Measuring the Usability of Triple Stores for Knowledge Management on Trauma Care Organizations

Joseph Utecht and Mathias Brochhausen PhD

Department of Biomedical Informatics, University of Arkansas for Medical Sciences, Little Rock, AR

Abstract. The CAFE project aims to provide a semantic web technology-based approach to compare the organizational structures of trauma centers and trauma systems. To achieve this we plan to use an RDF triple store that employs automatic inferences based on OWL representations. In order to engage users with the CAFE application real time feedback is a requirement. Although many RDF triple store performance measure have been published, there appears to be a gap when it comes to their use as the primary storage for real time applications. The performance needs for this use case differ from the triple stores more traditional use of offline reasoning and inference over large data sets. The objective of this research is to determine the feasibility of modern RDF triple stores as the primary storage for a real time application. To make this determination we measured the load time of just over one million triples of synthetic data and then ran various queries to emulate the types of demands this use would create. The resulting time measurements showed a difference in load times between the different stores and a large range of results in query performance. The lower end results are fast enough to compete with traditional relational databases while the results on the high end would not be suitable for this use case. The conclusion of this for the CAFE project is that we will be able to use a triple store, but we need to use the performance implications gained from this study to carefully tailor the queries we will run.

1 Introduction

Trauma Centers and trauma systems have been proven to increase positive outcomes in trauma care [1–3]. However, comparison of trauma centers and the patient outcome created by them is obstructed by a lack of comparability regarding the organizational components and the terms used to refer to organizational structures and roles. The NIH-funded project *Comparative Assessment For Environments of trauma care* (CAFE) (1R01GM111324-01A1) aims to develop a semantically driven, web-based IT framework to collect data about the organizational structure of trauma centers and systems to allow comparison of different organizational structures. To achieve that CAFE will create a knowledge resource based on an RDF triple store to store data about the organizational

structure of both trauma centers and trauma systems. Knowledge about types of entities relevant to the domain will be provided by an ontology coded in Web Ontology Language 2 (OWL2) [1]. The Ontology of Trauma Care Organizational Structures is currently under development. This process is guided by domain experts guiding the domain analysis and reviewing the OWL file. Our system will allow users to enter data about the organizational structure of a trauma center or system. This information will be captured from a web application in RDF and the progress in entering data will be perceivable to user by the growing graphical representation of the organizational structure. In the next step the user can request a comparison of their organization with canonical organizations of the same type. The data about those organizations will be stored in the triple store. The system will run a set of queries that retrieve data allowing representations of the same organizational units described by the users data and their interrelations. After the process of evaluation the data entered will be added to the triple store for future usage.

The CAFE project requires the ability to insert, reason and query data in as close to real time as is possible so the user experience does not suffer. Research has shown that even small delays in a website while attempting to perform some task will greatly decrease the rate at which people complete said task [4]. We see two methods of accomplishing this: A) a relational database to serve the application that is populated through offline reasoning, or B) real time reasoning provided by an RDF triple store that is populated directly from the application. Method A is the more commonly applied however to simplify the architecture of CAFE we would like to use method B, where the triple store is utilized as the primary data storage. While other research has measured the performance of triple stores for reasoning or complex queries, that only covers half of our use case. We need to know the simple query throughput that modern triple stores can produce so that when designing the CAFE application we will know the upper limit of queries per page. In this paper we measure the performance of various RDF triple stores as they would be used to drive a real time application. We also compare the performance to that of a relational database.

2 Material and Methods

2.1 RDF Triple Stores

We have decided to focus on Apache Jena [2], Blazegraph [3], and Sesame [4] RDF stores for our testing due to their support for RDFS reasoning, open source licenses, ability to handle large datasets, and REST endpoints for interaction. [5]

[1] http://www.w3.org/TR/owl2-primer/
[2] https://jena.apache.org/
[3] http://www.blazegraph.com/bigdata
[4] http://rdf4j.org/

Testing was performed on a VirtualBox VM [5] with 2x 2.4 GHz 6-core Xeon processors and 8GB of ram. The operating system was CentOS 7 [6] and all of the stores were hosted on the Java application server Tomcat [7].

Apache Jena uses a Java based front-end called Fuseki distributed under the Apache License 2.0. Fuseki can either be run standalone or under a Java application server. For testing, the most recent version Fuseki 2.0.0 with Tomcat was used with RDFS level reasoning.

Systrap's Blazegraph (formerly known as Bigdata) is distributed under either the GPLv2 or a commercial license. It can also either be run as a standalone or in a Java application server. Blazegraph 1.5.1 under Tomcat was used for testing with a triple + inference graph. Blazegraph uses up to six indexes to store the data and has a mature query optimizer [6]. If inference is enabled Blazegraph will materialize the inferences with a context which allows them to be removed if the RDF classes are modified.

Much like Jena, Sesame is a framework that also includes a web front-end and native triple store. Sesame version 2.8.4 was used for testing with a native store and RDFS inference enabled. Sesame allows arbitrary indexes to be created when the graph is initialized but we kept the default two. Like Blazegraph, Sesame materializes RDF inference when data is added.

2.2 Testing Method

We used Lehigh University Benchmark (LUBM) [7] generated data to test performance and capability of the triple stores. LUBM was used to produce a synthetic dataset of arbitrary size, that represents people and activities in an academic setting, such as students, professors, classs, departments, and universities. We did not use the default testing queries with this dataset as they were more designed to measure the performance of OWL inference models.

To service a high performing client-side application the triple store must be able to both return queries quickly and in large volume. We built a testing framework to benchmark the triple stores, measuring two areas, data loading time and throughput of small queries.

Data load time was measured by converting the LUBM dataset into groups of 10 n3 formatted triples and then measuring the time it took to load one million of those groups. These were loaded through the HTTP REST interfaces for each of the stores and the times were measured from the python script that was loading them.

SPARQL queries were used to generate lists of URIs as a starting point for the small queries with which we were testing query runtime. To this end we queried for the URIs of all students, professors, classes, departments and universities. Once we had these we ran the five queries below 1,000 times through the HTTP REST interface measuring the response time with a python script.

[5] `https://www.virtualbox.org/`
[6] `https://www.centos.org/`
[7] `http://tomcat.apache.org/`

2.3 Relational Database Baseline

To have a baseline to measure the performance of the RDF triple stores we decided to use a relational database performing close to the same task. To accomplish this we converted a subset of the LUBM data into a relational format and loaded it into a MariaDB [8] installation on an identical VM to the triple stores. Once the data was converted and loaded, indexes were places upon the columns which would act as keys. To make the comparison closer, a miniature HTTP REST interface was created in Python's Flask framework [9]. This interface would take a key as an HTTP parameter and then run a SQL query to the database returning the results as JSON encoded data [10]. As we were not interested in attempting a comparison between the relative run times of SQL vs SPARQL queries only two simple SQL queries were used to determine the baseline time this relational database setup would return data.

2.4 Queries

To begin, a query to determine the baseline time that each system will be able to return a query is needed. Therefor we designed a query that would return a single object with a subject level lookup, which should be the fastest possible lookup operation for any store. STUDENT is replaced with the URI of a random student pulled from a list of all students.

Listing 1.1. Benchmark Query

```
PREFIX ub: <http://swat.cse.lehigh.edu/onto/univ-bench.owl#>
SELECT ?name
WHERE {
        <STUDENT> ub:name ?name .
      }
```

Query 1 simulates returning the details about an individual where fields are not required to be present. The *optional* keyword can potentially cause long runtimes so this is a naive approach to this query. In the query STUDENT is replaced with the URI of a random student pulled from a list of all students.

[8] https://mariadb.org/
[9] http://flask.pocoo.org/
[10] http://json.org/

Listing 1.2. Query 1

```
PREFIX ub: <http://swat.cse.lehigh.edu/onto/univ-bench.owl#>
SELECT ?name ?advisor ?email ?telephone
WHERE {
        optional {?x ub:name ?name .}
        optional {?x ub:advisor ?advisor .}
        optional {?x ub:email ?email .}
        optional {?x ub:telephone ?telephone .}
        values ?x { <STUDENT> }
      }
```

Because of the performance implications of the *optional* keyword a second method for querying of the same information was performed. Query 1.5 was actually 4 separate queries each asking for one piece of the original data returned by query 1. Time on this query was a measure of how long it took to perform all 4 queries.

Listing 1.3. Sample of Query 1.5

```
PREFIX ub: <http://swat.cse.lehigh.edu/onto/univ-bench.owl#>
SELECT ?name
WHERE {
        <STUDENT> ub:name ?name .
      }
```

Query 2 returns all of the URIs of the authors on a given paper. This is testing return time where multiple rows will be returned. PAPER is replaced with a random URI of a paper.

Listing 1.4. Query 2

```
PREFIX ub: <http://swat.cse.lehigh.edu/onto/univ-bench.owl#>
SELECT ?author
WHERE {
        ?author ub:publicationAuthor <PAPER> .
      }
```

Query 3 also returns multiple rows and also has a join involved. PROFESSOR is replaced with a random URI of a professor.

Listing 1.5. Query 3

```
PREFIX ub: <http://swat.cse.lehigh.edu/onto/univ-bench.owl#>
SELECT ?student
WHERE {
        ?student ub:takesCourse ?course .
        <PROFESSOR> ub:teacherOf ?course .
      }
```

Query 4 is a highly selective query to test inference as both ub:Faculty and ub:Student are inferred RDF-level relationships.

Listing 1.6. Query 4

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://swat.cse.lehigh.edu/onto/univ-bench.owl#>
SELECT ?student ?faculty
WHERE {
        ?student rdf:type ub:Student .
        ?faculty rdf:type ub:Faculty .
        ?student ub:advisor ?faculty
        values ?faculty { <PROFESSOR> }
       }
```

3 Results

The relational database baseline returned on average in 5ms for a single row lookup and 6ms for a join.

Table 1. Data load times for 10 triples

Store	Total	Mean	Stdev	Median
Jena	1776k	13	101	5
Blazegraph	39341k	309	429	12
Sesame	8414k	66	205	35

Measured in milliseconds

The results of the loading test show a large difference between the three storage models of the triple stores. Blazegraph which has the largest number of indexes takes the longest to insert new data into. Sesame and Jena have similar numbers of indexes, however Sesame is materializing the inferences and thus has slower load times.

Table 2. Mean runtime for 1,000 queries

Store	Baseline $\overline{x}$	Query 1 $\overline{x}$	Query 1.5 $\overline{x}$	Query 2 $\overline{x}$	Query 3 $\overline{x}$	Query 4 $\overline{x}$
Jena	8	4413	29	7	1678	151
Blazegraph	38	36	146	37	36	38
Sesame	8	253	29	8	8	7

Measured in milliseconds

Jena's performance was highly volatile based on the query. The baseline speed was very fast as expected from the low number of indexes and the lack of inference in the baseline query. The *optional* keyword in the first query is known

to cause potential slowdowns and takes a heavy toll on Jena. Query 3 also has very poor performance but the reason for this is not clear. The inference required for query 4 slows it down in comparison to both Blazegraph and Sesame as it must calculate the inference at query time.

Blazegraph's large number of indexes and well established query optimizer result in extremely stable query time. Blazegraph was the only store that did not take a performance hit from the *optional* keyword in the first query. There however appears to be a 30ms processing time on anything Blazegraph is doing, of which we were unable to locate the cause.

Sesame showed the best performance overall only having problems with the *optional* keyword in the first query.

4 Conclusion

This paper examined the performance of RDf triple stores for query throughput in a real time application and compared it to the performance of a relational database. We found that performance in two of the stores, Jena and Sesame, were within a few milliseconds as a highly optimized relational databases for some queries. Based on the performance we will move forward with our plans to use an RDF triple store as the primary storage for the real time web application in the CAFE project.

Acknowledgement Research reported in this publication was supported by the National Institute of General Medical Sciences of the National Institutes of Health under award number 1R01GM111324.

References

1. D. Demetriades, M. Martin, A. Salim, P. Rhee, C. Brown, J. Doucet, and L. Chan, "Relationship between American College of Surgeons trauma center designation and mortality in patients with severe trauma (injury severity score > 15)," *J. Am. Coll. Surg.*, vol. 202, pp. 212–215, Feb 2006.
2. T. L. Sanddal, T. J. Esposito, J. R. Whitney, D. Hartford, P. P. Taillac, N. C. Mann, and N. D. Sanddal, "Analysis of preventable trauma deaths and opportunities for trauma care improvement in Utah," *J Trauma*, vol. 70, pp. 970–977, Apr 2011.
3. R. J. Winchell, N. Sanddal, J. Ball, H. Michaels, C. R. Kaufmann, R. Gupta, T. J. Esposito, and H. Subacius, "A reassessment of the impact of trauma systems consultation on regional trauma system development," *J Trauma Acute Care Surg*, vol. 78, pp. 1102–1110, Jun 2015.
4. D. F. Galletta, R. M. Henry, S. Mccoy, and P. Polak, *Web site delays: How tolerant are users?* PhD thesis, 2002.
5. M. Voigt, A. Mitschick, and J. Schulz, "Yet another triple store benchmark? Practical experiences with real-world data," *CEUR Workshop Proceedings*, vol. 912, no. Sda, pp. 85–94, 2012.
6. G. B. Olivier Curé, *RDF Database Systems: Triples Storage and SPARQL Query Processing*. Morgan Kaufmann, 1st ed., December 2014.
7. Y. Guo, Z. Pan, and J. Heflin, "LUBM: A benchmark for OWL knowledge base systems," *Web Semantics*, vol. 3, no. 2-3, pp. 158–182, 2005.