

Big Linked Data ETL Benchmark on Cloud Commodity Hardware

iMinds – Ghent University

Dieter De Witte, Laurens De Vocht,
Ruben Verborgh, Erik Mannens, Rik Van de Walle

Ontoforce

Kenny Knecht, Filip Pattyn, Hans Constandt

Introduction

Approach

Benchmark

Results


Conclusions & Next Steps

Introduction

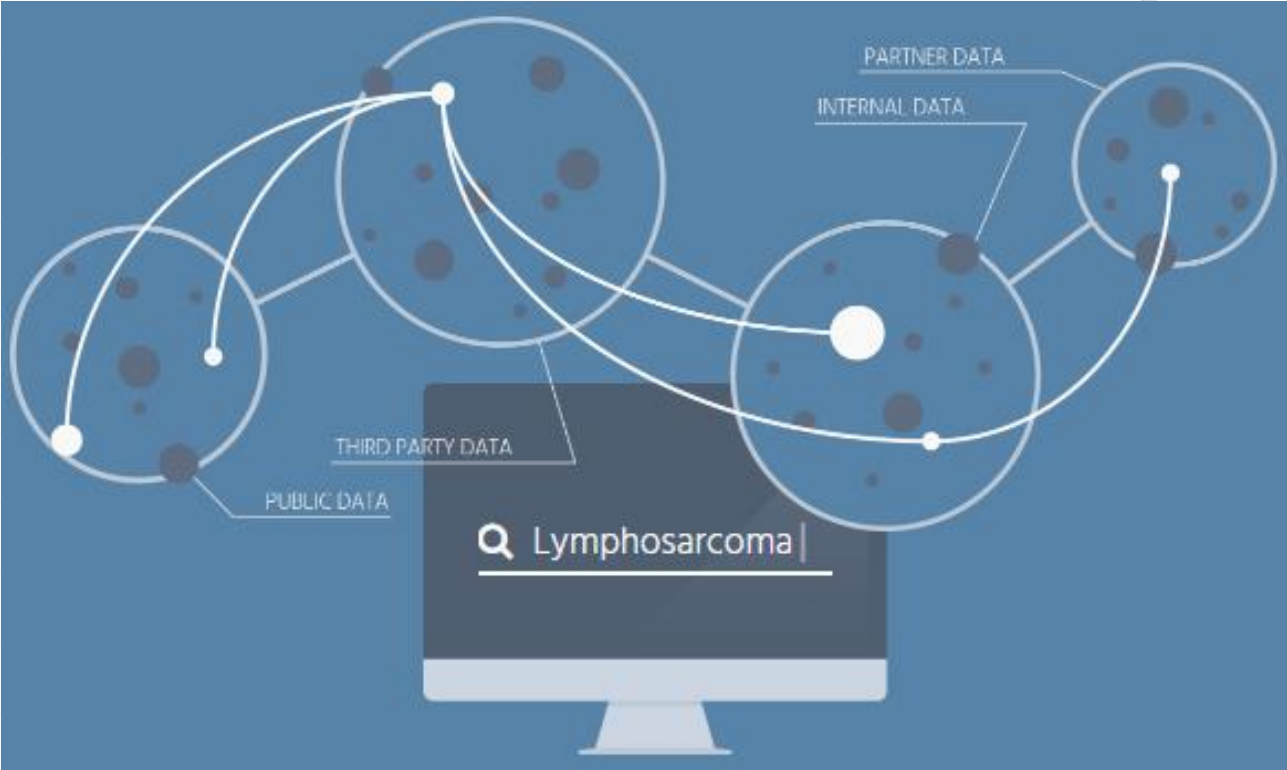
- Facilitate development of semantic federated query engine
close the (semantic) analytics gap in life sciences.
- The query engine drives an exploratory search application: **DisQover**
- Approach to federated querying by implementing ETL pipeline
indexes the user views in advance.
- Combine Linked Open Data with private and licensed (proprietary) data
 - ➔ discovery of biomedical data
 - ➔ new insights in medicine development.



DisQover: which data?

 **Datasets (113)**

Title	
UNII	
PubMed	
Provenance	
EPO	
DailyMed	
ClinicalTrials.gov	
ChEBI	
WHO ICTRP	
Taxonomy	
RxNorm	6/17/2016
Reactome	6/17/2016
Orphanet Rare Disease Ontology	6/17/2016
ONTOFORCE	6/17/2016
National Drug Code	6/17/2016
MedDRA	6/17/2016



The diagram illustrates the integration of various data sources for a search query. A central computer monitor displays the search term "Lymphosarcoma". Four overlapping circles represent different data sources: "PUBLIC DATA", "THIRD PARTY DATA", "INTERNAL DATA", and "PARTNER DATA". Lines connect these sources to the search bar, indicating data integration.

Challenges

- Ensure minimal knowledge about data linking or annotation is required to explore and find results.
- Write SPARQL directly
 - detailed knowledge of the predicates is required
 - might require first exploring to determine the URIs.
- Scaling out to more data
- Search queries are complex because search spans two distinct domains:
 1. the 'space' of clinical studies;
 2. 'drugs/chemicals'.

Introduction

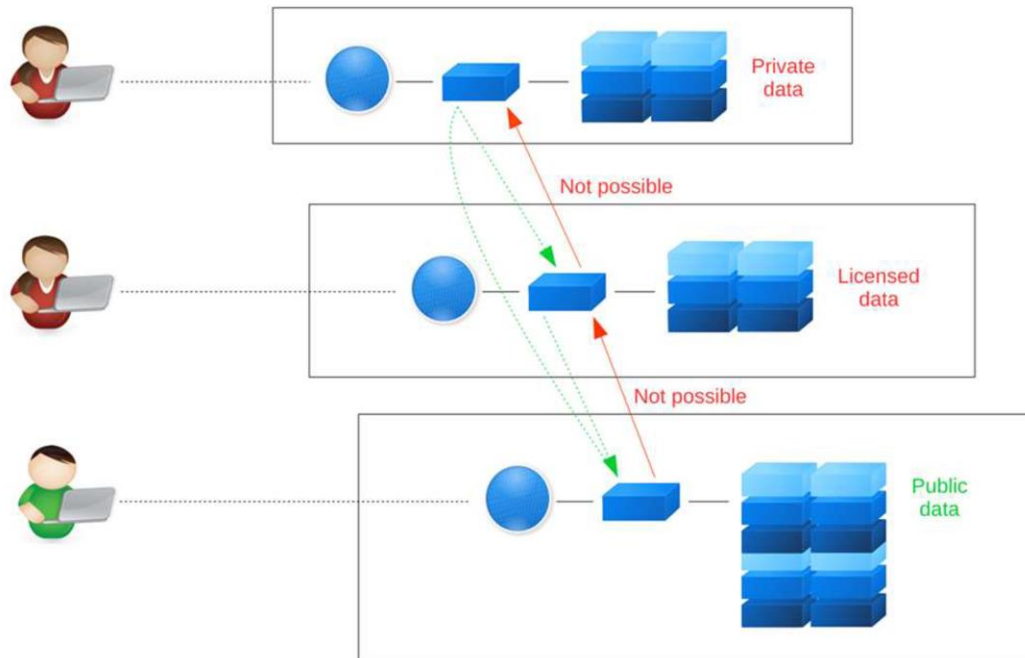
Approach

Benchmark

Results

Conclusions & Next Steps

Approach



How to do federated search with minimal latency for end-user?

Which RDF stores support the infrastructure?

What aspects should the design of a reusable benchmark take into account?

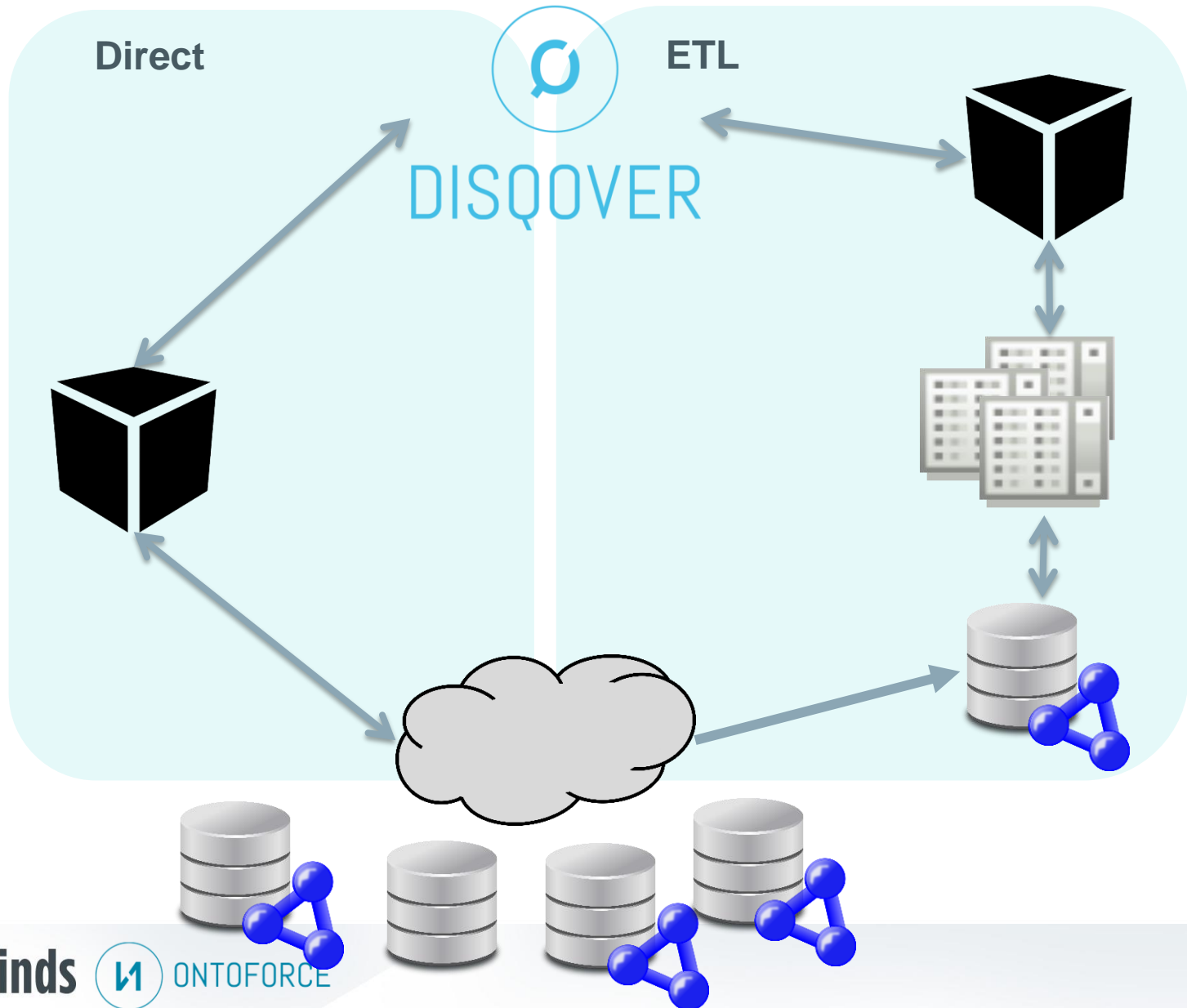
Scaling out: techniques

The *scaling-out* approach relies on low-end commodity hardware but uses many nodes in a distributed system:

1. **Specialized scalable RDF stores, the focus of this work;**
2. Translating SPARQL and RDF to existing NoSQL stores;
3. Translating SPARQL and RDF to existing Big Data approaches such as MapReduce, Impala, Apache Spark;
4. Distributing the data in physically separated SPARQL endpoints over the Semantic Web, using federated querying techniques to resolve complex questions.

Note: Compression (in-memory) is an alternative for distribution. RDF datasets can be compressed (e.g. “Header Dictionary Triples” – HDT).

ETL in instead of direct querying



Why?

- Typical DisQover queries introduce much query latency when directly federated.
- Facets consist of multiple separate SPARQL queries and serve both as filter and as dashboard.
- Data integration in DisQover:
Facets filter across all data originating from multiple different sources.

Introduction

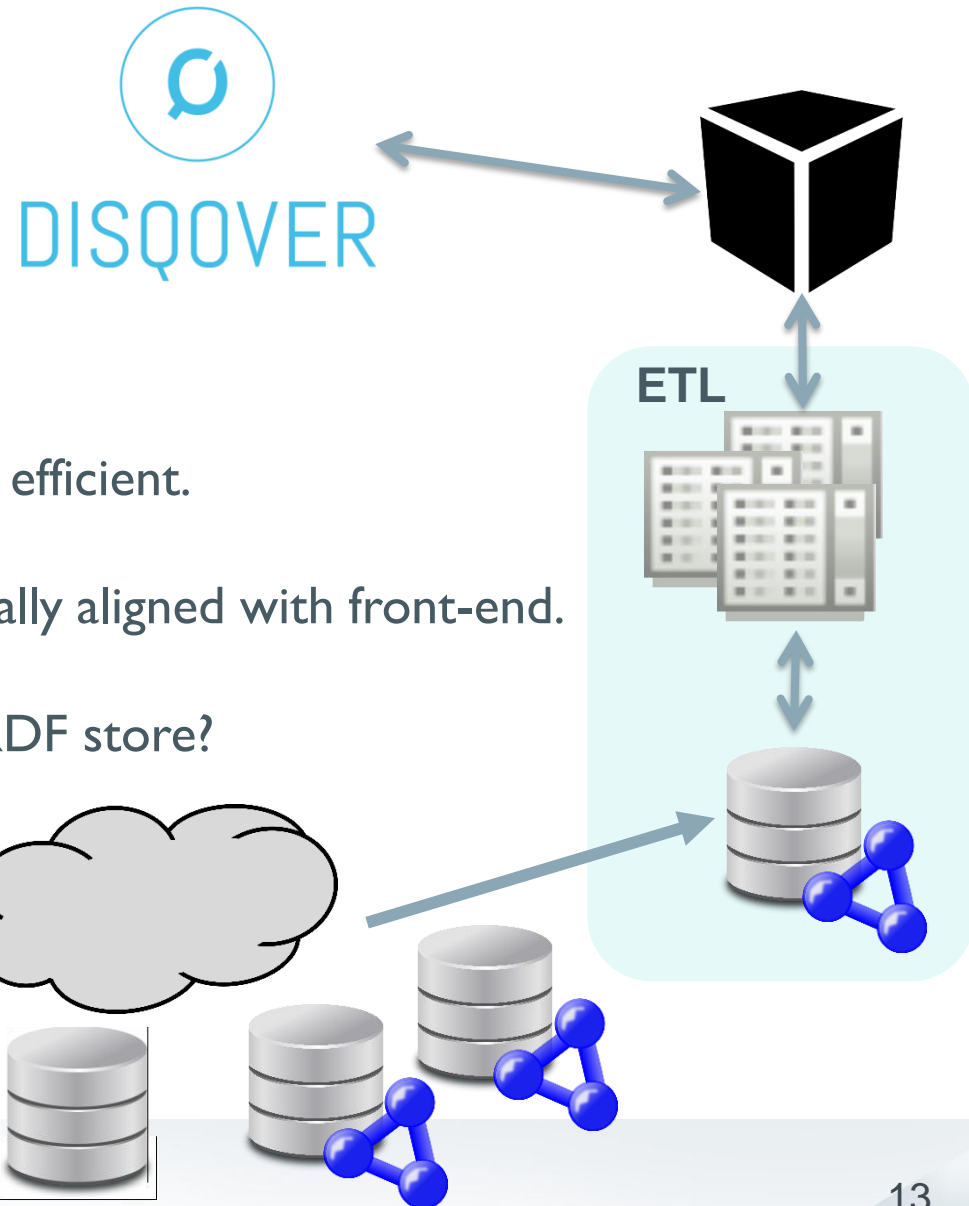
Approach

Benchmark

Results

Conclusions & Next Steps

Benchmark



Design of benchmark focus:

- ETL part needs to be optimally cost efficient.
- SPARQL queries for indexes maximally aligned with front-end.
- What is are the tradeoffs for each RDF store?

Questions the benchmark answers

- What is the most cost-effective storage solution to support Linked Data applications that need to be able to deal with heavy ETL query workloads?
- Which performance trade-offs do storage solutions offer in terms of scalability?
- What is the impact of different query types (templates)?
- Is there a difference in performance between the stores based on the structural properties of the queries?

Note: not taken into account implicitly derived facts, inference or reasoning.

Data and Query Generation

WatDiv provides stress testing tools for SPARQL

➔ existing benchmarks not always suitable for testing systems in diverse queries and varied workloads:

- generic benchmark + not application specific;
 - covers a broad spectrum
 - result cardinality
 - triple-pattern selectivity
- ensured through the data and query generation method;
- Benchmark is repeatable with different dataset sizes or numbers of queries.

RDF Store Selection

The RDF store should be capable of serving in a production environment with Linked Data in Life Sciences.

The initial selection was made by choosing stores with:

- a high adoption/popularity as defined by DB-Engines.com ranking for RDF stores;
- enterprise support;
- support for distributed deployment;
- full SPARQL 1.1 compliance.

The four stores we selected all comply with these constraints.

Note: The names of two stores we tested could not be disclosed.

They are being referred to as Enterprise Store I and II (ESI and ESII)

Process

The benchmark process consists of a data loading phase, followed by running the SPARQL benchmarker:

1. The data is loaded in compressed format (gzip).
2. The benchmarker runs in multi-threaded mode (8 threads), runs a set of 2000 queries multiple times.
3. These runs consists of at least one warm-up run which is not counted.
4. In order to obtain robust results the tail results (most extreme) are discarded before calculating average query runtimes.
5. The benchmarker generates a CSV file containing the run times and response times etc. of all queries which we visualized.

Infrastructure

Query Driver

“SPARQL Query Benchmark” is a general purpose API and CLI that is designed primarily for testing remote SPARQL servers.

By default operations are run in a random order to avoid the system under test (SUT) learning the pattern of operations.

Hardware

Executed all benchmarks on the Amazon Web Services (AWS) Elastic Compute Cloud (EC2) and Simple Storage Solutions (S3).

Used the default (commercial) deployments of the SUT for the results to be reproducible:

- both the hardware and the machine images can be easily acquired.
- more generally, cloud deployments offer the advantage of not requiring dedicated on-premises hardware.

Introduction

Approach

Benchmark

Results

Conclusions & Next Steps

Results

Cost

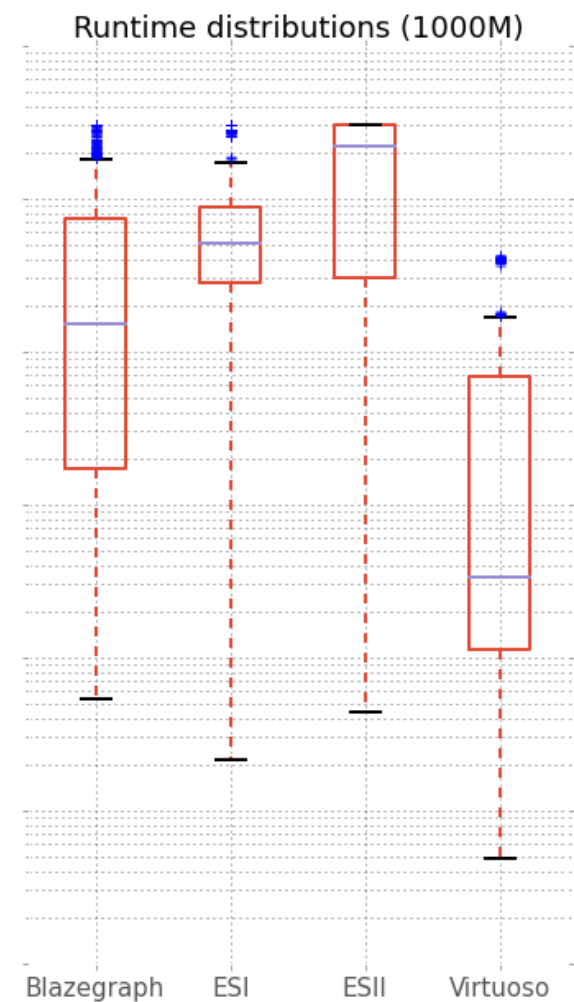
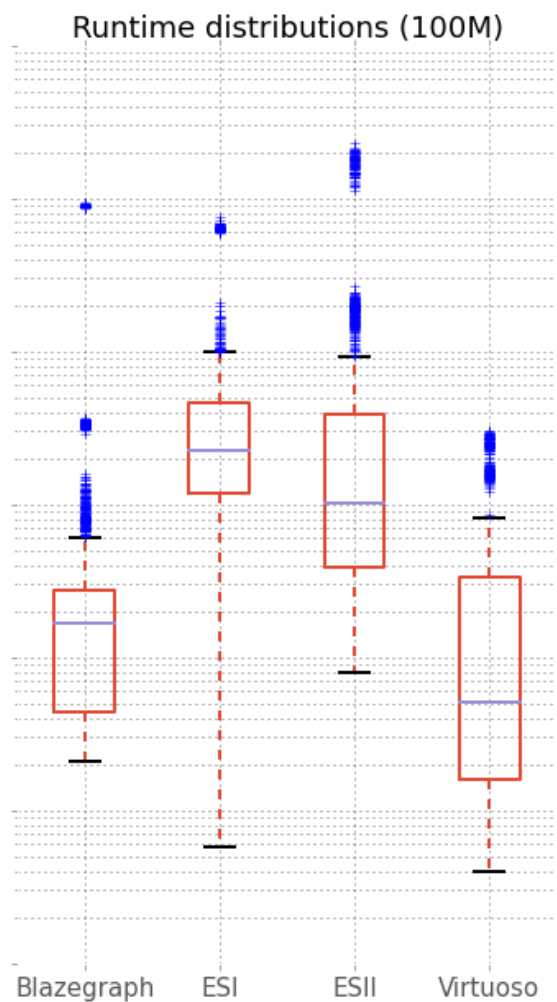
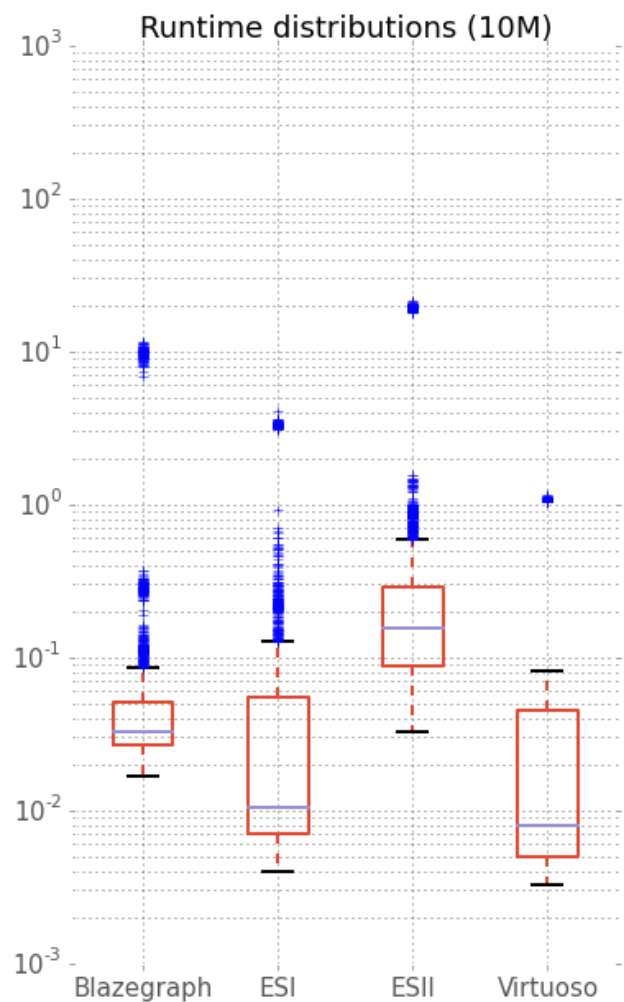
Scalability

Behavior (Different Query Types)

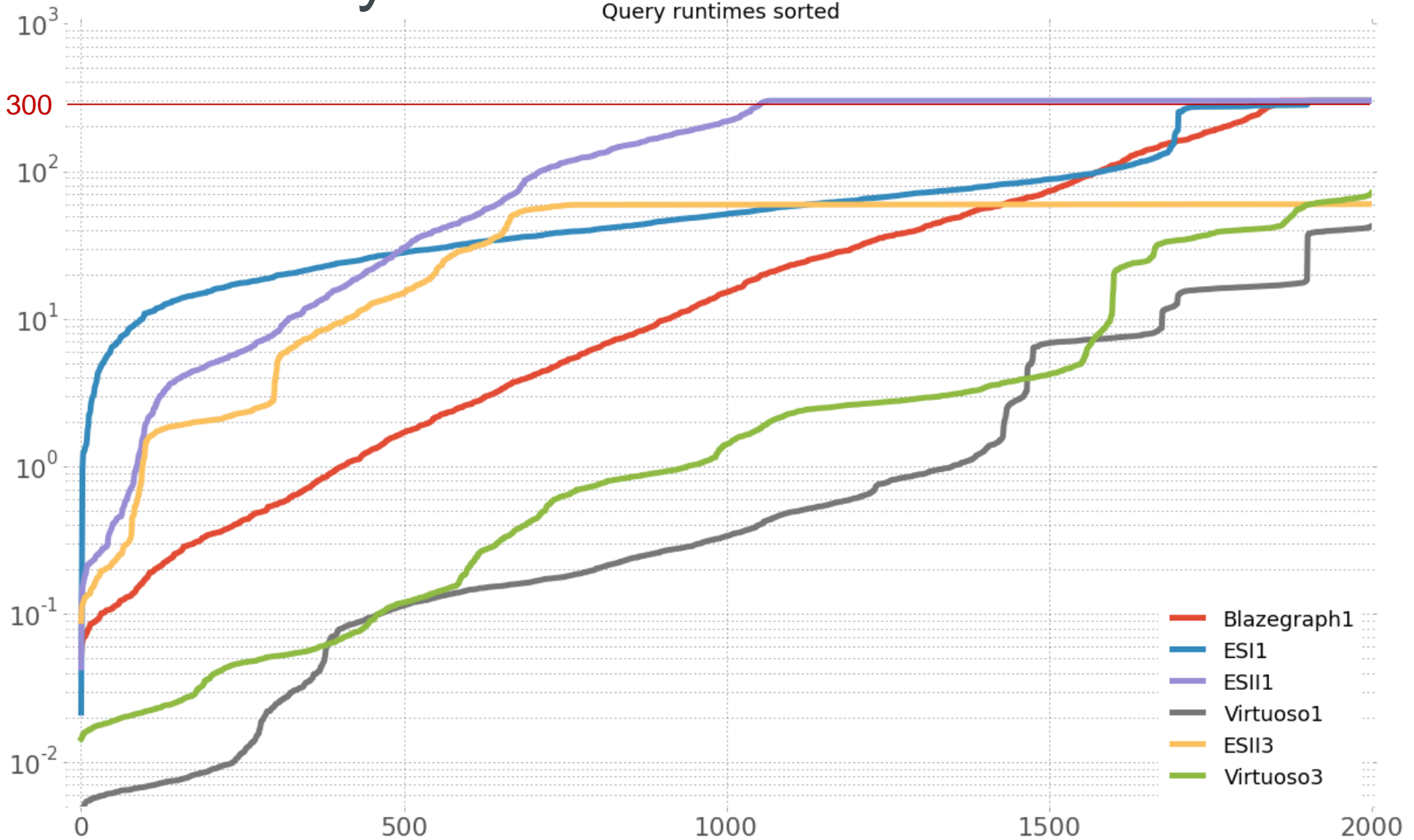
Errors and Time-outs

Store	Dataset size (million triples)	Load Time (s)	Median Runtime(s)	Median Responsetime (s)	Instance cost (\$/Hr)	Load cost (\$)	Run cost (\$)	Total cost (\$)
Blazegraph 2.0.0	10	246	1,578	142	0.33	0.02	0.15	0.17
	100	5,784	13,343	754	0.33	0.54	1.23	1.77
	1000	181,362	141,897	83,442	0.33	16.78	13.13	29.9
Enterprise Store I	10	641	1,069	721	0.33	0.06	0.1	0.51
	100	10,457	29,776	22,832	0.33	0.97	2.75	12.1
	1000	168,780	285,350	262,672	0.33	15.61	26.39	136.62
Enterprise Store II	10	601	3,242	3,047	0.33	0.06	0.3	1.41
	100	5,498	33,524	33,522	0.33	0.51	3.1	14.34
	1000	58,912	357,478	357,460	0.33	5.45	33.07	153.02
Virtuoso 7.2	10	68	152	138	0.33	0.01	0.01	0.06
	100	1,290	797	778	0.33	0.12	0.07	0.57
	1000	13,940	9,802	9,629	0.33	1.29	0.91	6.48
Enterprise Store II (3 nodes)	1000	56,915	86,757	86,754	1	15.79	24.08	158.4
Virtuoso 7.2(3 nodes)	1000	54,850	20,570	20,392	1	15.22	5.71	61.78

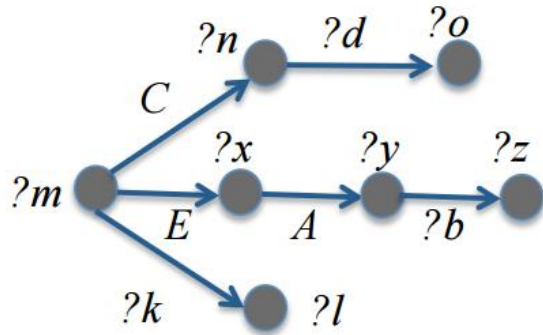
Scalability: 0.01 B – 0.1 B – 1 B



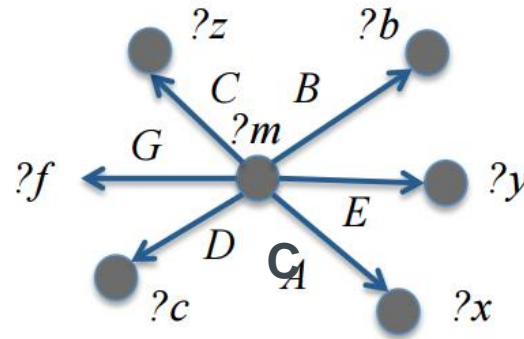
Scalability: 1B



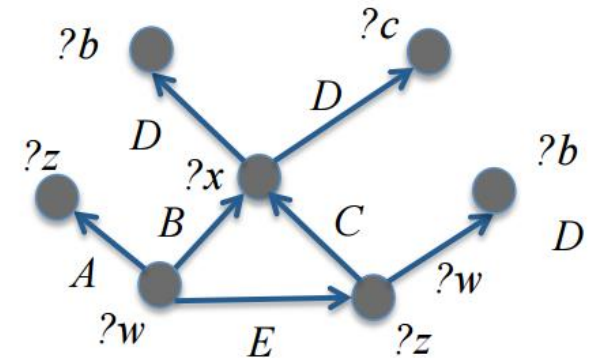
Behavior: different query types



Linear query **L**



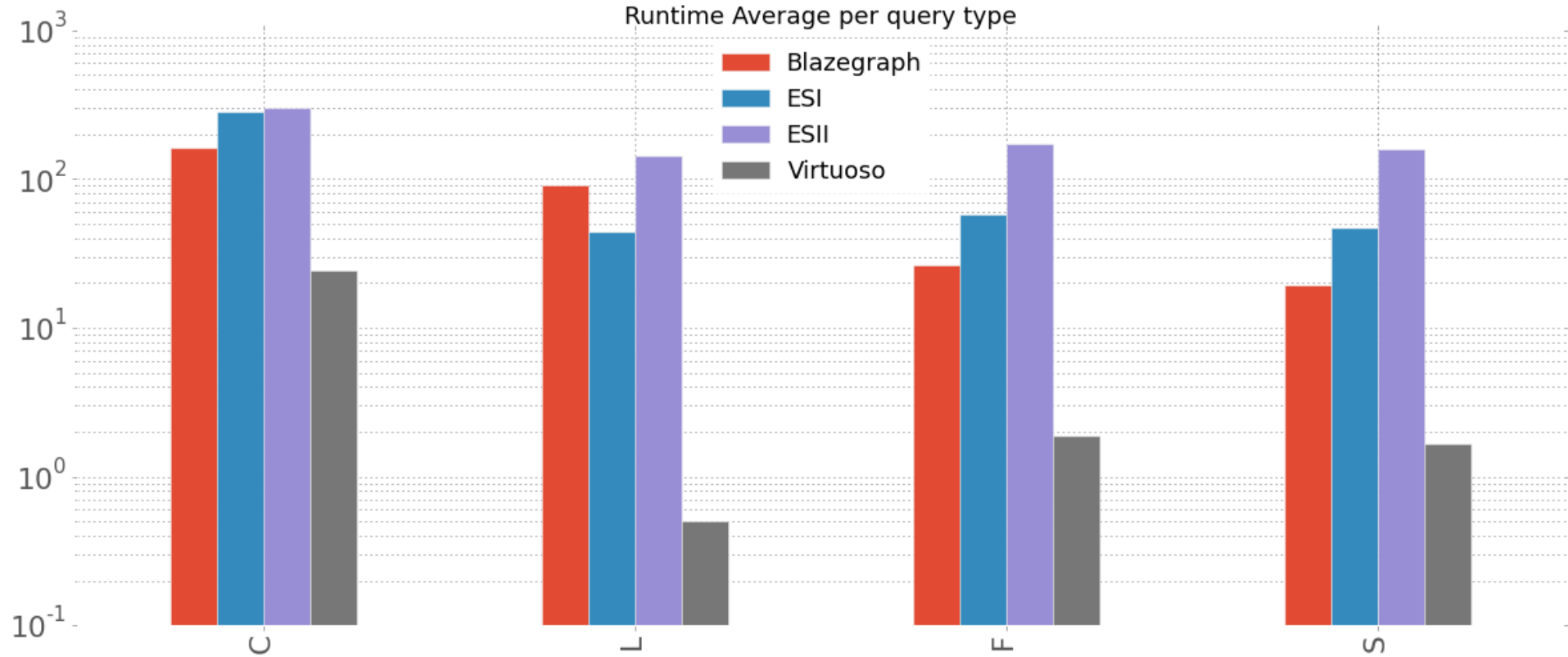
Star query **S**



Snowflake query **F**

Combinations of those **C**

Behavior: different query types

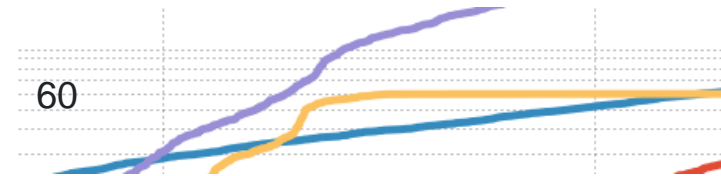


Errors and time-outs

Every runtime > 300s is a time-out.

If the run-time reaches a maximum of < 300s we detect an internal set time-out.

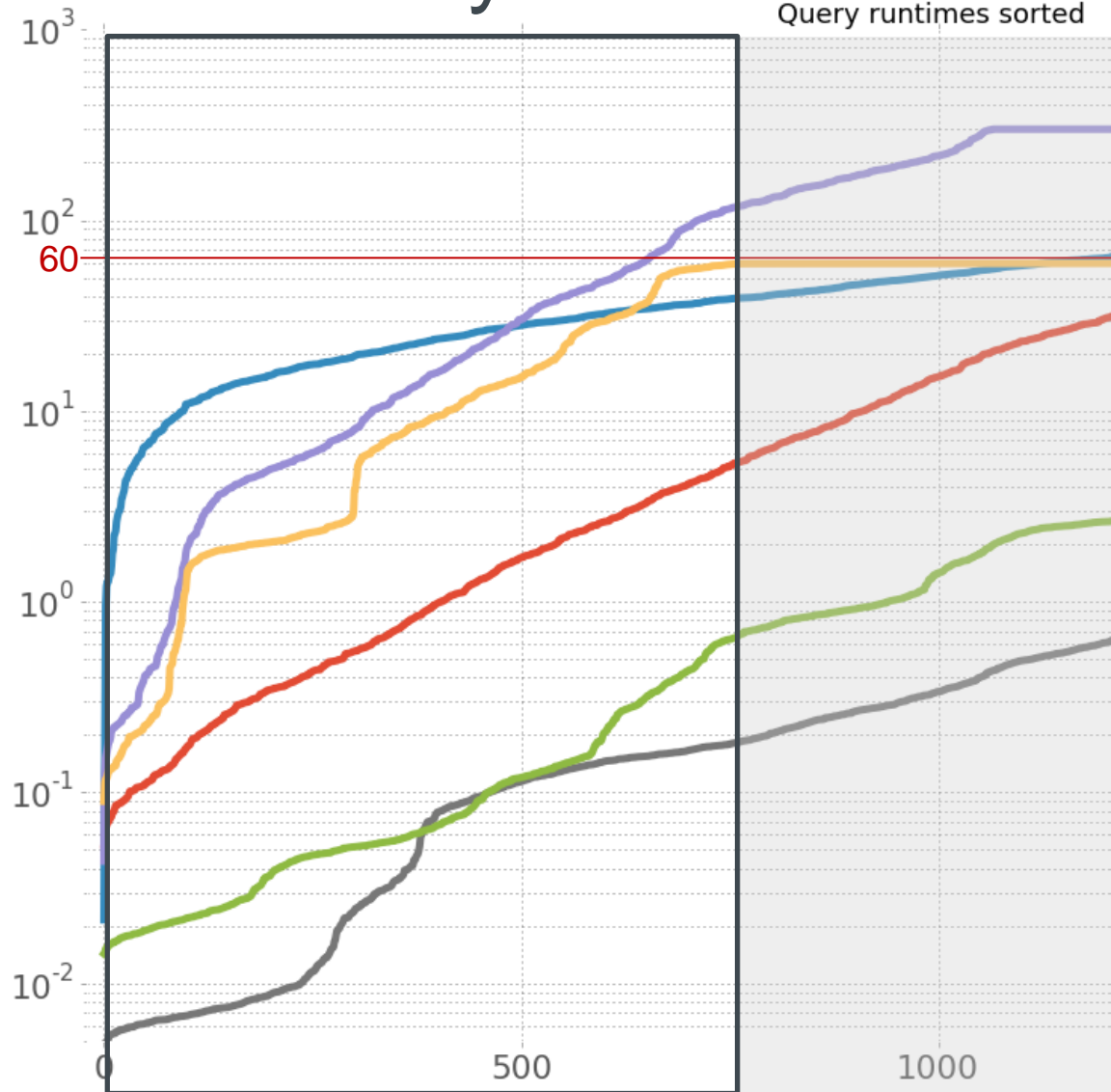
This was in particular the case voor **ESII** (3 nodes)



Benchmark	Timeout Once	Timeout Always
Blazegraph	11.15%	7.05%
ESI	34.20%	5.00%
ESII	46.75%	46.75%
Virtuoso	0.00%	0.00%
ESII (3 nodes)	66.95%	61.50%

Scalability: 1B revisited

Query runtimes sorted



ESII-3 still outperforms ESII-1 when looking at queries that did not time-out

- Blazegraph1
- ESI1
- ESII1
- Virtuoso1
- ESII3
- Virtuoso3

Issues in the followed approach

- Choose for virtual machine images in the cloud (AWS) for reproducibility; but cloud solutions might not always be best suited for production.
- The results of different benchmark studies might depend on many (hidden) configuration factors leading to different or even contradicting results.
- The difference in performance between the stores might be attributed to the use of commodity hardware in the cloud.
- Differences partially attributed to the quality of the recommended configuration parameters as provided by the virtual machine images.

Introduction

Approach

Benchmark

Results

Conclusions & Next Steps

Conclusions & Next steps

- Compared enterprise RDF stores
default configuration
without the intervention of enterprise support.
- Run stores in their optimal configuration (reflecting a production setting) with more instances (> 3).
- Repeat the benchmark with DisQover data and queries.
- Create overview of RDF solutions for different use cases, configurations and real-world (life science) datasets.
- Investigate whether the WatDiv results are confirmed when running the benchmark with other queries and data.
- Release tools for repeating the benchmark with new storage solutions.

Contact Details

E-MAIL: laurens.devocht@ugent.be

TWITTER: [@laurens_d_v](https://twitter.com/laurens_d_v)

SLIDES: slideshare.net/laurensdv