OPAL
OPEN DATA PORTAL

Deliverable 7.2
Search Component Benchmark

Autoren: Caglar Demir
Reviewer: Adrian Wilke

Veröffentlichung	Öffentlich
Fälligkeitsdatum	31.12.2019
Fertigstellung	14.04.2020
Arbeitspaket	AP7
Typ	Bericht
Status	final
Version	1.0

Kurzfassung:
In diesem Deliverable wird die Durchführung eines Benchmarks der Datenhaltungslösungen Elasticsearch und dem RDF Triplestore Apache Fuseki beschrieben. Der empirische Vergleich der Laufzeiten zeigt, dass die neu eingesetzte Software Elasticsearch performanter als der vormals verwendete RDF Triplestore ist. Die durchgeführten Tests der Laufzeitmessungen ergeben einen Laufzeitgewinn des Faktors 20.

Schlagworte:
Benchmark, Elasticsearch, Apache Fuseki

mFUND
Das Startkapital für die Mobilität 4.0

Bundesministerium
für Verkehr und
digitale Infrastruktur

Inhalt

1 Introduction

One of the primary goals of OPAL is to extract metadata of datasets from different catalogs and ensure easy access to Open Data as specified in Deliverable D1.3 and D4.1. To attain such non-trivial tasks, Deliverable D4.2 highlighted the importance of unifying multi formatted datasets. In Deliverable D4.3, it has been shown that ensuring the accessibility of metadata stored in the form of RDF triples entails efficient storage and retrieval of RDF data. To facilitate efficient storage and retrieval, Deliverable D4.3 proposed to employ a novel RDF storage solution based on Elasticsearch and empirically evaluated the performance of the proposed approach with a state-of-the art approach (Virtuoso). In this work, we compare an RDF storage solution based on Elasticsearch against Apache Jena Fuseki. Our empirical comparison shows that the new storage solution used in OPAL outperforms Apache Jena Fuseki up to 20 factor faster w.r.t. the runtime in seconds. In Section 2, we give a brief overview pertaining to Elasticsearch and Apache Jena Fuseki. Following, we elucidate the evaluation setup in Section 3. Thereafter, we evaluate the competing approaches in Section 4. Finally, we conclude our work in Section 5.

2 Preliminaries

2.1 ElasticSearch

Elasticsearch is a distributed search engine based on Apache Lucene that provides full-text search engine over the stored data [1]. To provide fast querying over metadata of data sets, Elasticsearch provides a full-text search and it is equipped with tools such as Kibana and Metrics that enables system administrators to maintain the search system.

2.2 Apache Jena Fuseki

Apache Jena Fuseki is a SPARQL server that can either run as an operating system server, Java web application as well as a standalone server [2]. In our work, we employ Apache Jena Fuseki as a standalone server.

3 Evaluation Setup

3.1 Evaluation Scenario

We compare our data storage approach based on Elasticsearch against Apache Jena Fuseki. The used datasets comprise 58,368 DCAT datasets in Elasticsearch and 57,008 in Fuseki. Additionally to the datasets, DCAT distributions are included in the benchmark data. The difference of the dataset sizes are a result of necessary preprocessing steps. The benchmark is required to test the performance of simple queries for extraction to display the results at the user interface. To this end, we construct SPARQL Queries [4] based on our previous work D4.3. By constructing the SPARQL queries, we focus on the text queries as seen in Section 4. Next, we translate the SPARQL queries into the domain specific language (DSL) formatted used in Elasticsearch [5]. After we obtain a set of queries for our approach and for Fuseki, we perform each query in our approach and Fuseki. Performing a query consists of sending an HTTP request and receiving an HTTP response. We evaluate the performance of an RDF Triplestore w.r.t. the time elapsed between sending HTTP requests and receiving HTTP responses. To attain an empirical

comparison, we perform each query 100 times and report the average and the standard deviation of the time elapsed between HTTP requests and responses.

3.2 Implementation, Software Versions and Hardware

We implement our evaluation scenario in Python 3.6.4. All experiments were carried out on a virtual machine with Ubuntu 18.04, 16 GB RAM and four Intel Core i5-7300U CPU @2.60GHz processors. OPAL Github repository contains the Python implementation of evaluation, the third-party software components along with the software versions.

4 Results

We report the average and standard deviation of runtimes in seconds. Table 1 shows that our RDF Triplestore based on Elasticsearch significantly outperforms Fuseki w.r.t the runtime requirement. We perform each query 100 times in Elasticsearch and Fuseki and report the average and the standard deviation of the runtimes. Runtime of Elasticsearch (respectively Fuseki) denotes the time elapsed between sending HTTP requests and receiving HTTP responses. Given that both approaches run as standalone servers on the same hardware, RDF Triple store based on Elasticsearch outperforms Fuseki up to 20 factor. The outperformance of our approach stems from a full-text search capability of Elasticsearch and mapping structure used in Elasticsearch.

5 Conclusion

In this document, we evaluated the performance of the data storage solution used in OPAL against the performance of Fuseki. Our data storage solution based on Elasticsearch outperforms the Fuseki by up to factor of 20. The significant outperformance stems from Elasticsearch that scales to large numbers of metadata of datasets (documents in the terminology of Elasticsearch) while allowing flexible queries at retrieval of documents.

Query	Elasticsearch	Fuseki
SELECT (COUNT(distinct ?s) AS ?num) WHERE { GRAPH ?g {?s a dcat:Dataset } }	**.030** +- .002	.118 +-0.028

SELECT (COUNT(distinct ?s) AS ?num)
WHERE
{
GRAPH ?g {
?s a dcat:Dataset .
?s dct:title ?o .
FILTER isLiteral(?o)
FILTER contains(STR(?o), "Berlin")
}}
	.023 +- .002	.349 +- .156

SELECT (COUNT(distinct ?s) AS ?num)
WHERE
{
GRAPH ?g {
?s a dcat:Dataset .
?s dct:description ?o .
FILTER isLiteral(?o)
FILTER contains(STR(?o), "Baustelle")
}}
	.023 +- .002	.329 +- .058

SELECT (COUNT(distinct ?s) AS ?num) WHERE{
GRAPH ?g {
?s a dcat:Dataset .
?s dcat:keyword ?o .
FILTER isLiteral(?o)
FILTER contains(STR(?o), Bahnhof)}}
	.240 +- .002	1.234 +- .03564

SELECT (COUNT(distinct ?s) AS ?num)
WHERE
{
GRAPH ?g {
?s a dcat:Dataset .
?s dct:description ?o .
FILTER isLiteral(?o)
FILTER contains(STR(?o), "Berlin Flughafen")
}}
	.023 +- .002	0.4635 +- .1547

Table 1: The runtime results of Elasticsearch and Fuseki.

6 References

[1] What is Elasticsearch? - Amazon Web Services:
https://aws.amazon.com/elasticsearch-service/what-is-elasticsearch/

[2] Apache Jena Fuseki Documentation
https://jena.apache.org/documentation/fuseki2/

[3] Search Component Benchmark repository
https://github.com/projekt-opal/Search-Component-Benchmark

[4] SPARQL Query Language
https://www.w3.org/TR/rdf-sparql-query/

[5] Query DSL
https://www.elastic.co/guide/en/elasticsearch/reference/7.6/query-dsl.html