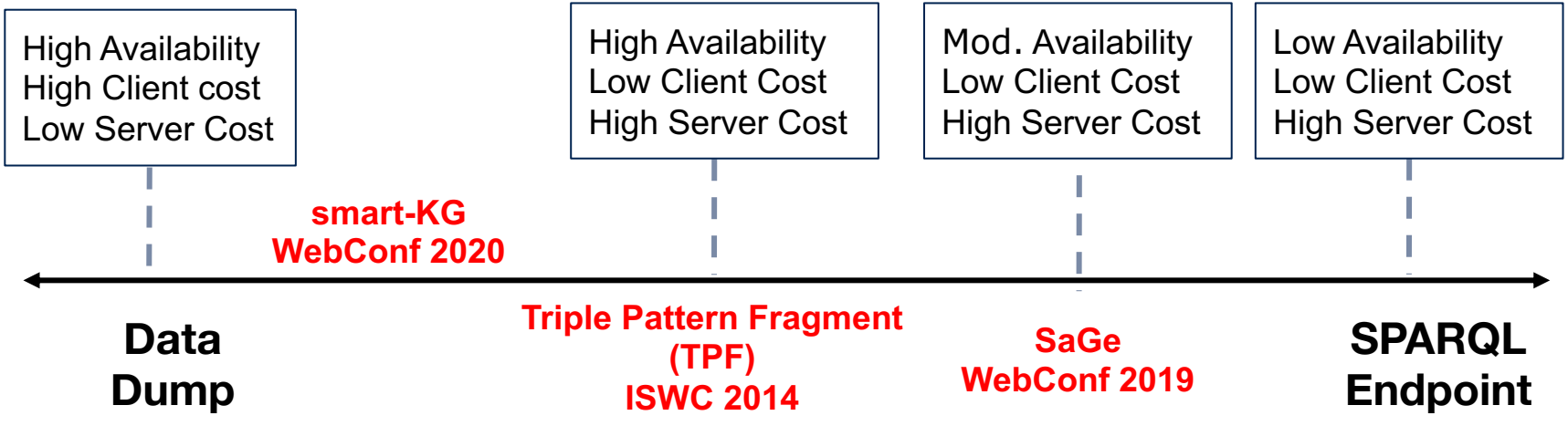
Related works: Existing approaches address this issue by decomposing SPARQL queries into subqueries that can be executed under the quotas and produce complete results [4]. Finding such

Variants in between:

What are the current mitigations to the availability problem of the open RDF KGs?

Linked Data Fragment Framework(LDF)

Proposed to design new mixes of trade-offs.

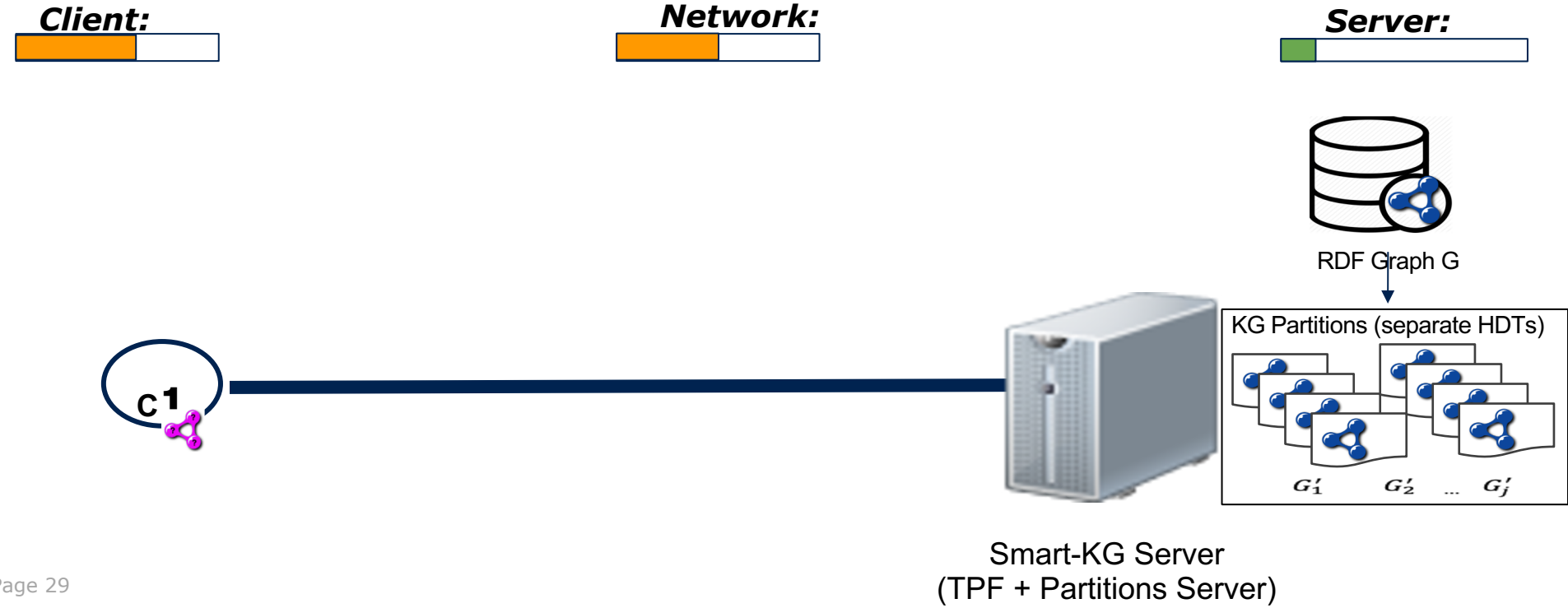


"Partition" Shipping

Smart-KG

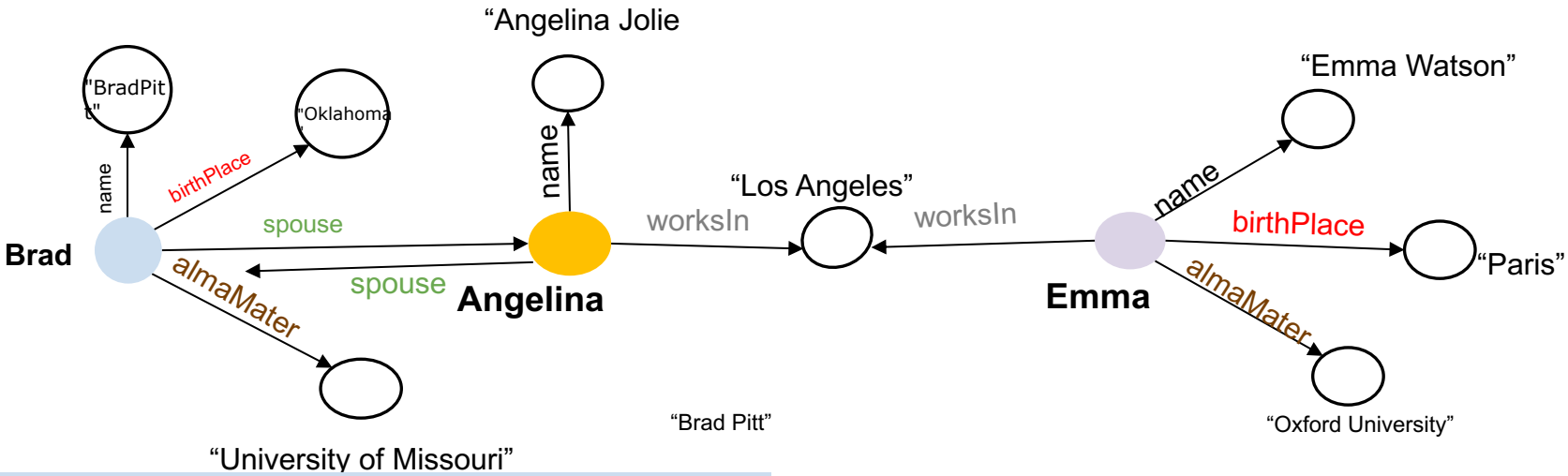
Idea:

- Partition graph by "predicate families", i.e. characteristic sets,
- create 1 HDT per family
- Combine TPF with partition shipping



smart-KG server: Family Generator

Partition Generator (PG): Upon loading a graph G, decompose it into partitions G1,...,Gm, one per "predicate family".



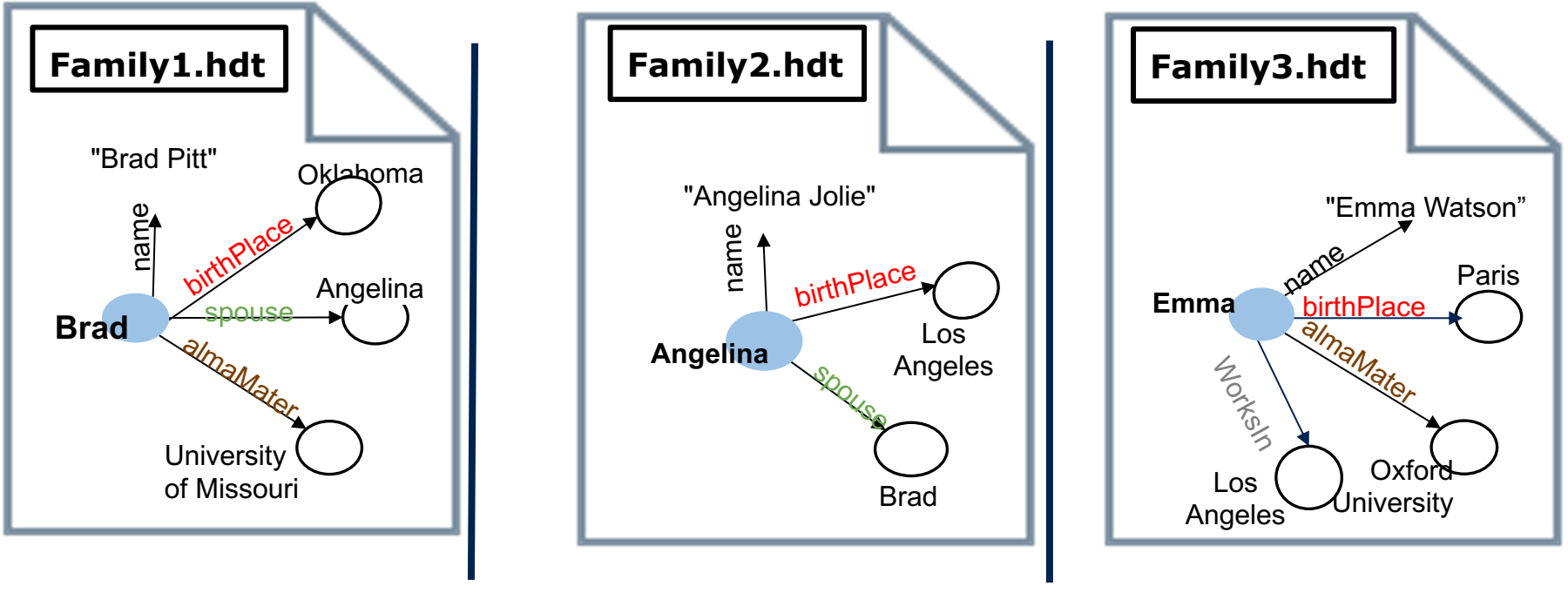
F1: {name, birthPlace, almaMater, spouse}

F2: {name, spouse,worksIn}

F3: {name, birthPlace,almaMater, worksIn}

smart-KG server: Family Generator

Partition Generator (PG): Upon loading a graph G, decompose it into partitions G1,...,Gm, one per "predicate family"… and convert these to HDTs.



Smart-KG:

Simplified client Query processing:

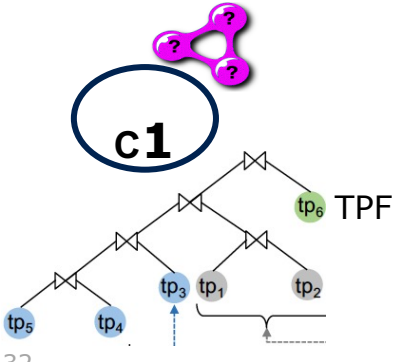
1. Client decomposes BGPs into "stars"
2. retrieves relevant information from server to make a query plan
3. retrieves and joins matching partitions one by one

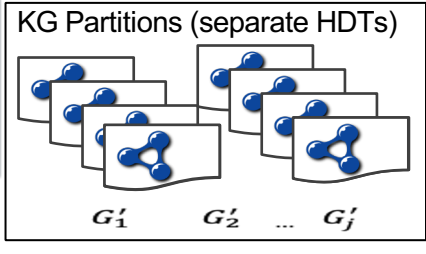
   (use TPF for 1-triple patterns)

```
SELECT * WHERE {
?film  dbo:starring  ?actress  .          # tp1
?film  foaf:name ?filmName .              # tp2
?actress  dbo:wikiPageExternalLink  ?link  .  # tp3
?actress  dbo:birthPlace  ?city  .        # tp4
?actress  foaf:gender  "female"@en .      # tp5
?city  dbo:country  ?country  . }         # tp6
```

F?:{starring, name}

F?:{wikiPageExtLink, birthPlace, gender}





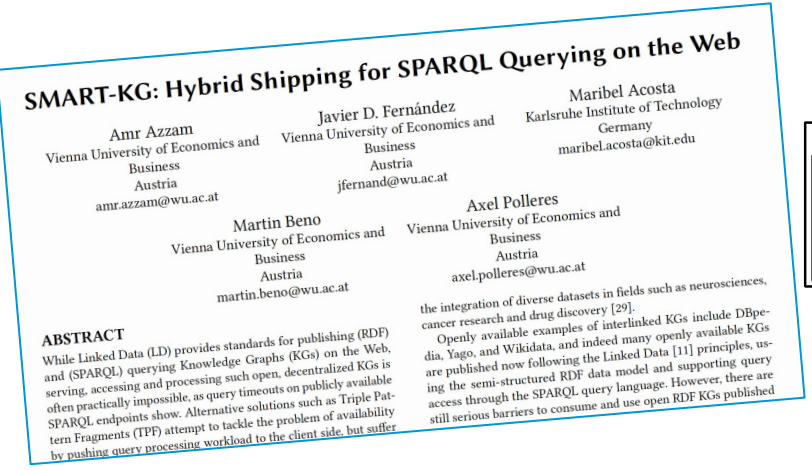
Smart-KG:

Further details, cf. our paper:

- predicate-restricted families, i.e. pruning
  - too rare or
  - too common

  predicates for partitioning.

  Example: for DBpedia, a naive partitioning would create +600k partially very large families, which are unfeasible to serve.
- partition caching



SMART-KG: Hybrid Shipping for SPARQL Querying on the Web

Amr Azzam
Vienna University of Economics and Business
Austria
amr.azzam@wu.ac.at

Javier D. Fernández
Vienna University of Economics and Business
Austria
jfernand@wu.ac.at

Maribel Acosta
Karlsruhe Institute of Technology
Germany
maribel.acosta@kit.edu

Martin Beno
Vienna University of Economics and Business
Austria
martin.beno@wu.ac.at

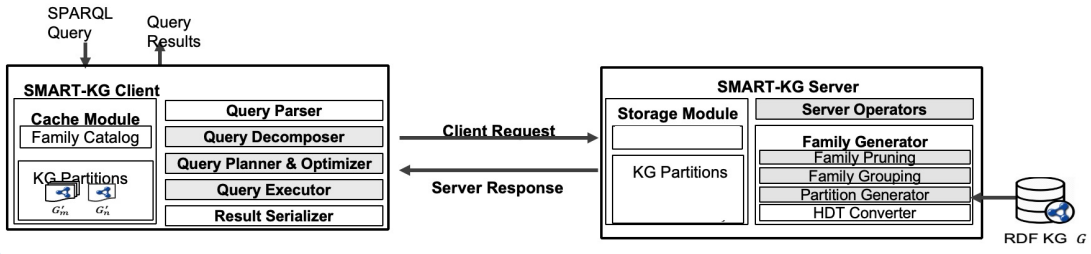
Axel Polleres
Vienna University of Economics and Business
Austria
axel.polleres@wu.ac.at

ABSTRACT

While Linked Data (LD) provides standards for publishing (RDF) and (SPARQL) querying Knowledge Graphs (KGs) on the Web, serving, accessing and processing such open, decentralized KGs is often practically impossible, as query timeouts on publicly available SPARQL endpoints show. Alternative solutions such as Triple Pattern Fragments (TPF) attempt to tackle the problem of availability by pushing query processing workload to the client side, but suffer

the integration of diverse datasets in fields such as neurosciences, cancer research and drug discovery [29].

Openly available examples of interlinked KGs include DBpedia, Yago, and Wikidata, and indeed many openly available KGs are published now following the Linked Data [11] principles, using the semi-structured RDF data model and supporting query access through the SPARQL query language. However, there are still serious barriers to consume and use open RDF KGs published



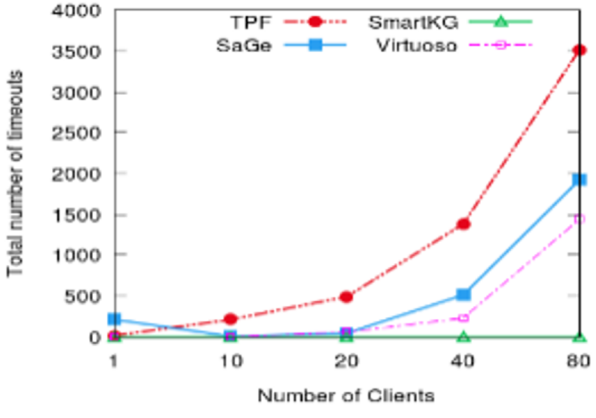
Experiments:

server, 384 GB RAM
up to 80 clients, 32 RAM
1 GBit/s network (limited to 20Mbit/s per client

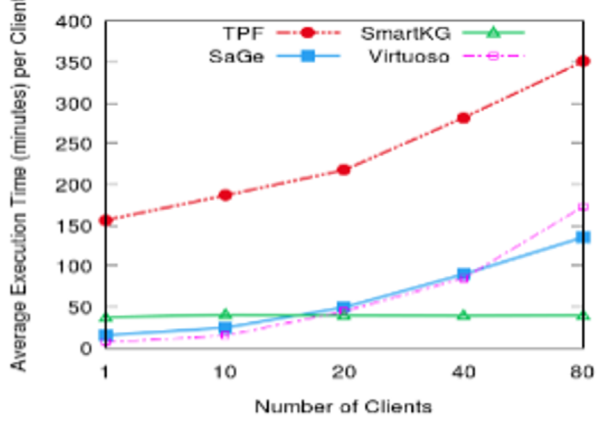
- WatDiv up to 1B triples, up to 10joins
- DBpedia, 12 random BPGs from LSQ

Overall Query Performance

Increasing Number of Clients



(a) Number of timeouts

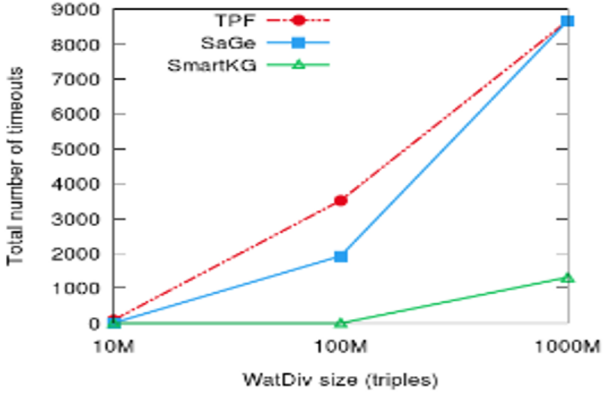


(b) Average execution time

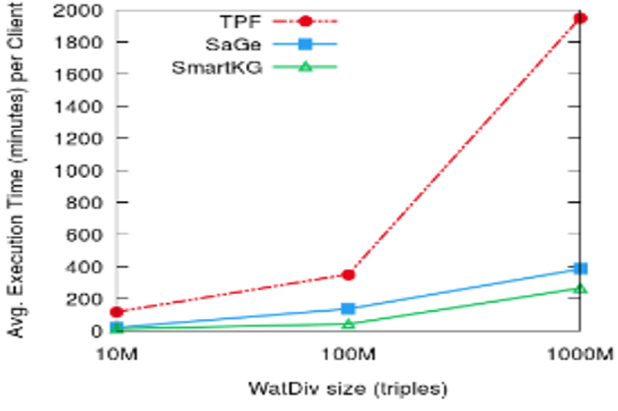
100M Watdiv

Overall Query Performance

Increasing KG size



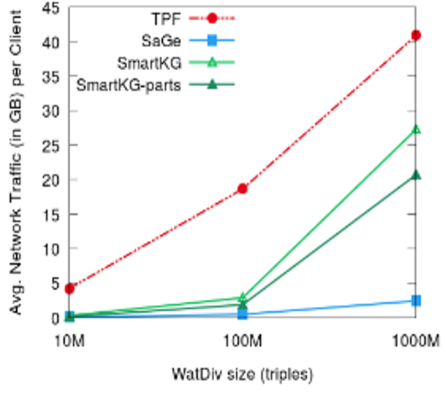
(a) Number of Timeouts



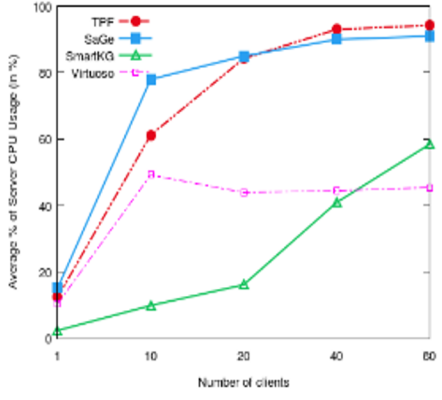
(b) Average Workload Execution Time

Resources Consumption

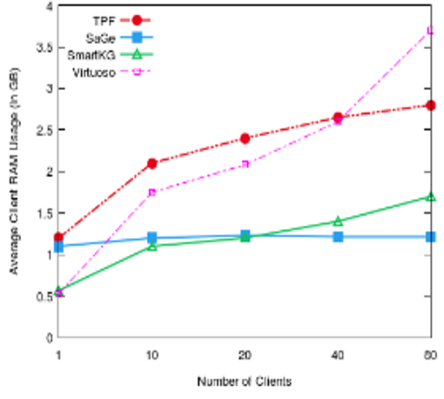
Server Network & CPU & RAM



(a) Network traffic per client (in GB) on the intensive workload at increasing KG sizes



(b) Avg. Server CPU Usage (in %) at increasing number of clients (WatDiv-100M)



(c) Avg. Server RAM Usage (in GB) at increasing number of clients (WatDiv-100M)

Next Step/Extension:

- execute star-patterns directly on the server (resources allowed)
  - … using an extension of TPF called SPF…

or on the client using SmartKG…

- … based on comparing COST models for client and server execution, taking into account current server load:

$$cost(sp) = \underbrace{W_{CPU} \times (\#CPU)}_{\text{Processing}} + \underbrace{W_{MSG} \times (\#M)}_{\text{Messaging}} + \underbrace{W_{BYT} \times (\#BYT)}_{\text{Data transfer}} + \underbrace{W_{IO} \times (\#IO)}_{\text{I/O}}$$

WiseKG (WebConf 2021)

WiseKG: Balanced Access to Web Knowledge Graphs

Amr Azzam*
Vienna Univ. of Economics & Business
amr.azzam@wu.ac.at

Christian Aebeloe*
Aalborg University
caebel@cs.aau.dk

Gabriela Montoya
Aalborg University
gmontoya@cs.aau.dk

Ilkcan Keles
Aalborg University
Turkcell
ilkcan.keles@turkcell.com.tr

Axel Polleres
Vienna Univ. of Economics & Business
Complexity Science Hub Vienna
axel.polleres@wu.ac.at

Katja Hose
Aalborg University
khose@cs.aau.dk

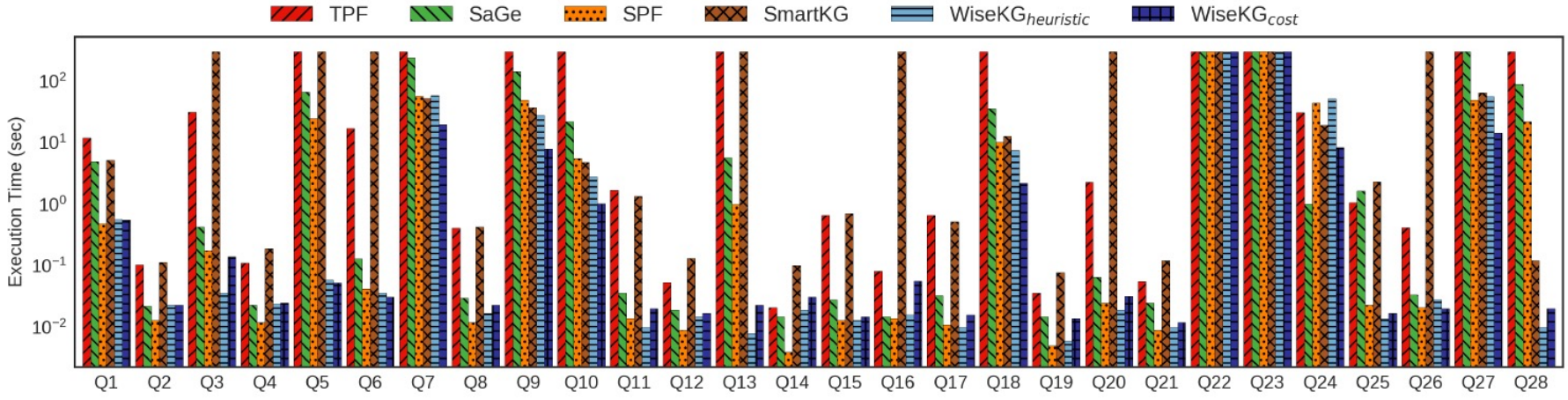
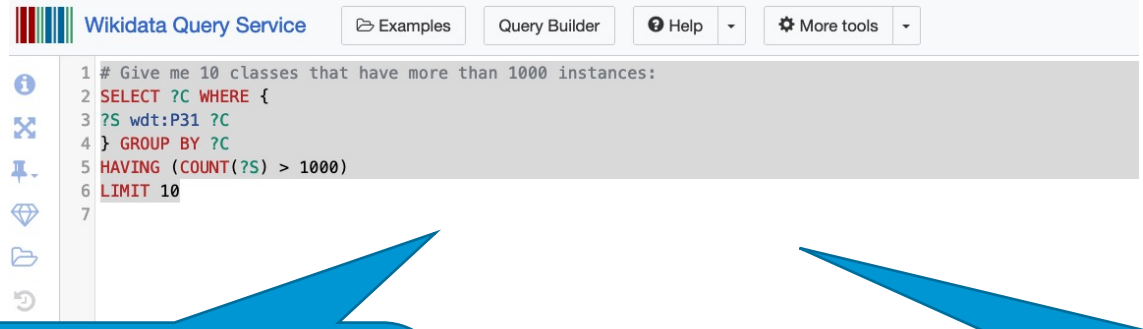


Figure 3: Execution time (in seconds) for 28 diverse queries over the dbpedia dataset.

Challenge 3: Scalability of SPARQL endpoints? - It's often too expensive to host Open KGs



Challenge 3.1: serve complex/long running queries to single users

Challenge 3.2: serve many queries to many users ***concurrently***

Summary: HDT helps ☺
more ideas for improvements, e.g.:
- peer-to-peer?
- optimize partitioning (based onquery logs?)

TODO:

I owe you a full list of references, will be added shortly! ☺