Benchmarking in Semantic Web

Mahmudur Rahman SANIAT

Semantic Web

- Term created by Sir Tim Berners-Lee
- Think about Web-Perspective
  - We started with small and simple datasets
  - Data grew up
  - Lack of common schema
  - Need of Scalability
- It breaks down the information into its simplest form so that it is quickly and easily understood by the user
- RDF is the standard for encoding metadata and other knowledge on the Semantic Web

SPARQL

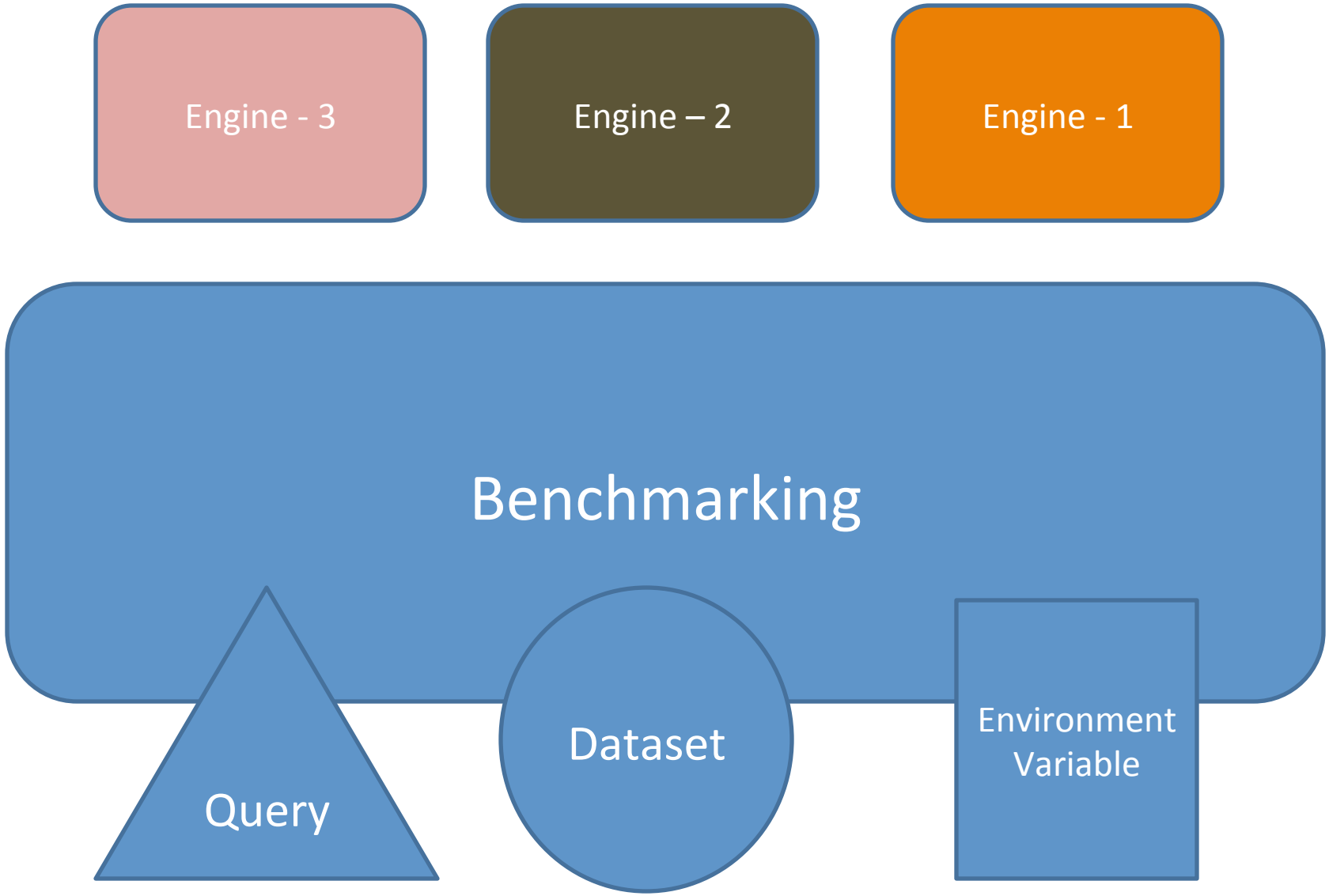
- Simple Protocol And RDF Query Language
- Defines a standard query language and data access protocol for use with the Resource Description Framework (RDF) data model.

SPARQL Endpoint

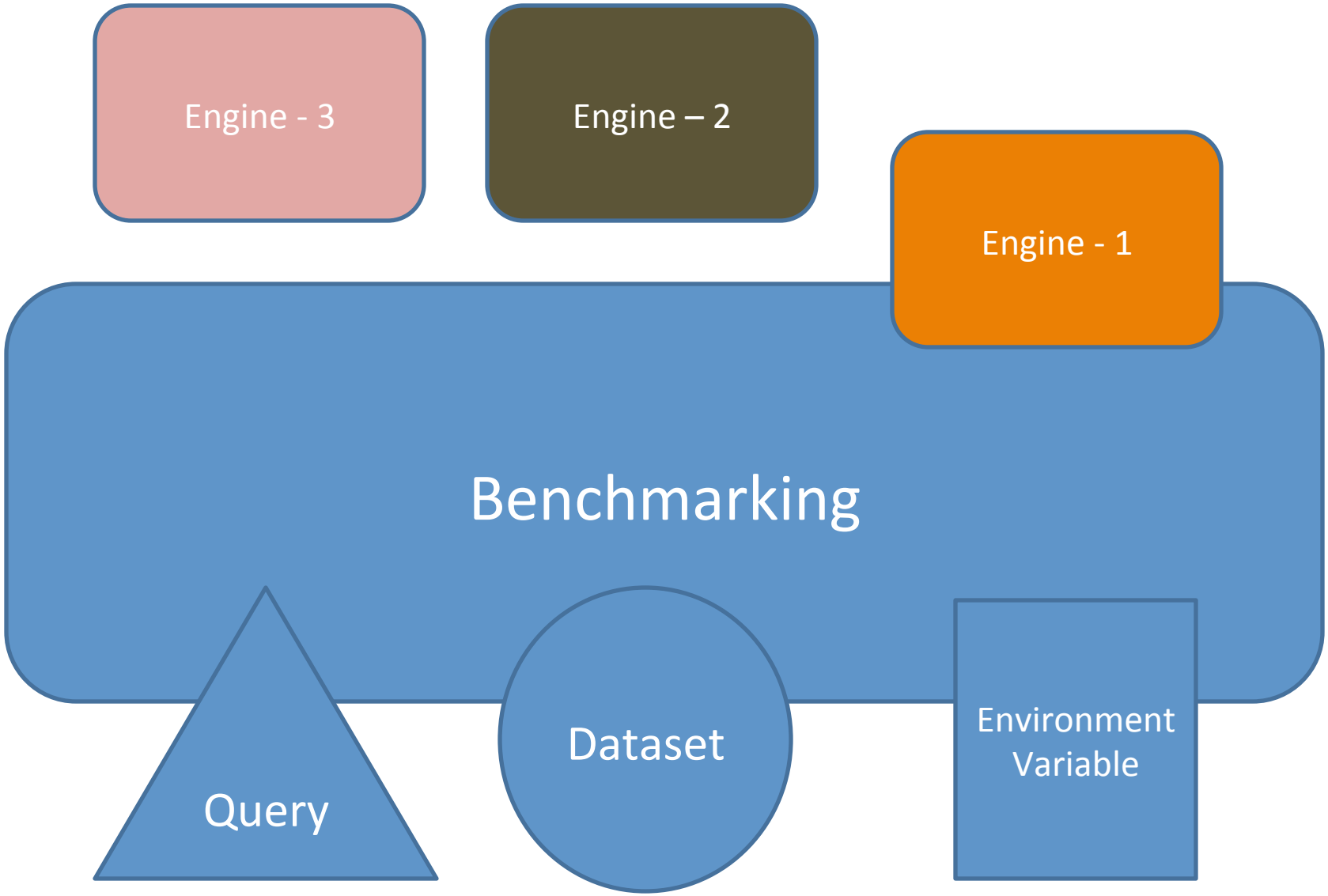
- Is a URI to which Queries can be sent, and which returns answers to the Queries as a response.

- For example DBPedia has a SPARQL endpoint at http://dbpedia.org/sparql

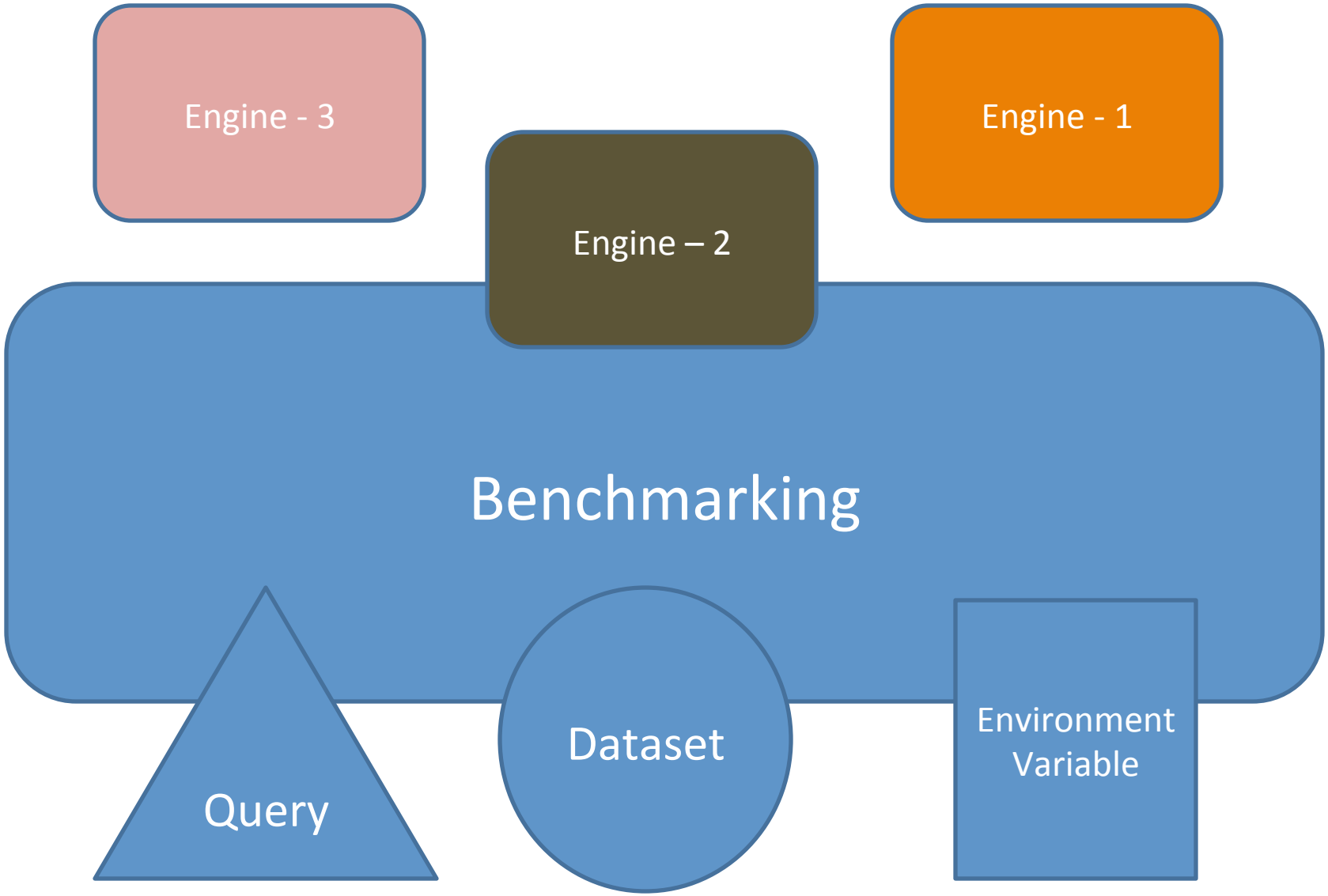
Benchmarking Basics



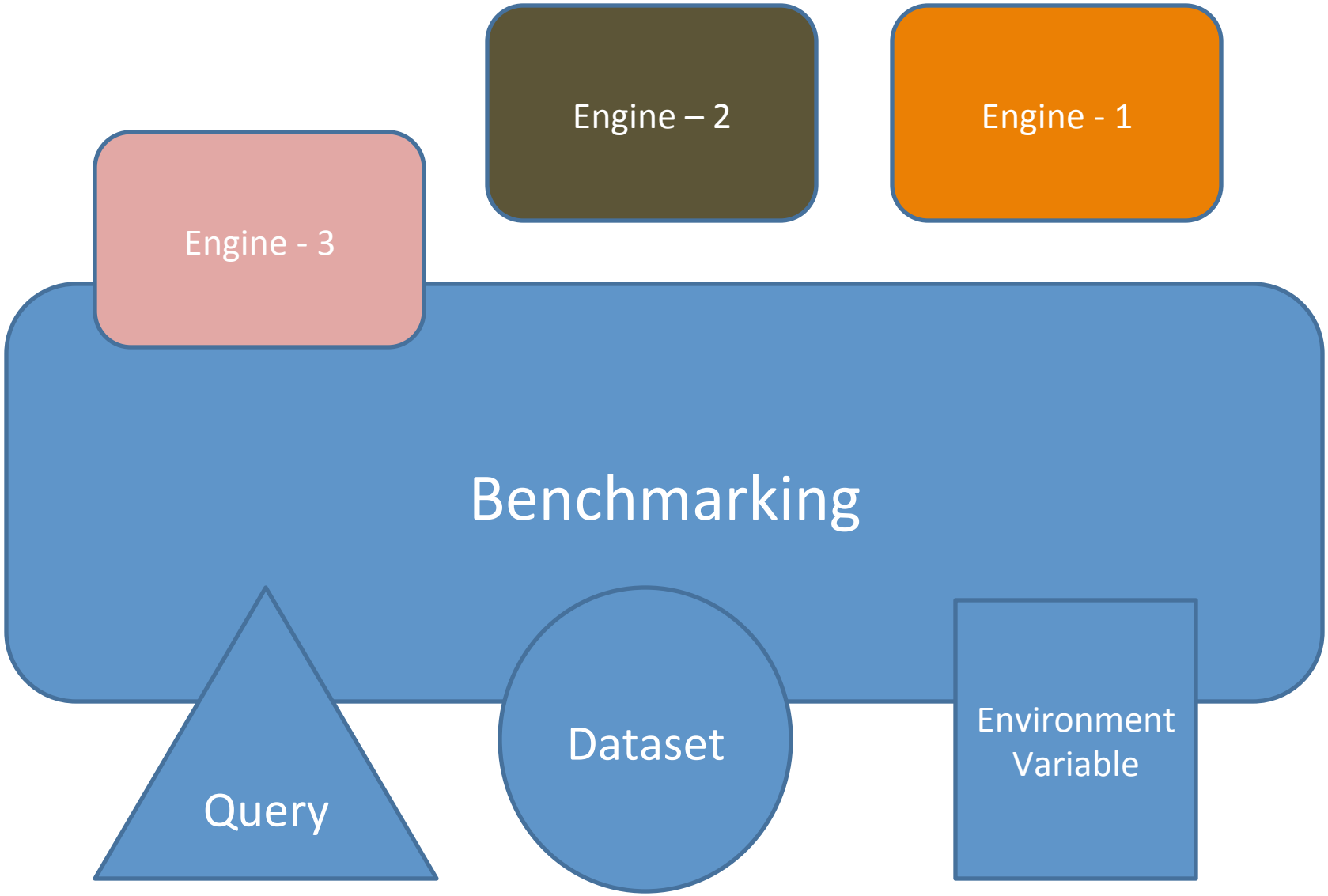
Benchmarking Basics



Benchmarking Basics



Benchmarking Basics



Major Challenge in Benchmarking

- Heterogeneity
- What heterogeneity means here ??
  - Applications may face different settings from both data and query level

- Data Level:
  - Different Physical Distribution of Datasets
  - Different Interfaces for Data Access
  - Incomplete Knowledge

- Query Level:
  - Query Language
  - Expressiveness
  - Ranking

Benchmarking Standards

- Lehigh University Benchmark (LUBM)
  - intended to evaluate the performance of semantic web repositories with respect to extensional queries over a large data set that commits to a single realistic ontology
- The **SP²Bench** SPARQL Performance Benchmark
  - provides a scalable RDF data generator and a set of benchmark queries, designed to test typical SPARQL operator constellations and RDF data access patterns.
- Ontology Benchmark (UOBM)
  - extends the LUBM benchmark in terms of inference and scalability testing.
- Social Network Intelligence Benchmark (SIB)
  - A benchmark suite developed by people at CWI and Openlink taking the schema from Social Networks for generating test areas where RDF/SPARQL can truly excel, and challenging query processing over highly connected graph

FedBench

- Benchmarking suite for testing and analyzing the performance of federated query processing strategies on semantic data

FedBench Claims

- Flexible
- Suitable for variety of use cases
- Elaborated data and query sets
- Various application scenarios

"Our results are published in a Wiki, where we also maintain data, statistics, queries, and scenarios. We invite researchers and benchmark users to customize and extend the benchmark suite according to their own needs" [1]

FedBench Claim

Apples and Oranges: A Comparison of RDF Benchmarks and Real RDF Datasets States that:

"Artificial datasets used in benchmarks are typically highly structured, while Linked Data are less structured. They conclude that benchmarks should not solely rely on artificial data but also consider real world datasets."

[FedBench Claims to solve this] – discussion later

Related works on Benchmarking

- The Lehigh University Benchmark
  - designed to test the reasoning capabilities of systems over a single ontology.
- The SPARQL-specic, use-case driven Berlin SPARQL Benchmark
  - comes with a set of queries implementing meaningful requests on top of an eCommerce scenario modeled in RDF.
- The SPARQL Performance Benchmark (SP2Bench)
  - puts a stronger focus on language specific features of the SPARQL query language, addressing optimization in complex scenarios

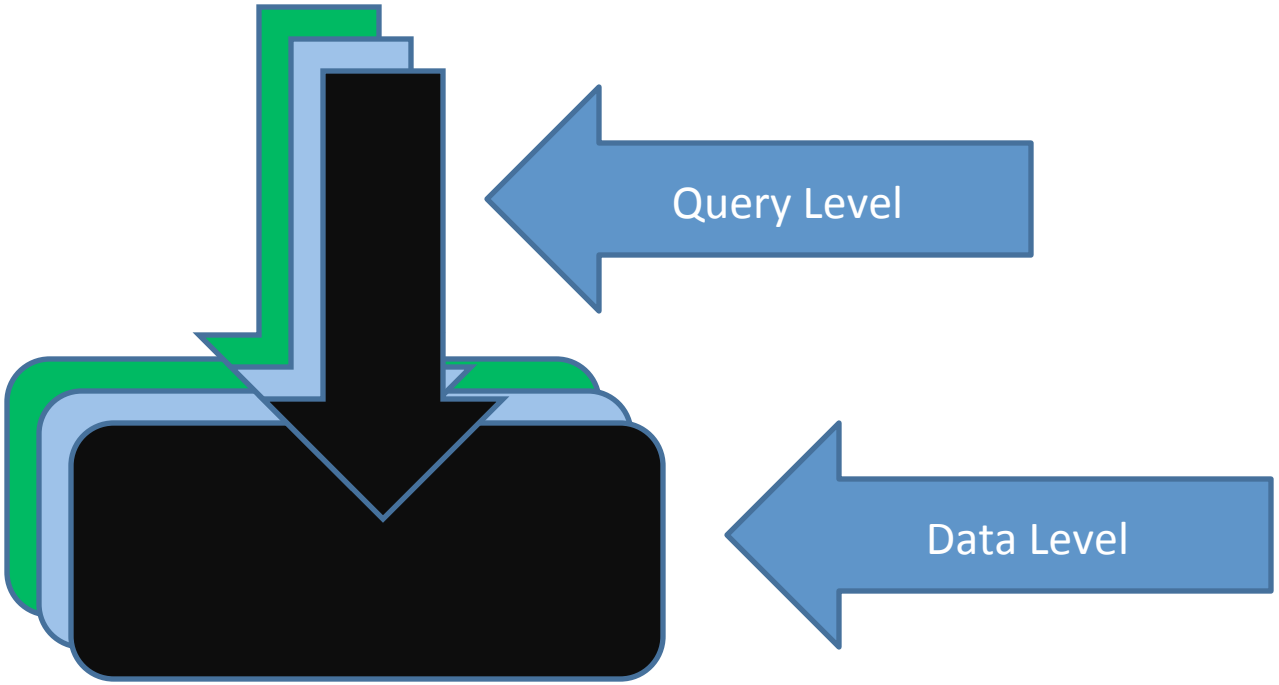
Why a New Benchmark ??

- None of the benchmarks mentioned previously considers federation at data level, nor does provide data collections consisting of multiple interlinked datasets.
- They used queries defined in SPARQL 1.0, without requiring specific extensions for explicit specification of the endpoint services as proposed by the SPARQL 1.1 federation extensions. There is currently no public implementation of the SPARQL 1.1 federation extension.

Previous Papers about FedBench

- Presented a holistic benchmark suite, including a variety of new data and query sets, new scenarios such as Linked Data access (i.e., via HTTP requests),
- An automated evaluation framework (with support for various metrics like counting the number of requests, automated evaluation, interfaces for connecting new systems etc.),
- Discussion and classification of state-of-the-art approaches
- Discussion of statistics, as well as novel experiments and findings.

Challenges for Benchmarking



Heterogeneity in Data Level

- Physical Distribution
- Data Access Interface
- Data Source Existence
- Data Statistics

Heterogeneity in Data Level

Physical Distribution

Federated query processing systems may either access and process:

- Global data from the Web
- Locally stored data
- Mixed

Heterogeneity in Data Level

Data Access Interface

- Semantic data may be accessible through different interfaces:
  - There may be native repositories,
  - SPARQL endpoints,
  - Linked Data accessible through HTTP requests.

- These interfaces provide different access paths to the data - URI lookups, expressive queries in different languages

Data Source Existence

- In particular in Linked Data scenarios, not all sources may be known a priori. Hence, applications may have only few entry points into the data graph
- These can be used to iteratively deep-dive by exploring links

Heterogeneity in Data Level

Data Statistics

- Best Case: Advanced statistical information about properties, counts, and distributions in the form of histograms for all data sets are available

- Worst case: (in particular if data is not stored locally) only few or no information about the data sources may be given

Heterogeneity in Query Level

- Query Language
- Result Completeness
- Ranking

Heterogeneity in Query Level

Query Language

Some applications get around with simple conjunctive queries, others may rely on the full expressive power of RDF query languages, such as the de facto standard SPARQL

Heterogeneity in Query Level

Result completeness

Certain applications may rely on complete results, but others may ignore them

Because: **Responsiveness is the first priority to them!**

Heterogeneity in Query Level

Ranking

Applications may be interested in top-k results

Reminder of the claim!

Apples and Oranges: A Comparison of RDF Benchmarks and Real RDF Datasets States that:

"Artificial datasets used in benchmarks are typically highly structured, while Linked Data are less structured. They conclude that benchmarks should not solely rely on artificial data but also consider real world datasets."

FedBench Dataset Collection

- General Linked Open Data Collection
  - Dbpedia
  - GeoNames (information about geographic entities like countries and cities)
- Life Science Data Collection
  - KEGG (Chemical compound and Reaction data)
  - ChEBI (a dictionary of molecular entities focused on small chemical compounds)
- SP2Bench
  - The data generator provides a single dataset, from which we created a collection by clustering by the types occurring in the dataset, finally obtaining 16 sub-datasets (10M triples in total)

Benchmark Queries

- There are two reasonable options to design a query:
  - Language Specific
  - Use Case Driven

- They have taken the both approaches

Life Science and Cross Domain Queries

- Focuses on:
  - number of data sources involved,
  - join complexity
  - Types of links used to join sources
  - varying query (and intermediate) result size

SP2Bench Queries

- Reused the queries from the SP2Bench SPARQL performance benchmark, which were designed to test a variety of SPARQL constructs and operator constellations, but also cover characteristics like data access patterns, result size, and different join selectivities.

Linked Data Queries

- Today's Linked Data engines typically focus on basic graph patterns (Conjunctive Queries). Therefore, all LD queries are basic graph pattern queries

Evaluation of the Benchmark

- They Developed a Java Benchmark Driver
- It provides an integrated execution engine for the different scenarios and is highly configurable
- The driver comes with predefined configurations for the benchmark scenarios
- Custom scenarios can be created intuitively by writing config files that define properties for data configuration and other benchmark settings (query sets, number of runs, timeout, output mediator, etc)

Benchmark Evaluation

1. Centralized processing, where all data is held in a local, central store
2. Local federation, where they used a federation of local repository endpoints, all of which are linked to each other in a federation layer
3. A federation of SPARQL endpoints, also linked to each other in a common local federation layer.
   - This pursue the goal to test the overhead that is imposed by the SPARQL requests exchanged over the associated HTTP layer

Experiments

- Alibaba Federation vs. SPLENDID Federation
- SP2Bench scenario
- Linked Data
- RDF Database - SPARQL Endpoints (described before)

Experiment Output ??

- FedBench Claims: *"As witnessed by the evaluation, our benchmark is flexible enough to cover a wide range of semantic data application processing strategies and use cases"*

- BUT: What about
  - Network Latency ??
  - Data Partitioning ??

Are Existing Testbeds enough ?

- FedBench – the most recent and complete testbed
- But it do not specify/consider some of the variables and configuration setups that actually affects the quality and performance of different solutions
  - "This may lead to incorrect characterizations when these testbeds are used to select the most appropriate systems in a given scenario, or to decide the next steps in their development."

Testing FedBench

- Tried FedBench on 3 systems:
  - ANAPSID
  - ARQ
  - FedX
- 3 sets of queries proposed:
  - Life Science
  - Cross Domain
  - Linked Data
- Network Latency added
- Different Data Distributions added

Designing Benchmark

- Independent Variable
  - those characteristics that need to be minimally specified in the benchmark in order to ensure that evaluation scenarios are replicable
  - Query, Data, Platform, Endpoint
- Dependent (observed) Variable
  - Endpoint Selection time
  - Execution time
  - Answer Completeness

Dependent and Independent Variable

	Independent Variables	Observed Variables		
		Endpoint Selection Time	Execution Time	Answer Completeness
Query	query plan shape	✓	✓	✓
	# basic triple patterns	✓	✓	✓
	# instantiations and their position	✓	✓	
	join selectivity		✓	
	# intermediate results		✓	
	answer size		✓	
	usage of query language expressivity	✓	✓	
	# general predicates	✓	✓	✓
Data	dataset size		✓	
	data frequency distribution		✓	
	type of partitioning	✓	✓	✓
	data endpoint distribution	✓	✓	✓
Platform	cache on/off	✓	✓	
	RAM available	✓	✓	
	# processors	✓	✓	
Endpoint	#endpoints	✓	✓	✓
	endpoint type	✓	✓	
	relation graph/endpoint/instance		✓	✓
	network latency	✓	✓	✓
	initial delay	✓	✓	
	message size		✓	
	transfer distribution	✓	✓	✓
	answer size limit		✓	✓
	timeout		✓	✓

Table 2. Variables that impact the behavior of SPARQL federated engines

(1) Network Latency Effect

	Number of results			Execution time (secs.) (first tuple)			Execution time (secs.) (all tuples)		
Query Engine	Medium	Fast	Perfect	Medium	Fast	Perfect	Medium	Fast	Perfect
ANAPSID	61	61	61	0.98	0.17	0.16	0.98	0.17	0.16
FedX	61	61	61	16.93	2.23	0.72	16.93	2.23	0.72
ARQ	–	–	63	–	–	0.98	–	–	0.98

Table 1. Evaluation of FedBench query CD1-Number of results and execution time (secs.) under different network latency conditions. Timeout was set up to 30 minutes. Perfect Network (No Delays); Fast Network (Delays follow Gamma distribution ($\alpha = 1$, $\beta = 0.3$); Medium-Fast Network (Delays follow Gamma distribution ($\alpha = 3$, $\beta = 1.0$))

(2) Reason/Complexity of Queries

- We use the queries in FedBench, but there is no clear reasoning for them.
- Why do we use them ??
- The Reasoning may be like this:

  "CD5 is a chain-like query for finding film entities linked via owl:sameAs and restricted on genre and director"

- Some Characteristics are not sufficiently discussed
  - the number of triple patterns in each basic graph pattern appearing in the query, the selectivity of each part of the query, etc

(3) Data Partitioning

- Partitioning refers to the way that the RDF dataset is fragmented.
- Horizontal partitioning impacts on the completeness of the answer
- Vertical partitioning affects the execution time.
- Replication: Increases availability

(4) Platform Dimension

- Cache
- RAM
- Processor

The meaning of dropping and warming up cache needs to be clearly specified as well as the number of iterations where an experiment is run in warm cache, and when cache contents are dropped off!

(5) Endpoint Dimension

- No. of endpoints
- Type of endpoints

  - Result completeness depends on them because different endpoints may produce different answers

“There is no end to Perfectness”

References

- A Short Description of Semantic Web

  http://www.semagix.com/what-is-semantic-data/short-description-of-semantic-data.htm

- RDF Store Benchmarking

http://www.w3.org/wiki/RdfStoreBenchmarking

[1] FedBench- A Benchmark Suite for Federated Semantic Data Query Processing