Versioned Querying with OSTRICH and Comunica in MOCHA 2018

Ruben Taelman, Miel Vander Sande, Ruben Verborgh

IDLab, Department of Electronics and Information Systems, Ghent University – imec

Abstract. In order to exploit the value of historical information in Linked Datasets, we need to be able to store and query different versions of such datasets efficiently. The 2018 edition of the Mighty Storage Challenge (MOCHA) is organized to discover the efficiency of such Linked Data stores and to detect their bottlenecks. One task in this challenge focuses on the storage and querying of versioned datasets, in which we participated by combining the OSTRICH triple store and the Comunica SPARQL engine. In this article, we briefly introduce our system for the versioning task of this challenge. We present the evaluation results that show that our system achieves fast query times for the supported queries, but not all queries are supported by Comunica at the time of writing. These results of this challenge will serve as a guideline for further improvements to our system.

1. Introduction

The Semantic Web [1] of Linked Data [2] is continuously growing and changing over time [3]. While in some cases only the latest version of datasets are required, there is a growing need for access to prior dataset versions for data analysis. For example, analyzing the evolution of taxonomies, or tracking the evolution of diseases in biomedical datasets.

Several approaches have already been proposed to store and query versioned Linked Datasets. However, surveys [4, 5] have shown that there is a need for improved versioning capabilities in the current systems. Existing solutions either perform well for versioned query evaluation, or require less storage space, but not both. Furthermore, no existing solution performs well for all versioned query types, namely querying at, between, and for different versions.

In recent work, we introduced a compressed RDF archive indexing technique [6]— implemented under the name of OSTRICH— that enables highly efficient triple pattern-based versioned querying capabilities. It offers a new trade-off compared to other approaches, as it calculates and stores additional information at ingestion time in order to reduce query evaluation time. This additional information includes pointers to relevant positions to improve the efficiency of result offsets. Furthermore, it supports efficient result cardinality estimation, streaming results and offset support to enable efficient usage within query engines.

The Mighty Storage Challenge (MOCHA) (https://project-hobbit.eu/challenges/mighty-storage-challenge2018/) is a yearly challenge that aims to measure and detect bottlenecks in RDF triple stores. One of the tasks in this challenge concerns the storage and querying of versioned datasets. This task uses the SPBv [7] benchmark that consists of a dataset and SPARQL query workload generator for different versioned query types. All MOCHA tasks are to be evaluated on the HOBBIT benchmarking platform (https://project-hobbit.eu/). SPBv evaluates SPARQL queries [8], hence we combine OSTRICH, a versioned triple index with triple pattern interface, with Comunica [9], a modular SPARQL engine platform.

The remainder of this paper is structured as follows. First, the next section briefly introduces the OSTRICH store and the Comunica SPARQL engine. After that, we present our preliminary results in Section 3. Finally, we conclude and discuss future work in Section 4.

2. Versioned Query Engine

In this section we introduce the versioned query engine that consists of the OSTRICH store and the Comunica framework. We discuss these two parts separately in the following sections.

2.1. OSTRICH

OSTRICH is the implementation of a compressed RDF archive indexing technique [6] that offers efficient triple pattern queries in, between, and over different versions. In order to achieve efficient querying for these different query types, OSTRICH uses a hybrid storage technique that is a combination of individual copies, change-based and timestamp-based storage. The initial dataset version is stored as a fully materialized and immutable snapshot. This snapshot is stored using HDT [10], which is a highly compressed, binary RDF representation. All other versions are deltas, i.e., lists of triples that need to be removed and lists of triples that need to be added. These deltas are relative to the initial version, but merged in a timestamp-based manner to reduce redundancies between each version. In order to offer optimization opportunities to query engines that use this store, OSTRICH offers efficient cardinality estimation, streaming results and efficient offset support.

2.2. Comunica

Comunica [9] is a highly modular Web-based SPARQL query engine platform. Its modularity enables federated querying over heterogeneous interfaces, such as SPARQL endpoints [11], Triple Pattern Fragments (TPF) entrypoints [12] and plain RDF files. New types of interfaces and datasources can be supported by implementing an additional software component and plugging it into a publish-subscribe-based system through an external semantic configuration file.

In order to support versioned SPARQL querying over an OSTRICH backend, we implemented (https://github.com/rdfostrich/comunica-actor-rdf-resolve-quad-pattern-ostrich) a module for resolving triple patterns with a versioning context against an OSTRICH dataset. Furthermore, as versions within the SPBv benchmark are represented as named graphs, we rewrite these queries in a separate module (https://github.com/rdfostrich/comunica-actor-query-operation-contextify-version) to OSTRICH-compatible queries in, between, or over different versions as a pre-processing step. Finally, we provide a default Comunica configuration and script (https://github.-

com/rdfostrich/comunica-actor-init-sparql-ostrich) to use these modules together with the existing Comunica modules as a SPARQL engine. These three modules will be explained in more detail hereafter.

OSTRICH Module OSTRICH enables versioned triple pattern queries at, between, and for different versions. These query types are respectively known as Version Materialization (VM), Delta Materialization (DM) and Version Querying (VQ). In the context the SPBv benchmark, only the first two query types (VM and DM) are evaluated, which is why only support for these two are implemented in the OSTRICH module at the time of writing.

The OSTRICH Comunica module consists of an actor that enables VM and DM triple pattern queries against a given OSTRICH store, and is registered to Comunica's `rdf-resolve-quad-pattern` bus. This actor will receive messages consisting of a triple pattern and a context. This actor expects the context to either contain VM or DM information, and a reference to an OSTRICH store. For VM queries, a version identifier must be provided in the context. For DM queries, a start and end version identifier is expected.

The `rdf-resolve-quad-pattern` bus expects two types of output:

1. A stream with matching triples.
2. An estimated count of the number of matching triples.

As OSTRICH enables streaming triple pattern query results and corresponding cardinality estimation for all query types, these two outputs can be trivially provided using the JavaScript bindings for OSTRICH (https://github.com/rdfostrich/ostrich-node).

Versioned Query Rewriter Module The SPBv benchmark represents versions as named graphs. Listing 1 and Listing 2 respectively show examples of VM and DM queries in this representation. Our second module is responsible for rewriting such named-graph-based queries into context-based queries that the OSTRICH module can accept.

```
SELECT ?s ?p ?o WHERE {
  GRAPH <http://graph.version.4> { ?s ?p ?o }
}
```

Listing 1: Version Materialization query for the `?s ?p ?o` pattern in version `http://graph.version.4` in SPV's named graph representation.

```
SELECT * WHERE {
  GRAPH <http://graph.version.4> { ?s ?p ?o } .
  FILTER (NOT EXISTS {
    GRAPH <http://graph.version.1> { ?s ?p ?o }
  })
}
```

Listing 2: Delta Materialization query for the `?s ?p ?o` pattern to get all additions between version `http://graph.version.1` and `http://graph.version.4` in SPV's named graph representation.

In order to transform VM named-graph-based queries, we detect `GRAPH` clauses, and consider them to be identifiers for the VM version. Our rewriter unwraps the pattern(s) inside this `GRAPH` clause, and attaches a VM version context with the detected version identifier.

For transforming DM named-graph-based queries, `GRAPH` clauses with corresponding `FILTER-NOT EXISTS-GRAPH` clauses for the same pattern in the same scope are detected. The rewriter unwraps the equal pattern(s), and constructs a DM version context with a starting and ending version identifier. The starting version is always the smallest of the two graph URIs, and the ending version is the largest, assuming lexicographical sorting. If the graph URI from the first pattern is larger than the second graph URI, then the DM queries only additions. In the other case, only deletions will be queried.

SPARQL Engine The Comunica platform allows SPARQL engines to be created based on a semantic configuration file. By default, Comunica has a large collection of modules to create a default SPARQL engine. For this work, we adapted the default configuration file where we added our OSTRICH and rewriter modules. This allows complete versioned SPARQL querying, instead of only versioned triple pattern querying, as supported by OSTRICH. This engine is available on the npm package manager (https://www.npmjs.com/package/@comunica/actor-init-sparql-ostrich) for direct usage.

3. Evaluation

In this section, we introduce the results of running the SPBv benchmark on Comunica and OSTRICH.

As the MOCHA challenge requires running a system within the Docker-based HOBBIT platform, we provide a system adapter with a Docker container (https://github.com/rdfostrich/challenge-mocha-2018) for our engine that is based on Comunica and OSTRICH. Using this adapter, we ran the SPBv benchmark on our system on the HOBBIT platform with the parameters from Table 1.

Parameter	Value
Seed	100
Data form	Changesets
Triples in version 1	100,000
Versions	5
Version deletion ratio	10%
Version addition ratio	15%

Table 1: Configuration of the SPBv benchmark for our experiment.

For the used configuration, our system is able to ingest 29,719 triples per second for the initial version, and 5,858 per second for the following changesets. The complete dataset requires 17MB to be stored using our system. The initial version ingestion is significantly faster because the initial version is stored directly as a HDT snapshot. For each following changeset, OSTRICH requires more processing time as it calculates and stores additional metadata and converts the changeset to one that is relative to the initial version instead of the preceding version.

For the 99 queries that were evaluated, our system failed for 27 of them according to the benchmark. The majority of failures is caused by incomplete SPARQL expression support in Comunica, which is not on par with SPARQL 1.1 at the time of writing. The other failures (in task 5.1) are caused by an error in the benchmark where changes in literal datatypes are not being detected. We are in contact with the benchmark developer to resolve this.

For the successful queries, our system achieves fast query evaluation times for all query types, as shown in Table 2. In summary, the query of type 1 (queries starting with a 1-prefix) completely materializes the latest version, type 2 queries within the latest version, type 3 retrieves a full past version, type 4 queries within a past version, type 5 queries the differences between two versions, and type 8 queries over two different versions. Additional details on the query types can be found in the SPBv [7] article.

Query	Time	Results
1.1	34,071	141,782
2.1	49	128
2.2	59	32
2.3	27	12
2.5	233	969
2.6	1,018	19
3.1	18,591	100,006
2.6	230	46
4.1	37	91
4.2	43	16
4.3	21	2
4.5	197	708
4.6	1,119	25
5.1	13,871	59,229
8.1	59	171
8.2	56	52
8.3	31	22
8.4	44	0
8.5	709	2,288
8.6	8,258	346

Table 2: Evaluation times in milliseconds and the number of results for all SPBv queries that were evaluated successfully.

4. Conclusions

This article represents an entry for the versioning task in the Mighty Storage Challenge 2018 as part of the ESWC 2018 Challenges Track. Our work consists of a versioned query engine with the OSTRICH versioned triple store and the Comunica SPARQL engine platform. Preliminary results show fast query evaluation times for the queries that are supported. The list of unsupported queries is being used as a guideline for the further development of OSTRICH and Comunica.

During the usage of the SPBv benchmark, we identified several KPIs that are explicitly supported by OSTRICH, but were not being evaluated at the time of writing. We list them here as a suggestion to the benchmark authors for future work:

- Measuring storage size after each version ingestion.
- Reporting of the ingestion time of each version separately, next of the current average.
- Evaluation of querying all versions at the same time, and retrieving their applicable versions.
- Evaluation of stream-based query results and offsets, for example using a diefficiency metric [13].

In future work, we intend to evaluate our system using different configurations of the SPBv benchmark, such as increasing the number of versions and increasing the change ratios. Furthermore, we intend to compare our system with other similar engines, both at triple index-level, and at SPARQL-level.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American. 284, 28–37 (2001).
2. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - the story so far. Semantic Services, Interoperability and Web Applications: Emerging Concepts. 205–227 (2009).
3. Umbrich, J., Decker, S., Hausenblas, M., Polleres, A., Hogan, A.: Towards dataset dynamics: Change frequency of Linked Open Data sources. 3rd International Workshop on Linked Data on the Web (LDOW). (2010).
4. Fernández, J.D., Polleres, A., Umbrich, J.: Towards efficient archiving of Dynamic Linked Open Data. In: Debattista, J., d'Aquin, M., and Lange, C. (eds.) Proceedings of te First DIACHRON Workshop on Managing the Evolution and Preservation of the Data Web. pp. 34–49 (2015).
5. Papakonstantinou, V., Flouris, G., Fundulaki, I., Stefanidis, K., Roussakis, G.: Versioning for Linked Data: Archiving Systems and Benchmarks. In: BLINK ISWC (2016).
6. Taelman, R., Vander Sande, M., Verborgh, R.: OSTRICH: Versioned Random-Access Triple Store. In: Proceedings of the 27th International Conference Companion on World Wide Web (2018).

7. Papakonstantinou, V., Flouris, G., Fundulaki, I., Stefanidis, K., Roussakis, G.: SPBv: Benchmarking Linked Data Archiving Systems. In: Proceedings of the 2nd International Workshop on Benchmarking Linked Data and NLIWoD3: Natural Language Interfaces for the Web of Data (2017).
8. Harris, S., Seaborne, A., Prud'hommeaux, E.: SPARQL 1.1 Query Language. W3C, http://www.w3.org/TR/2013/REC-sparql11-query-20130321/ (2013).
9. Taelman, R., Van Herwegen, J., Vander Sande, M., Verborgh, R.: Comunica: a Modular SPARQL Query Engine for the Web. In: Proceedings of the 17th International Semantic Web Conference (2018).
10. Fernández, J.D., Martínez-Prieto, M.A., Gutiérrez, C., Polleres, A., Arias, M.: Binary RDF Representation for Publication and Exchange (HDT). Web Semantics: Science, Services and Agents on the World Wide Web. 19, 22–41 (2013).
11. Feigenbaum, L., Todd Williams, G., Grant Clark, K., Torres, E.: SPARQL 1.1 Protocol. W3C, http://www.w3.org/TR/2013/REC-sparql11-protocol-20130321/ (2013).
12. Verborgh, R., Vander Sande, M., Hartig, O., Van Herwegen, J., De Vocht, L., De Meester, B., Haesendonck, G., Colpaert, P.: Triple Pattern Fragments: a Low-cost Knowledge Graph Interface for the Web. Journal of Web Semantics. 37–38, (2016).
13. Acosta, M., Vidal, M.-E., Sure-Vetter, Y.: Diefficiency metrics: Measuring the continuous efficiency of query processing approaches. In: International Semantic Web Conference. pp. 3–19. Springer (2017).