Evaluating an Elasticsearch-backed Triple Store

Ethan Rambacher, Rensselaer Polytechnic Institute

Background

Elasticsearch is a scalable document database that implements an inverted index for fast exact matches. An RDF triple store backed by Elasticsearch was implemented using the Apache Jena RDF API. By using Elasticsearch, we aim to create a scalable triple store capable of fast exact query matches.

Motivation

This project aims to determine if Elasticsearch is viable for use as a triple store for realistic use cases.

What is a triple store?

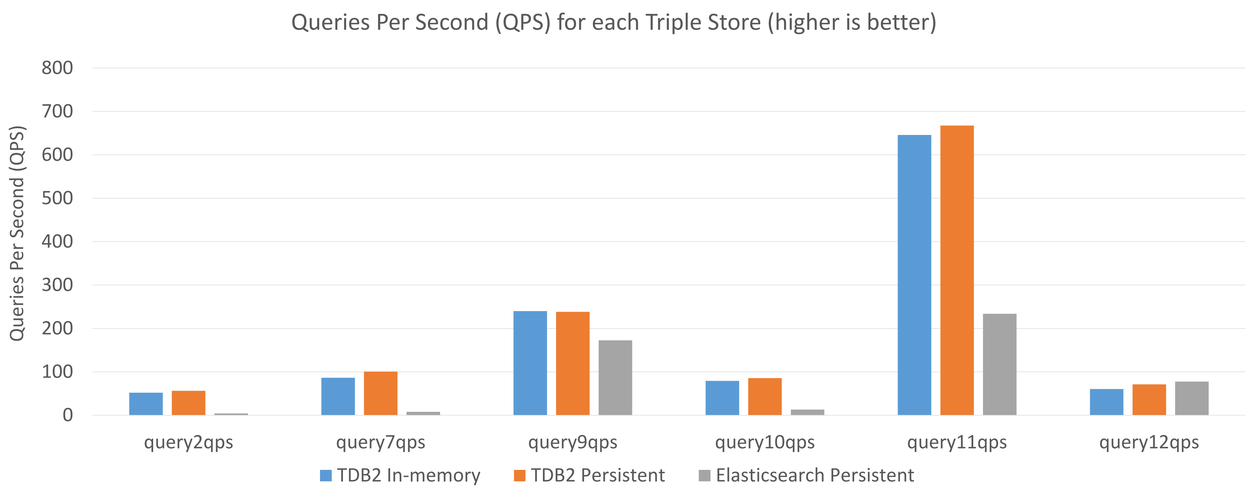
An RDF triple is a statement that consists of three parts – a subject, a predicate, and an object. An RDF graph is a set of multiple triples. RDF triples and graphs can be stored in dedicated triple stores that are designed to store and access triples.

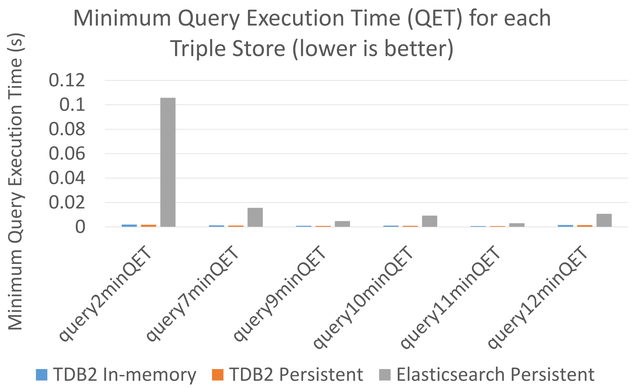
Design

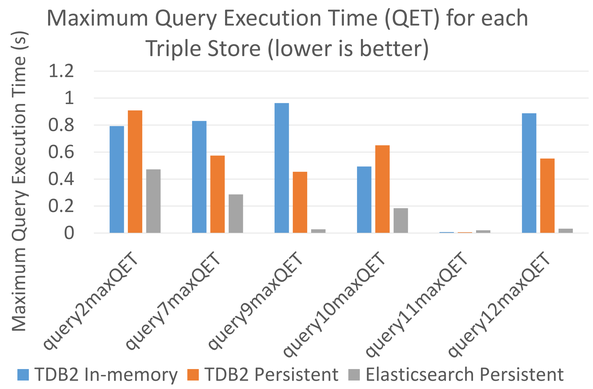
- Each RDF graph is a separate Elasticsearch index
- Synchronous and asynchronous operational modes for read visibility after update
- Each triple is stored as a single document with three keyword fields (subject, predicate, object)

Methods

- Apache Jena RDF API provides consistent API for manipulating triple data
- Apache Jena Fuseki used to publish SPARQL endpoint
- Evaluated the Elasticsearch triple store using Berlin SPARQL Benchmark (BSBM):
  - Evaluates performance of a SPARQL endpoint
  - Designed to mimic the “explore use case” of a customer using a faceted search to find a product
  - Data generation tool scales data based on number of products







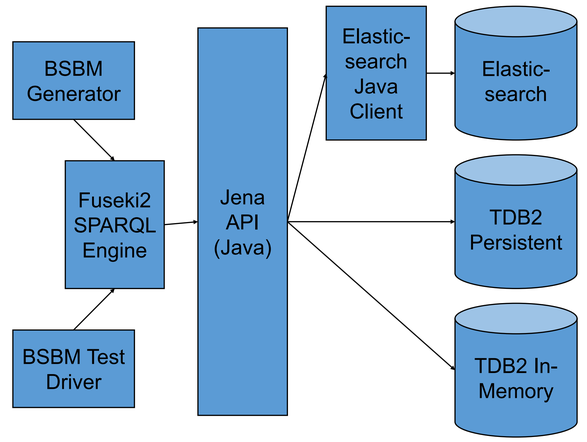
Benchmarking Parameters

- Benchmarking dataset consisted of 40,377 triples about 100 products
- Benchmarked against Jena TDB2 in-memory and persistent stores

Discussion of Results

- Elasticsearch triple store exhibits slower minimum query execution time (QET) on all queries
- Elasticsearch triple store exhibits faster maximum QET on all queries except query 11
- Some performance loss can be explained by Elasticsearch running in a separate process from Fuseki
- Current implementation is not yet optimized

Query	Description	Properties
2	Retrieve basic information about a single product	Small amount of data, large set of triple patterns
7	Retrieve in-depth information about a single product	Large amount of data
9	Get all information about a reviewer	Uses DESCRIBE
10	Get offers for a product	Uses DISTINCT, ORDER BY, LIMIT
11	Get all information about an offer	Similar to 9, but w/o DESCRIBE
12	Export information about an offer	Large amount of data; uses CONSTRUCT



Challenges

- Elasticsearch is more suited to asynchronous operation, so forcing data refreshes is costly
- Returning a large number of triple results is very costly

Future Work

- Bulk data ingest
- Benchmark against another triple store such as Blazegraph or Neptune that runs in a separate process from Fuseki
- Benchmark performance in distributed architecture
- Benchmark ingest performance

Acknowledgments

Minor Gordon, Deborah McGuinness from the Tetherless World Constellation at RPI.

References

1. C. Bizer and A. Schultz, “The Berlin SPARQL Benchmark,” Semantic Services, Interoperability and Web Applications, pp. 81–103, 2011.

Rensselaer
why not change the world?®

TetherlessWorld