Best-effort Linked Data Query Processing with time constraints using ADERIS-Hybrid

Steven Lynden, Isao Kojima, Akiyoshi Matono, and Akihito Nakamura

Information Technology Research Institute
National Institute of Advanced Industrial Science and Technology (AIST), Japan
{steven.lynden|isao.kojima|a.matono|nakamura-akihito}@aist.go.jp

Abstract. Answering SPARQL queries over the Web of Linked Data is a challenging problem. Approaches based on distributed query processing provide up-to-date results but can suffer from delayed response times, indexing-based approaches provide fast response times but results can be out-of-date and the costs of indexing the growing Web of Linked Data are potentially huge. Hybrid approaches try to offer the best of both. In this demo paper we describe a system for answering SPARQL queries within fixed time constraints by accessing SPARQL endpoints and the Web of Linked Data directly.

1 Introduction

Answering Linked Data queries in a timely manner is a challenging problem. An example of one approach towards this is Sindice [1], which provides a SPARQL query interface over RDF data that has been indexed via crawling the Web of Linked Data. Other examples include approaches based on distributed query processing over SPARQL endpoints [5], in addition to link traversal, live exploration and hybrid approaches as surveyed in [3], which access the Web of Linked Data directly during query execution. However, such approaches may result in unpredictable query execution times in the order of minutes for even basic queries, and while there are obviously applications that would utilise such results, it is sometimes more important that an approximate or incomplete answer is provided within a shorter time frame. In this paper we introduce a system based on a hybrid approach where SPARQL endpoints such as Sindice and the Web of Linked Data are accessed in parallel to answer queries within fixed time constraints. The system can be found at `http://aderis.linkedopendata.net`.

2 Hybrid Linked Data Query Processing with Time Constraints

The approach, illustrated in Figure 1, proceeds as follows:

1. A federated SPARQL query is parsed and compiled into a set of triple patterns. The query is entirely declarative written without knowledge of the

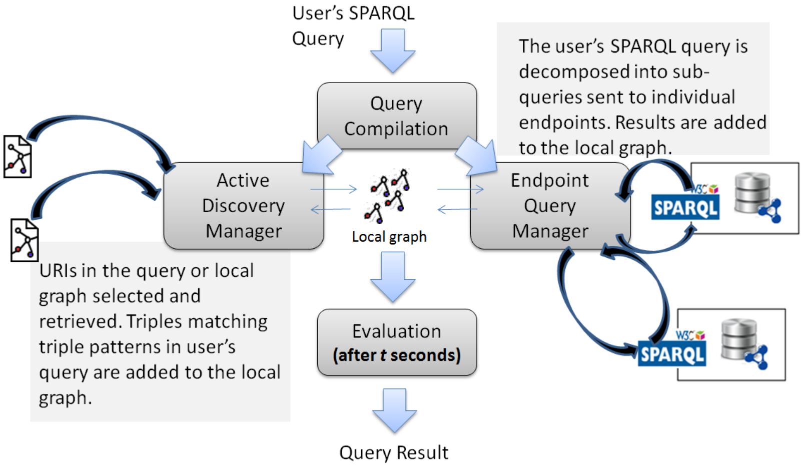

User's SPARQL Query
Query Compilation
The user's SPARQL query is decomposed into sub-queries sent to individual endpoints. Results are added to the local graph.
Active Discovery Manager
Local graph
Endpoint Query Manager
SPARQL
SPARQL
URIs in the query or local graph selected and retrieved. Triples matching triple patterns in user's query are added to the local graph.
Evaluation (after *t* seconds)
Query Result

Fig. 1. System details

The active discovery manager and endpoint query manager run in parallel for a fixed time, are then terminated and the local graph is converted into a query result.

location of data, in contrast to, for example the SPARQL 1.1 Federation [2] extensions.

2. A *local graph* component is initialised to store intermediate results, i.e. triples which have been found to match the set of triple patterns in the query.
3. Two components are executed, the *endpoint query manager* and the *active discovery manager*. The endpoint query manager sends queries to SPARQL endpoints and the active discovery manager dereferences URIs and matches the RDF triples retrieved with triple patterns in the query.
4. After a time t, for which the query is scheduled to run, the endpoint query manager and active discovery manager are terminated and the local graph component is used to obtain the result of the federated query.

For a detailed description of the optimisation strategies implemented by the active discovery manager and endpoint query manager, please refer to [4]. For the purposes of an effective demonstration we have chosen t to be 10 seconds for the optimisation and configuration of the system, however the value can be changed. The rationale being that this is a response time within which at least some useful answers can usually be obtained and for which users are generally willing to wait. Compared with the work as presented in [4], we have extended the system with a cache and more extensive statistics from the Web of Linked Data aimed at prioritising the retrieval of fresh, up-to-date data by the active discovery manager. As queries are answered on a best-effort basis, it is important to give the user an idea of how complete the results are estimated to be. An estimate of the completeness of the results (low, medium, or high) is given to the user based on the number of relevant URIs that could not be dereferenced by the active discovery manager in the time allowed, combined with other indicators such as overlap between the triples retrieved from SPARQL endpoints and URIs (i.e. a high degree of overlap indicates that URIs not yet dereferenced would be likely to provide triples already retrieved from endpoints and be therefore unlikely to provide additional query results).

The proposed approach provides increased coverage and fault tolerance due to the fact that multiple data sources are used and the effects of individual data source unavailability can be mitigated. The system provides parallel query execution by pushing down query fragments to individual SPARQL endpoints and automatically optimises the queries sent to individual endpoints to comply with fair-use restrictions such as bounds on query execution time.

3 ADERIS-Hybrid Web Application

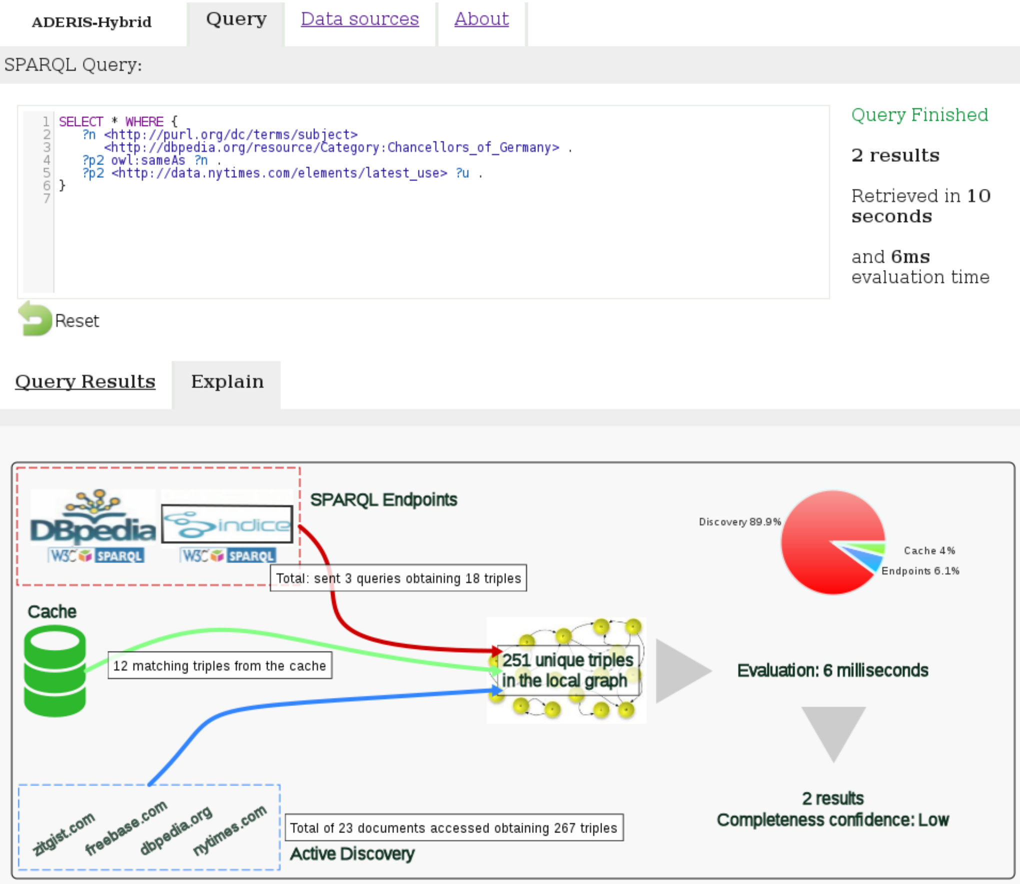
The proposed demo presents the ADERIS-Hybrid Web application implementing the previously described approach, built on our previous work on the Adaptive Distributed Endpoint Integration System (ADERIS) [5], to provide a Web application implementing the hybrid approach described in this paper. The proposed demonstration will highlight the salient aspects of the system including the construction of SPARQL queries, where a set of example queries are provided which can be easily edited by the user; results of queries are presented to the user using the Google Visualization API complemented by a visual representation of the query execution process, as shown in Figure 2. A summary of the statistics used by the query processor is also presented.

4 Conclusion

Answering SPARQL queries over the Web of Linked data with reasonable response times is an important, challenging problem. The proposed demo is a Web application based on our approach in [4], extended with a cache, additional data source statistics, and an estimate of the completeness of the result.

References

1. Sindice: The Semantic Web Index. http://sindice.com.
2. SPARQL 1.1 Federation Extensions. http://www.w3.org/2009/sparql/docs/fed/gen.html.
3. O. Hartig. An Overview on Execution Strategies for Linked Data Queries. *Datenbank-Spektrum*, 13(2):89–99, 2013.
4. S. Lynden, I. Kojima, A. Matono, A. Nakamura, and M. Yui. A Hybrid Approach to Linked Data Query Processing with Time Constraints. In C. Bizer, T. Heath, T. Berners-Lee, and M. Hausenblas, editors, *WWW2013 Workshop on Linked Data on the Web - LDOW 2013*.
5. S. Lynden, I. Kojima, A. Matono, and Y. Tanimura. ADERIS: Adaptively integrating RDF data from SPARQL endpoints (Demo Paper). In *Proceedings of the Database Systems for Advanced Applications (DASFAA) Conference 2010*, 2010.



```
SELECT * WHERE {
   ?n <http://purl.org/dc/terms/subject>
      <http://dbpedia.org/resource/Category:Chancellors_of_Germany> .
   ?p2 owl:sameAs ?n .
   ?p2 <http://data.nytimes.com/elements/latest_use> ?u .
}
```

Query Finished

2 results

Retrieved in **10 seconds**

and **6ms** evaluation time

Reset
Query Results
Explain
DBpedia
W3C SPARQL
indice
W3C SPARQL
SPARQL Endpoints
Discovery 89.9%
Cache 4%
Endpoints 6.1%
Total: sent 3 queries obtaining 18 triples
Cache
12 matching triples from the cache
251 unique triples in the local graph
Evaluation: 6 milliseconds
2 results
Completeness confidence: Low
zitgist.com
freebase.com
dbpedia.org
nytimes.com
Total of 23 documents accessed obtaining 267 triples
Active Discovery

Fig. 2. Web application

The figure is a screenshot of the Web application. Here, the user has executed a SPARQL query, which can be seen in the upper left-hand portion of the screen, and obtained 2 results (not shown in this screenshot due to space restriction). The user is utilising the system's "explain" feature to view how the results were obtained. The percentage of RDF triples that make up the local graph from SPARQL endpoints, the cache and the Web of Linked Data are shown, in addition to a confidence measure that the results are complete.