Sharing statistics for SPARQL Federation optimization, with emphasis on benchmark quality

Kjetil Kjernsmo[1]

Department of Informatics, Postboks 1080 Blindern, 0316 Oslo, Norway
kjekje@ifi.uio.no

Abstract. Federation of semantic data on SPARQL endpoints will allow data to remain distributed so that it can be controlled by local curators and swiftly updated. There are considerable performance problems, which the present work proposes to address, mainly by computation and exposure of statistical digests to assist selectivity estimation.
For an objective evaluation as well as comparison of engines, benchmarks that exhaustively covers the parameter space is required. We propose an investigation into this problem using statistical experimental planning.

1 Motivation

Query federation with SPARQL, which is a standardized query language for the Semantic Web, has attracted much attention from industry and academia alike, and four implementations of basic query federation were submitted to the SPARQL 1.1 Working Group as input for the forthcoming work[1]. This feature was supported by many group members, and the Last Call working draft of the proposed standard was published on 17 November 2011.

While the basic feature set of the proposed standard can enable users to create federated queries, it is not of great use as it requires extensive prior knowledge of both the data to be queried and performance characteristics of the involved query engines. Without this knowledge, the overall performance is insufficient for most practical applications.

To aid optimization, SPARQL endpoints should expose details about both data and performance characteristics of the engine itself. The proposed work has two focal points: *Statistical digests of data for optimizations* and *benchmarking SPARQL engines*.

The focus on SPARQL benchmarking is not only motivated from the perspective of optimization, as I have found the current state of the art in SPARQL benchmarking lacking in its use of statistics. The emphasis in the present paper is on statistics in benchmarking with the purpose of providing a firmer foundation on which assertions about engine performance can be backed with evidence.

[1] http://www.w3.org/2009/sparql/wiki/Feature:BasicFederatedQuery

The prior art in database theory is extensive, but I intend to focus on aspects that sets SPARQL apart from e.g. SQL, like the quad model, that it is a Web technology, or that data are commonly very heterogenous.

I have not yet started to explore the scientific literature around SPARQL Federation in any depth as I am still in an early phase of my work. I am currently focusing my efforts on benchmarking. The long-term goal of my work is SPARQL Federation, but that is a minor concern in this paper. Overall, the expected contributions are:

- Benchmarks that are able to cover all realistic performance-influencing parameters for SPARQL engines and make it possible to weigh different parameters, as well as quantify unexplained differences.
- To assist optimization, enable endpoint service descriptions to expose:
  - performance characteristics,
  - statistical digests of data that are optimized with respect to size and query performance.
- SPARQL engine developers can use the benchmark to reliably quantify unexpected adverse effects from a given modification.
- New SPARQL engine users can identify key differences between different engines, and therefore be able to more wisely choose engine to use.

2 State of the Art and Open Problems

2.1 In SPARQL Federation

I take the state of the art in technology to be represented by the current basic SPARQL 1.1 Federated Query Working Draft[2]. In addition, many have implemented federation that doesn't require explicit references to service endpoints, e.g. [8]. A recent scientific treatment of the current specification is in [1]. In that paper, the authors also show an optimization strategy based on execution order of so-called well-designed patterns.

A recent review of the state of the art is in [4]. In addition, [8] proposes *bound joins* and proves they can dramatically reduce the number of requests to federation members that are needed, as well as the implementation of FedX.

It has been my intention to focus on the two problems listed in section 3.3.1 in [4], i.e. strike a balance between accuracy and index size, and updating statistics as data changes. Notably, histogram approaches generally suffer from the problem that they grow too large or become an insufficiently accurate digest, especially in the face of very heterogeneous data. [5] introduced QTrees, which may alleviate the problem of histogram size, but which may not solve it.

Therefore, the core problem is: How do we compute and expose a digest that is of optimal size for the query performance problem?

[2] http://www.w3.org/TR/2011/WD-sparql11-federated-query-20111117/

2.2 In Benchmarking

Numerous benchmarks have been developed for SPARQL, but [2] showed that currently most benchmarks poorly represent the typical data and queries that are used on the Semantic Web. Most recently, [6] addressed some of these problems by using real data and real queries from DBpedia. [7] has developed a benchmark for the federated case.

Current common practice in benchmarking SPARQL-enabled systems is to use or synthesize a certain dataset, then formulate a number of queries seen as representative of SPARQL use in some way. These queries are then executed, and some characteristic of performance is measured, for example the time it takes for the engine to return the full result. Since there is a certain randomness in query times, this process is repeated a number of times and an average response time is found. Different engines can be compared based on these averages.

In many cases, this is sufficient. Sometimes, one engine can execute a query in an order of magnitude faster than another. If this happens systematically for many different queries, there is hardly reasonable doubt as to which is faster. In most cases, the query response times differs little, however. Small differences may seem unimportant but may become important if they are systematic. Even if one engine is dramatically better than another in one case, small deficiencies may add up to make the other a better choice for most applications anyway.

In this case, we must consider the possibility that the random noise can influence the conclusions. Whatever metric is used, it should be treated as a *stochastic variable*. This opens new methodological possibilities, first and foremost using well-established statistical hypothesis testing or ranking rather than just comparing averages.

Furthermore, the current approach presents merely anecdotal evidence that one engine is better than another with respect to performance. It may be that while the benchmarking queries do seem to favor one engine, other queries that have not been tried may reveal that there are strong adverse effects that may not have been anticipated. A more systematic approach is needed to provide a comprehensive and objective basis for comparing SPARQL engines.

In physical science and engineering, conventional wisdom has been that you should only vary one variable at a time to study the effects of that one variable. In medical science, this has been abandoned several decades ago, thanks to advances in statistics. In for example a case where the researcher administrates different treatments to terminally ill patients, some of which may be painful or shorten their lives, experimental economy is extremely important.

Using techniques from statistical experimental design, I propose that it is possible to design an experiment (i.e. a benchmark) which makes it possible to cover all realistic cases and with which we can justify why the remaining corner cases are unlikely to influence the result. For further elaboration, see Section 3.2.

So far, the benchmarking problem has been seen as a software testing problem, but as stated in the introduction this is not the only objective, we may now see if benchmark data can be exposed to help federation query optimizers along with a statistical digest.

The problems addressed by existing benchmarks such as the ones cited above are almost orthogonal to the problems considered by my proposed project. While I have seen some cases that compare performance based on box plots[3], it seems not to be common practice. Furthermore, I have not to date seen any work towards using methods like factorial designs to evaluate the performance of software, but there may be a limit in terms of complexity for where it is feasible, and I will restrict myself to SPARQL for this thesis.

3 Proposed Approach and Methodology

3.1 In SPARQL Federation

There are many possible approaches for this part of the thesis. As I expect substantial advances to be made before I start tackling this problem, I have not chosen any methodology, but an interesting direction for work seems to be to find more space-efficient ways to expose statistics in the service description and standardize them.

To this end, I have briefly looked into two approaches: [3] used Bayesian Networks and Probabilistic Relational Models to efficiently represent the joint distribution of database tables, a formalism that could be extended to RDF databases.

Another approach that I have not seen used in the literature is to use parametrized statistics. This would amount to an attempt to fit data to a known distribution function and expose which distribution and its parameters in the service description.

Finally, I have seen little work on the problem of rapidly changing data, so the adaption of existing techniques to such situations may also be interesting.

The evaluation methodology for the SPARQL federation work of the thesis will largely be covered by running the elaborate benchmark designs of the thesis.

3.2 In Benchmarking

Already in 1926, Ronald Fischer noted that complex experiments can be much more efficient than simple ones[4], starting the experimental design field. One of the simpler designs is "fractional factorial design", in which several "factors" are studied. In terms of SPARQL execution, the SPARQL engine is clearly a factor, but also, for example, the nestedness of `OPTIONAL`s can be a factor, or the number of triples in a basic graph pattern, etc. These numbers are varied to different "levels". The key to understanding why this can be efficient is that these variations need not occur in the same experiment. Therefore, for the SPARQL language, many combinations of factors can be studied by carefully designing queries to cover different factors, and a formalism called "resolution" has been

[3] See http://shootout.alioth.debian.org/ for example
[4] Cited in http://en.wikipedia.org/wiki/Factorial_experiment

developed to classify how well this has been achieved, partly answering the question of evaluation methodology for this part of the thesis. We should also validate by comparing this benchmark with conclusions from existing benchmarks.

A systematic approach based on statistical experimental design means developing methods to add factors (not only language features will be factors), and then use this methodology to add and then vary all the factors in an optimal fashion. Developing a software suite to perform most of the experiments is a key requirement as the number of experiments that results will be very large.

Factorial design inspires the present work and is covered in elementary textbooks in statistics but is inadequate for this purpose, so I intend to go further into the statistical literature to see if there is a methodology that is better suited to the problem. I also intend to use existing results on complexity bounds for SPARQL query answering to find suitable factors to see if my admittedly bold proposition that it is possible to design a benchmark to cover all realistic cases can be shown to hold.

With this analysis, I speculate based on superficial experience with factorial designs and analysis of variance that certain estimated coefficients can be exposed in the service description to give federated query engines assistance in optimizing for performance characteristics of certain SPARQL implementations.

This approach will yield the contributions listed above if completely successful, but will also advance the state of the art if only partially successful.

References

1. C. Buil-Aranda, M. Arenas, and O. Corcho. Semantics and optimization of the SPARQL 1.1 federation extension. In *The Semanic Web: Research and Applications*, LNCS 6644:1-15, 2011.
2. S. Duan, A. Kementsietsidis, K. Srinivas, and O. Udrea. Apples and oranges: a comparison of RDF benchmarks and real RDF datasets. In *Proc. of the 2011 Int. Conf. on Management of data*, SIGMOD '11, pages 145–156, 2011. ACM.
3. L. Getoor, B. Taskar, and D. Koller. Selectivity estimation using probabilistic models. In *Proc. of the 2001 ACM SIGMOD Int. Conf. on Management of data*, SIGMOD '01, pages 461–472, 2001. ACM.
4. O. Görlitz and S. Staab. Federated data management and query optimization for linked open data. In *New Directions in Web Data Management 1*, volume 331 of *Studies in Computational Intelligence*, pages 109–137. Springer, 2011.
5. A. Harth, K. Hose, M. Karnstedt, A. Polleres, K.-U. Sattler, and J. Umbrich. Data summaries for on-demand queries over linked data. In *Proc. of the 19th Int. Conf. on World wide web*, WWW '10, pages 411–420, 2010. ACM.
6. M. Morsey, J. Lehmann, S. Auer, and A.-C. N. Ngomo. DBpedia SPARQL Benchmark - Performance Assessment with Real Queries on Real Data. In *10th International Semantic Web Conference (ISWC2011)*, 2011.
7. M. Schmidt, O. Görlitz, P. Haase, G. Ladwig, A. Schwarte, and T. Tran. FedBench: a benchmark suite for federated semantic data query processing. In *Proc. of the 10th Int. Conf. on The semantic web - Volume Part I*, LNCS 7031:585–600, 2011.
8. A. Schwarte, P. Haase, K. Hose, R. Schenkel, and M. Schmidt. FedX: Optimization techniques for federated query processing on linked data. In *The Semantic Web - ISWC 2011*, LNCS 7031:601-616.