A Hybrid Approach to Linked Data Query Processing with Time Constraints

Steven Lynden, Isao Kojima, Akiyoshi Matono, Akihito Nakamura, Makoto Yui

National Institute of Advanced Industrial Science and Technology, Japan

Motivation

- Indexing systems, e.g. Sindice, can be used to query the Semantic Web, however:
  - *Hybrid SPARQL queries: fresh vs. fast results - Umbrich et al.*
    - Coherence
    - A significant proportion of data from Sindice etc. may not be up-to-date with sources.

- Existing distributed SPARQL query processing systems are often very unpredictable in terms of response time.

- Some applications may require a best effort in a fixed amount of time
  - e.g. a portal for browsing a Linked Data repository attempting to suggest related RDF data from other sources requiring answers from a query processing back-end within the average time a user stays on a page.

Proposed approach

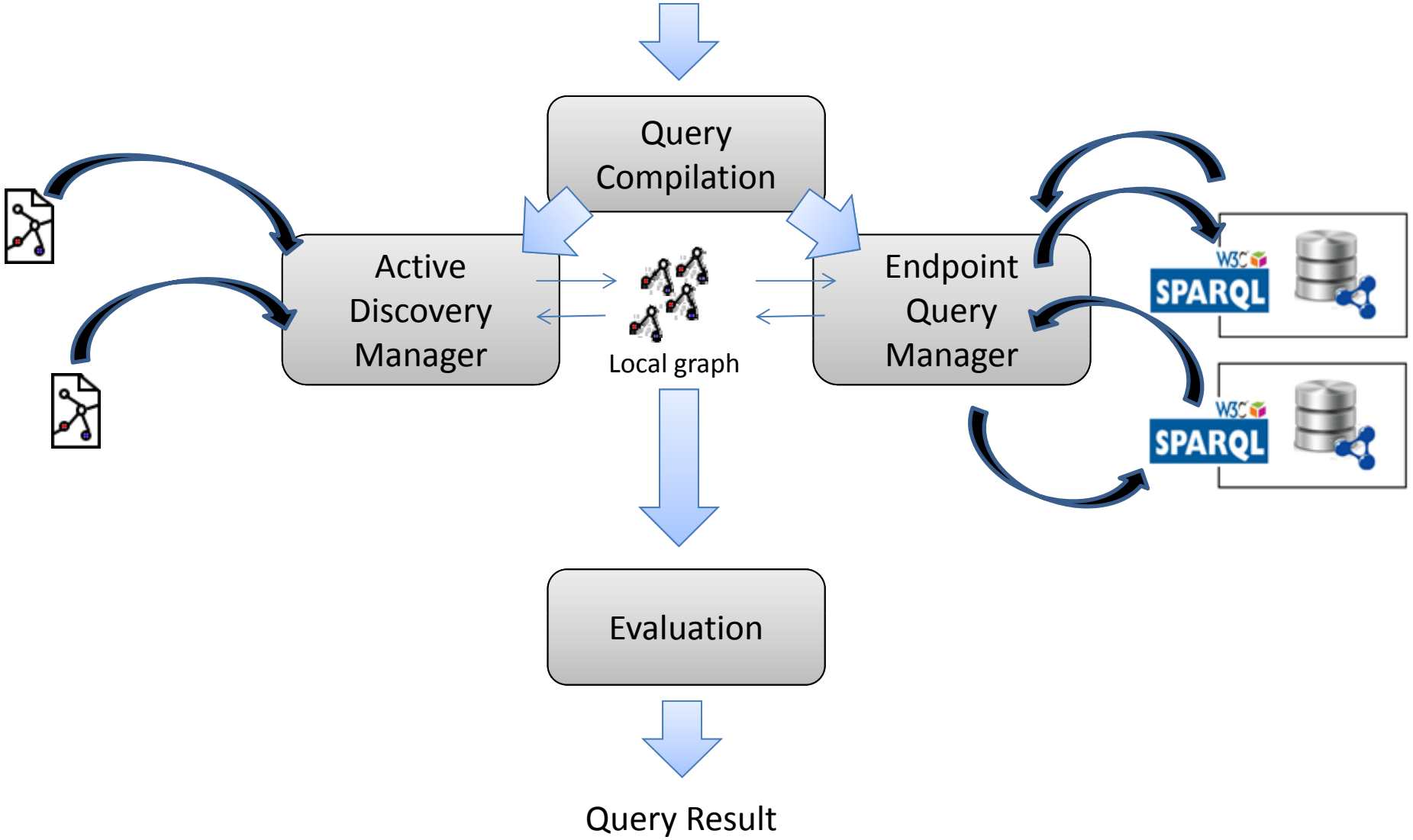
- Execute two components in parallel
  - Active discovery
    - Investigate URIs, retrieve RDF data, match against triple patterns in the query applying FILTER predicates
  - Query SPARQL endpoints
    - Construct sub-queries from the federated query, execute them using available SPARQL endpoints
- Both components share a local graph data structure in which a temporary result is constructed
- After a set time period, both components terminated and the local graph transformed into a query result

Hybrid Query Processing with Time Constraints

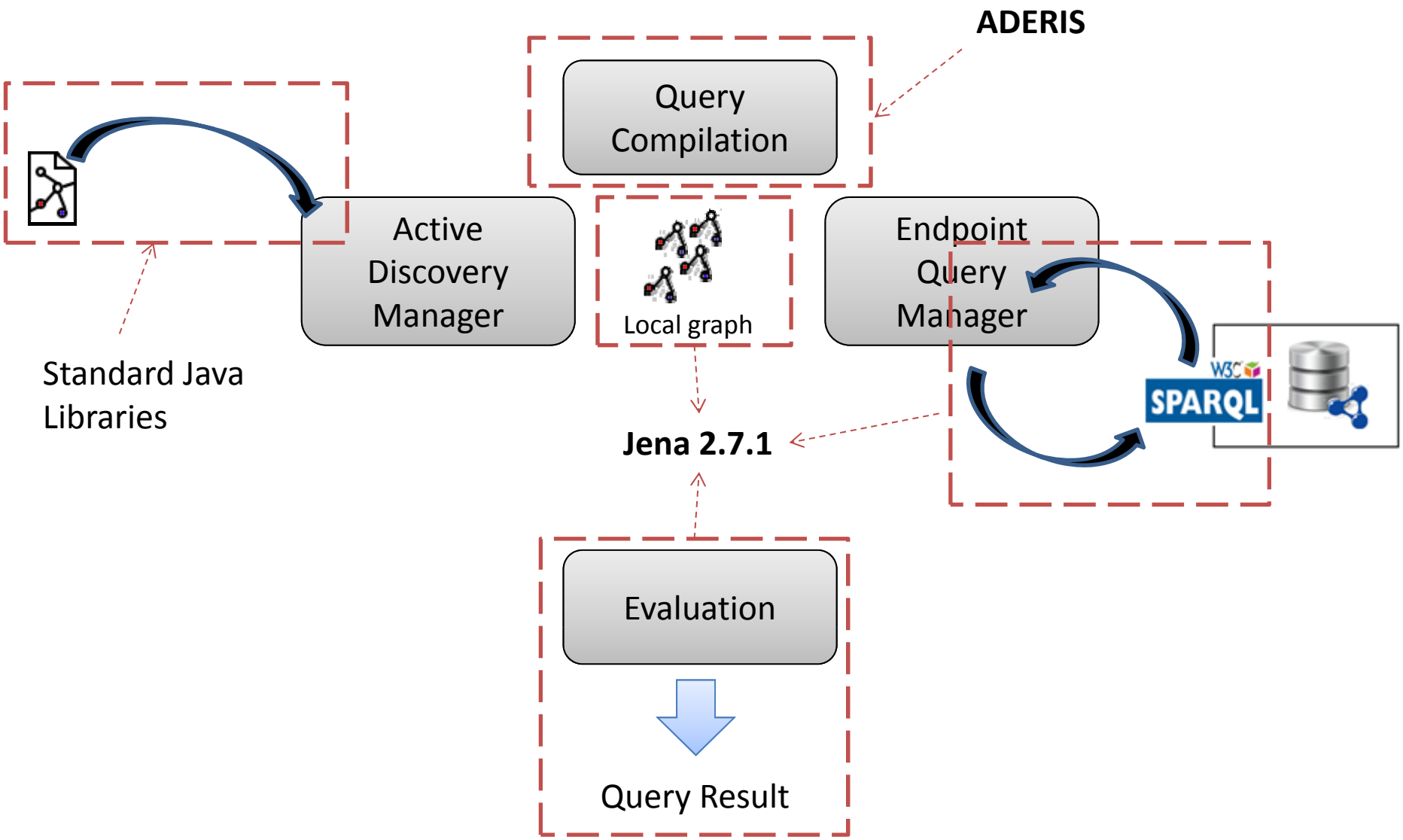
- Compile Query
- Access SPARQL endpoints and documents containing RDF data
- Stop and evaluate

User's SPARQL Query

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX dbp: <http://dbpedia.org/resource/property>
SELECT * WHERE {
    ?x dc:subject dbp:FIFA_World_Cup-winning_countries .
    ?p dbp:managerclubs ?x .
    ?p foaf:name "Luiz Felipe Scolari"@en .
}
```



Implementation



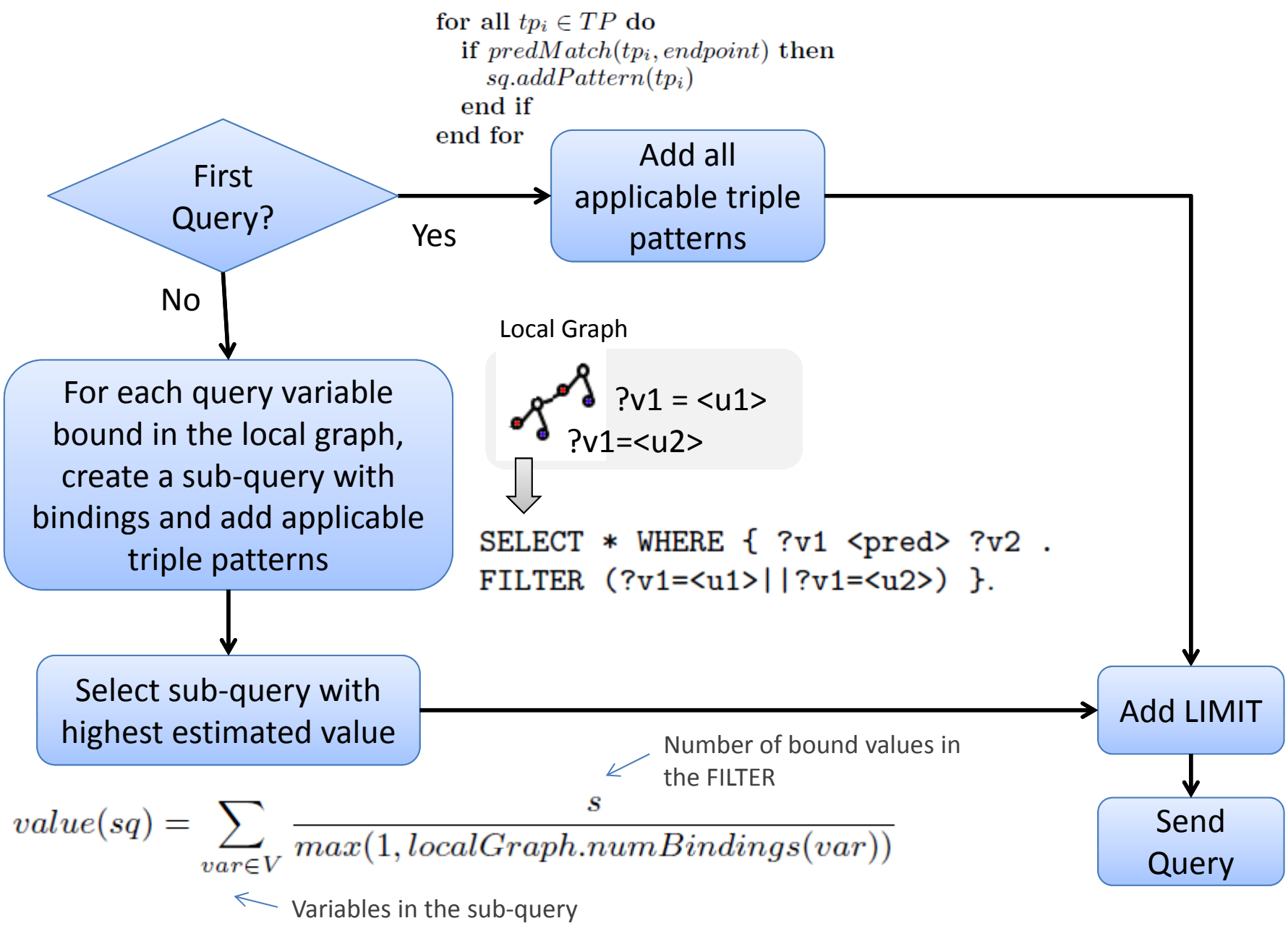
Endpoint Query Manager

- Prior to query execution the system is configured with a set of endpoints to be used
- Existence of triples with a given predicate assumed to be known, e.g:

  ?paper <**http://swrc.ontoware.org/ontology#author**> ?p

  triple pattern matches exist in the **data.semanticweb.org** endpoint

  (Predicates in query triple patterns are usually not variables)
- Objectives
  - Execute simple queries to provide results quickly that can be explored by the active discovery manager in parallel
  - Avoid placing excessive burden on endpoints and avoid fair-use restrictions


for all $tp_i \in TP$ do
if $predMatch(tp_i, endpoint)$ then
$sq.addPattern(tp_i)$
end if
end for
First Query?
Yes
No
Add all applicable triple patterns
For each query variable bound in the local graph, create a sub-query with bindings and add applicable triple patterns
Local Graph
?v1 = <u1>
?v1=<u2>
SELECT * WHERE { ?v1 <pred> ?v2 .
FILTER (?v1=<u1>||?v1=<u2>) }.
Select sub-query with highest estimated value
Add LIMIT
Send Query
Number of bound values in the FILTER
$value(sq) = \sum_{var \in V} \frac{s}{max(1, localGraph.numBindings(var))}$
Variables in the sub-query

Active Discovery Manager

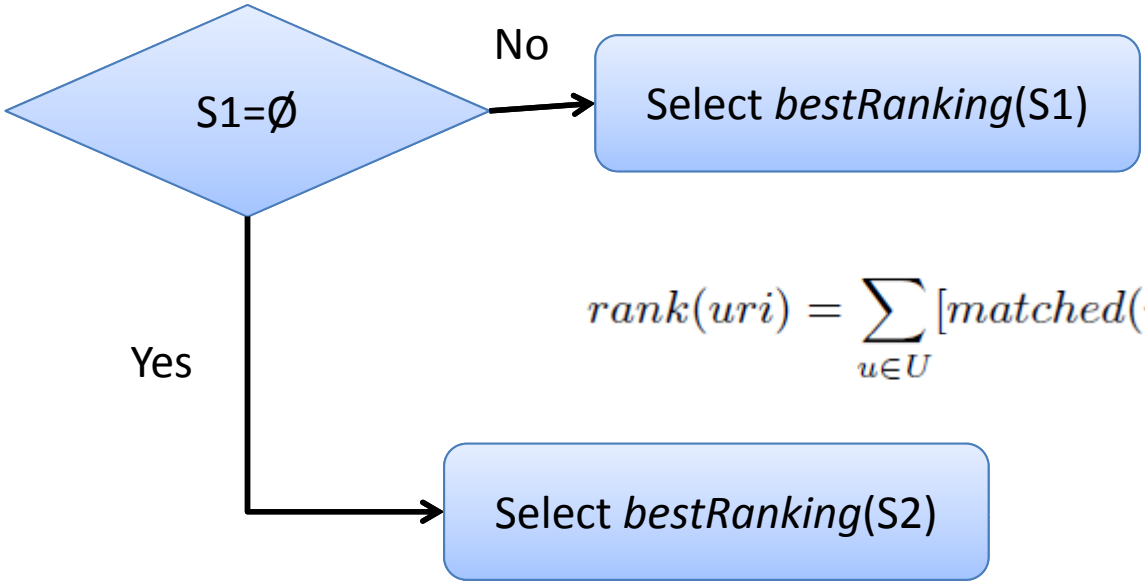
- The active discover manager starts a thread for each Pay Level Domain (PLD) present in URIs in the query and as they are added to the local graph.
- Each thread is able to choose two URIs to investigate each second.

- Objective:
  - Match triple patterns in the query with RDF data retrieved via dereferencing the URIs

DBpedia URIs investigated and the number of triples matching triple patterns in the query.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX dbp: <http://dbpedia.org/resource/property>
SELECT * WHERE {
    ?x dc:subject dbp:FIFA_World_Cup-winning_countries .
    ?p dbp:managerclubs ?x .
    ?p foaf:name "Luiz Felipe Scolari"@en .
}
```

```
/resource/England_national_football_team (25)
/resource/Spain_national_football_team (23)
/resource/Brazil_national_team (18)
/resource/FC_Bunyodkor  (3)
/resource/Vicente_Feola  (2)
/resource/Luiz_Felipe_Scolari (16)
```

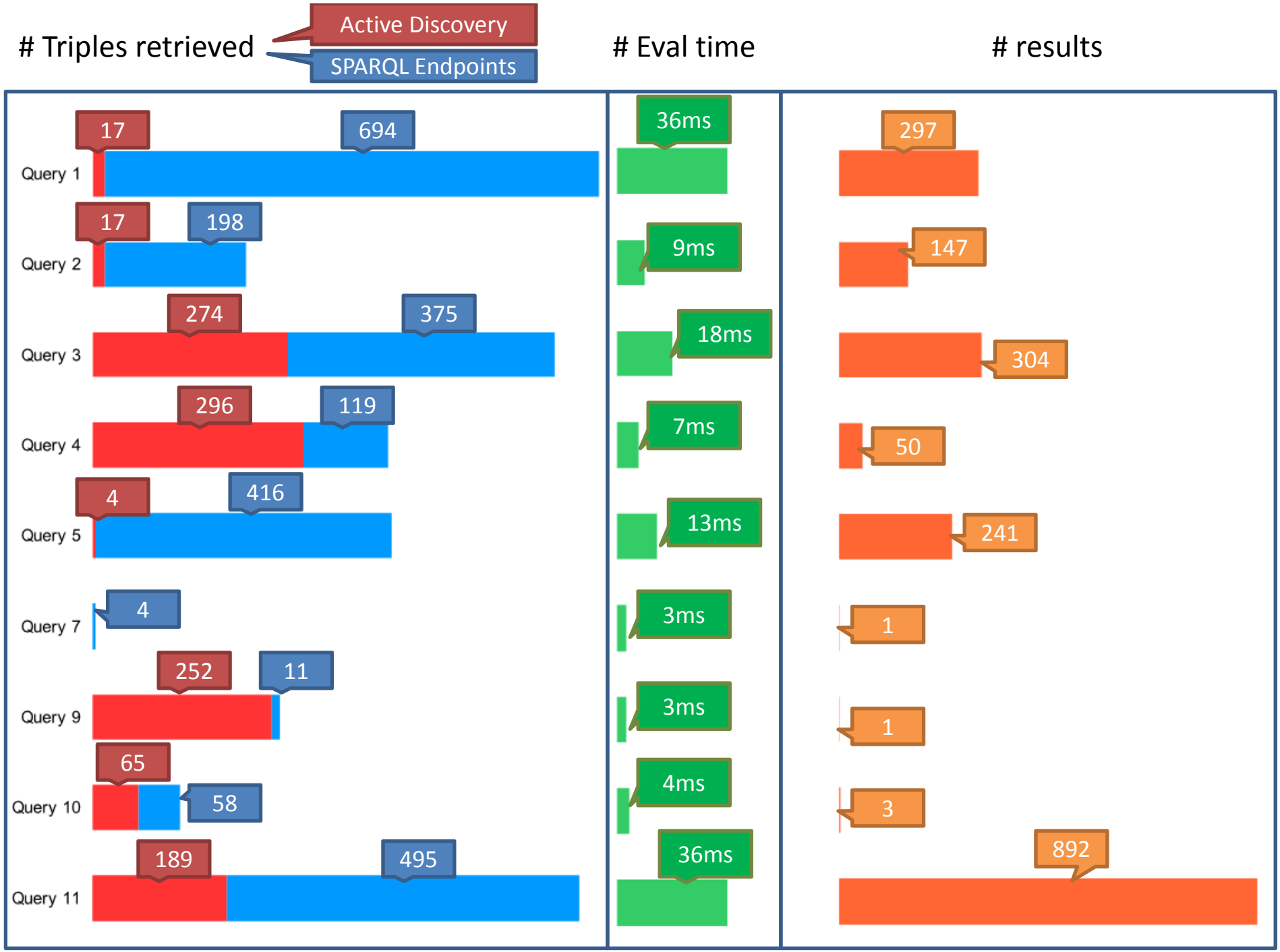


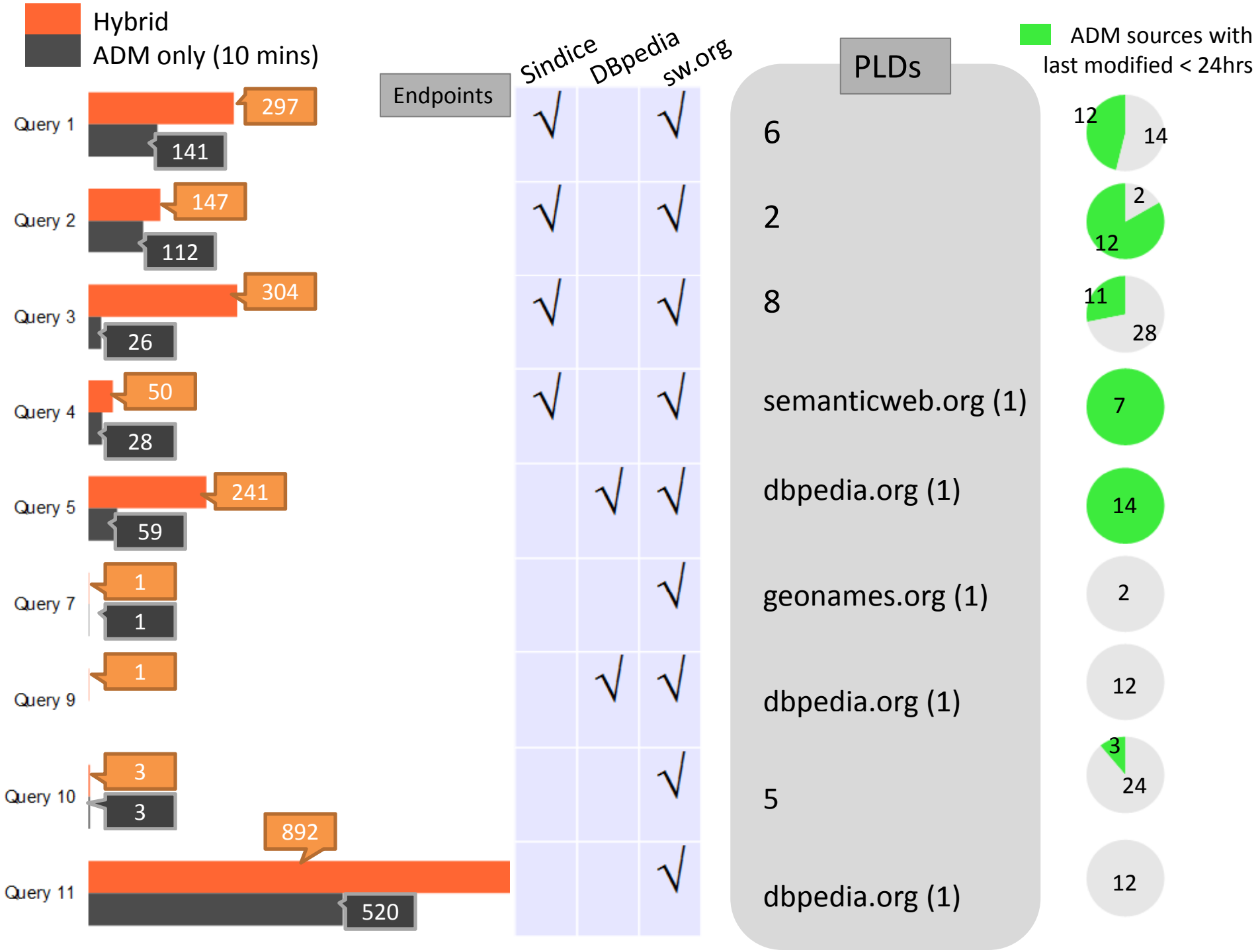
$$rank(uri) = \sum_{u \in U} [matched(u) * (1 - distance(u, uri))]$$

Levenshtein distance

Evaluation

- **FedBench**
  - Benchmark for testing the efficiency and effectiveness of federated query processing on semantic data.
- Multiple query sets, we used the Linked Data (LD) query set.
- 11 Queries, however some problems encountered with 2 of the queries.
- Remaining queries executed using the proposed approach with a limit of 10 seconds.


Triples retrieved
Active Discovery
SPARQL Endpoints
Eval time
results
Query 1
17
694
36ms
297
Query 2
17
198
9ms
147
Query 3
274
375
18ms
304
Query 4
296
119
7ms
50
Query 5
4
416
13ms
241
Query 7
4
3ms
1
Query 9
252
11
3ms
1
Query 10
65
58
4ms
3
Query 11
189
495
36ms
892


Hybrid
ADM only (10 mins)
Query 1
297
141
Query 2
147
112
Query 3
304
26
Query 4
50
28
Query 5
241
59
Query 7
1
1
Query 9
1
Query 10
3
3
Query 11
892
520
Endpoints
Sindice
DBpedia
sw.org
PLDs
6
2
8
semanticweb.org (1)
dbpedia.org (1)
geonames.org (1)
dbpedia.org (1)
5
dbpedia.org (1)
ADM sources with last modified < 24hrs
12
14
2
12
11
28
7
14
2
12
3
24
12

Conclusions

- Answering the FedBench Linked Data queries in accordance with our objective of within 10 seconds was possible using the proposed technique.
- Advantages include:
  - Fault tolerance
  - Freshness
  - Increased coverage
  - Mitigation of fair-use restrictions
- Future work will investigate benefits with more dynamic data, e.g. RDFa etc and optimisation based on relevance /quality of data sources