Studying Metadata for better client-server trade-offs in Linked Data publishing

Miel Vander Sande

Ghent University – iMinds
Sint-pietersnieuwstraat 41, B-9000 Ghent, Belgium
miel.vandersande@ugent.be

1 Problem statement

Since the introduction of the Semantic Web, querying Linked Data mostly utilizes two types of interfaces: Linked Data documents, or the SPARQL protocol. However, both do not cover the wide spectrum of possible use cases and their specific requirements. Not only is the amount of public SPARQL endpoints is limited, they also suffer from frequent downtime [6, 15]. Predicting the consumption of computational resources of an endpoint is hard, because of SPARQL's expressiveness and individual user demand. Linked Data documents are more predictable, but querying based on traversing links is significantly slower and renders less complete results. Unfortunately, both are very undesired for reliable user applications. These issues above hint at a need for other client/server trade-offs.

Such trade-offs can be analyzed using Linked Data Fragments (LDF) [17], which proposes an uniform view on all interfaces to RDF. A Linked Data Fragment is characterized by a specific **selector** (e.g., subject URI, SPARQL query), **metadata** (e.g., variable names, counts), and **controls** (e.g., links or URIs to other fragments). This reveals a complete spectrum between Linked Data documents and the SPARQL protocol, in which we can advance the state-of-the-art of Linked Data publishing. This spectrum can be explored in the following two dimensions: *i) selector*, allowing different, more complex questions for the server; and *ii) metadata*, extending the response with more information clients can use.

This work studies the second *metadata* dimension in a practical Web context. Considering the conditions on the Web, this problem becomes three-fold. First, analog to the Web itself, LDF interfaces should exist in a distributed, scalable manner in order to succeed. Generating additional metadata introduces overhead on the server, which influences the ability to scale towards multiple clients. Second, the communication between client and server uses the HTTP protocol. Modeling, serialization, and compression determine the extra load the overall network traffic. Third, with query execution on the client, novel approaches need to apply this metadata intelligently to increase efficiency.

Concretely, this work defines and evaluates a series of transparent, interchangeable, and discoverable interface features. We proposed Triple Pattern Fragments (TPF) [17], a Linked Data interface with low server cost, as a fundamental base. This interface uses a single triple pattern as selector. To explore this research space, we append this interface with different metadata, starting with an estimated number of total matching triples. By combining several TPFs, SPARQL

queries are evaluated on the client-side, using the metadata for optimization. Hence, we can study the impact of metadata on query execution time, bandwidth overhead, caching effectiveness, and server overhead.

2 Relevancy

The problem described in the previous section is relevant for both Linked Data consumption and publishing. Our approach specifically aims at introducing new client-server trade-offs. Thereby, our approach directly increases the granularity of engagement Linked Data publishers can take [13]. This ability to optimize between cost and utility, installs a lower threshold for publishing queryable Linked Data, ultimately leading to more available and easily consumable datasets.

In turn, this drastically increases reliability and strength of Linked Data based client applications [12]. Current infrastructure, such as SPARQL endpoints or data dumps, have proven to be insufficient to introduce major adoption in application development. This work facilitates reliable and dynamic data services in various domains, including eCommerce and public sector.

3 Related work

Many related works can be found in distributed databases [3] and hybrid query shipping [5, 7], which is already a very mature area. However, these works use a local dedicated network. Works that apply these techniques in a Web context and to Linked Data, which have different restraints, are still limited.

The works that do use metadata for SPARQL query optimization, either apply a centralized approach [11], or do not measure the process of metadata extraction and shipping (e.g., federated query systems) [1, 8, 14]. This is because most research considers the SPARQL specification a given,

Some work has been done on more specific types of metadata. Highly relevant is the proposal to extend the ASK query response [9] with a Bloom filter, representing a combinations of bindings, i.e. two variables in a triple pattern, to improve source selection in SPARQL query federation frameworks. However, the benefit in a single-server setup is unclear.

4 Research question(s)

This work seeks an answer to the following main research question:

How do different types of fragment metadata affect the relation between interface cost and utility with regard to client-side query execution?

In this respect, we also formulate the following subquestions for a series of selected types:

- Can such all selected metadata be modeled in RDF so it can be reconstructed on the client?
- How does out-of-band delivery of metadata, i.e. included in a separate HTTP resource, compare to in-band delivery, in terms of query execution time?

- What is the added server memory and CPU cost in generating such metadata?
- How does the type of metadata impact the shipping cost between server and client in a Web context?
- Can metadata decrease federated query execution time over multiple Linked Data sources?
- Can hypermedia to other relevant interfaces increase recall for federated queries?
- Can such metadata decrease the amount of HTTP requests used by the client-side query execution?

5 Hypotheses

In respect to the stated research questions, we formulate the following hypotheses:

- A client can reconstruct metadata described in a formalized vocabulary.
- Out-of-band delivery of metadata decreases query execution time compared to in-band.
- Generating metadata introduces an insignificant server cost compared to the total server cost.
- The metadata introduces a significant increase in shipping cost.
- Hypermedia can dynamically increase recall for queries federated of a Web of Data.
- Metadata significantly reduces the amount of HTTP request required by clients to answer a query.

6 Preliminary results

In the previous years, we have already conducted research with cardinality [18] and Approximate Membership metadata [16].

The results of the cardinality experiments indicate that, at the cost of increased query times, executing queries over TPF reduces server usage. TPF servers cope better with increasing numbers of clients than SPARQL endpoints. They have a generally low and regular CPU load, accompanied by less variation in response time. Furthermore, querying benefits strongly from regular HTTP caching, which can be added at any point in the network. These three facts validate that the interface reduces the server-side cost to publish knowledge graphs. This is all the more remarkable since, to allow comparisons with other work, these results were obtained with an *existing* SPARQL benchmark that focuses on performance, not server cost.

A second experiment validates that this behavior extends to real-world knowledge graphs such as DBpedia. A vast majority of queries stays well below the 1 second limit, despite being affected by the knowledge graph size. We note a strong influence of the type of query, especially when non-BGP SPARQL constructs are involved.

A third experiment shows that, although more compact formats show a decrease in query execution time, these findings no longer apply when responses are compressed by GZIP, commonly used within the HTTP protocol. Also, the

serialization and deserialization costs can be decisive, especially if they involve relatively few triples—which is the case for typical page sizes (e.g., 100) of a TPF interface. The experiment shows the importance of carefully considering serializations. Even though removing or shortening metadata and control triples would work for specialized TPF clients, the applicability of the application would be narrowed.

In terms of Approximate Membership metadata (AMFs), we augmented the TPF interface with both Bloom filters and Golomb-coded sets, which are two types of Approximate Membership Filters. We aimed at reducing HTTP requests by avoiding expensive triple membership checks, since for one third of a set of diverse query types, most of the request overhead are membership subqueries. At the expense of one extra request to fetch the approximate membership metadata, potentially many more could be saved. Indeed, the experimental results confirm a drastic decrease in requests for half of the 250 randomly generated WatDiv [2] queries, while others experience little overhead thanks to local caching. Furthermore, this addition does not affect the low-cost nature of the server, which only has a limited load increase. However, there is a computational overhead on the client for queries that are not improved. An intelligent client should minimize this, by deciding when to use membership metadata based on the query type.

Despite the reduction of requests, the total execution time is higher on average because of long delays introduced to generate AMFs. Therefore, we conclude that this metadata is not suitable for real-time computation. We therefore recommend to pre-compute or pre-cache it in advance. A strong benefit of HTTP caching has been proven for TPF querying [17] due to the limited possible number of requests, and this mechanism can be applied efficiently to TPFs with augmented metadata. While Bloom filters are preferred for lower computation time, the smaller size of Golomb-coded sets would prevail in the presence of caching. To prevent the overhead of generating and transferring AMFs, they could be served in a separate resource that clients explicitly request when needed.

7 Approach

Our approach defines a series of transparent, interchangeable, and discoverable interface features. These feature supply informative metadata, and can be ignored by the client if not needed. This process in split into five sequential steps, which are studied in this work: *selecting*, *generating*, *modeling*, *shipping* and *consuming*. The complete setup is illustrated in Figure 1.

7.1 Selecting metadata

This first step identifies candidate types to include in this work, originating from an extensive literature study (as briefly mentioned in Section 3). Good candidates are usable in the context of *i)* the RDF format, and *ii)* the Web, i.e. they are resistant to the delays, protocols and serializations. Thus, given these restrictions, we conduct a feature-based analysis to assess existing and novel metadata techniques for query optimization. For each selected metadata type, we compose a new interface based on the *Triple Pattern Fragments* interface.

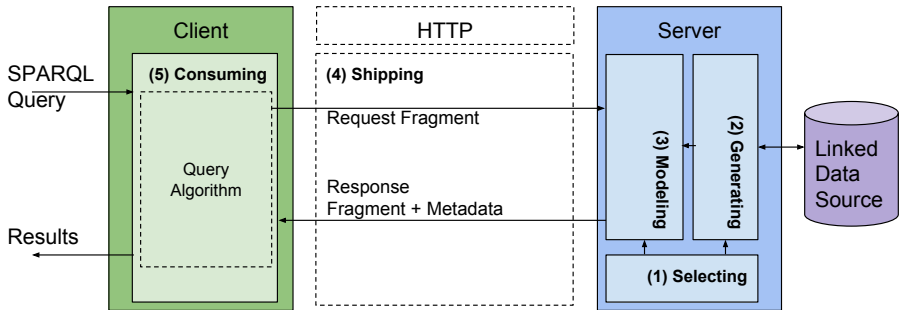

Fig. 1: Complete setup with 5 sequential steps that are subject to research.

As a primary focus, we selected four metadata types:

1. **cardinality**: the amount of triple patterns matches
2. **membership**: a compact representation of the set of matches
3. **summary**: a compact representation of the complete dataset
4. **discovery**: a set of links, i.e. hypermedia, to navigate to similar interfaces to retrieve more relevant data

However, future research may uncover new interesting types or variations that can be included in this research.

7.2 Generating metadata

Next, we study the methods that extract the necessary metadata. Important here is the introduced overhead on the server, which directly impacts the cost of hosting such interface reliably. Therefore, any extraction process should minimize its average CPU usage, relative to the overall CPU usage.

Accordingly, we propose evaluating the following algorithms to construct specific metadata from a existing knowledge base:

1. constructing a Approximate Membership filter (e.g., Bloom filter) from a fragment
2. profiling and summarizing an RDF dataset
3. triple pattern cardinality estimation
4. a summary index for constructing relevant hyperlinks to other interfaces.

7.3 Modeling metadata

To ensure the scalability of our approach in the distributed Web environment, we aim at loose coupling between client and server. Therefore, the notion of a *self-descriptive* interface is key. An RDF description on how the client should interpret the metadata is included in the server's response. Specifically, this is modeled using the Hydra core vocabulary from the Hydra W3C Community Group, augmented with VoID, a novel vocabulary for *Approximate Membership Filters*[1], and an adjusted data summary vocabulary loosely based on VoID.

[1] http://semweb.datasciencelab.be/ns/membership#

7.4 Shipping metadata

On the Web, shipping (meta)data from server to client, is subject to its characteristics: the HTTP protocol, the available network bandwidth, and the resource-oriented design.

Therefore, we study techniques that improve the effects of different metadata on *caching* and *response size*. The former determines how metadata should be embedded in the request-response cycle. For example, considering a single request, is the metadata supplied in-band or as a separate resource? The latter dictates download speeds, thus requires optimal serialization or compression.

7.5 Consuming metadata

Finally, we introduce techniques to improve client-side query execution using the metadata, provided by the interface. Improvements are made by *i)* lowering the number of required HTTP requests to solve a query; *ii)* improving the recall of query results by applying automatic dataset discovery.

8 Evaluation plan

The evaluation of our approach is specific to the task that a client needs to perform, i.e., the use case. For this work, we evaluate in context of client-side SPARQL querying, which is selected as main use case. Therefore, we rely on a few established query mixes for SPARQL endpoint testing:

1. the Berlin benchmark [4], for fair comparison with existing single-machine systems
2. the WatDiv benchmark [2], for in-depth analysis of the performance of specific query patterns
3. the DBpedia benchmark [10], for real-world scenarios
4. the FedBench benchmark, for measuring to what extend hypermedia can increase the recall of federated queries

We implement the aforementioned interfaces by extending the existing NodeJS server[2]. Also, we extend the NodeJS query client[3] to automatically discover all metadata and adjust the query execution accordingly.

Next, we have built a benchmarking tool that measures the following:

1. total query execution time
2. time to first result
3. response size
4. the number of HTTP requests
5. individual and average request duration
6. CPU and memory usage on both client and server

With this tool, we run all three query mixes in several iterations on datasets with different sizes and various HTTP cache setups. Results are compared against the TPF baseline (cardinality metadata) to assess the improvement, and against state-of-the-art SPARQL query systems.

[2] http://github.com/LinkedDataFragments/Server.js
[3] http://github.com/LinkedDataFragments/Client.js

9 Reflections

The practical aspects of Linked Data querying have been understudied so far. Focus has been on query execution time, precision and recall, while the feasibility of most of SPARQL and Linked Data query approaches is questionable. A Web context introduces many important characteristics, restrictions and opportunities, which are not mentioned or evaluated. As a result, we have not seen a widespread adoption of queryable Linked Data sources yet, or applications that rely on them.

LODstats (`http://stats.lod2.eu/`) counts 9960 Linked Datasets, of which only 187 endpoints exist error-free, which is only 0.02%. According to the more modern SPARQLES (`http://sparqles.ai.wu.ac.at/`), there are currently 535 endpoints, with currently only 44.67% with sufficient availability. Note that, although the number of endpoints has tripled since 2011, the availability rate has not improved. A recent count from LODLaundromat (`http://lodlaundromat.org/wardrobe/`), indicates that around 658,018 Linked Datasets exist, each of which is available as a Triple Pattern Fragments interface [13]. Thus, we can only conclude that relative number of endpoints is decreasing steadily, whereas the number of Triple Pattern Fragments interfaces is keeping up. This makes research for such lightweight interfaces important. The essence is that, by enabling more nuance and demanding less from servers, you can get more done with Linked Data.

References

1. Acosta, M., Vidal, M.E., Lampo, T., Castillo, J., Ruckhaus, E.: ANAPSID: An adaptive query processing engine for SPARQL endpoints. In: Proceedings of the 10th International Conference on The Semantic Web. pp. 18–34. ISWC'11, Springer-Verlag, Berlin, Heidelberg (2011), `http://dl.acm.org/citation.cfm?id=2063016.2063019`
2. Aluç, G., Hartig, O., Özsu, M.T., Daudjee, K.: The Semantic Web – ISWC 2014: 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I, chap. Diversified Stress Testing of RDF Data Management Systems, pp. 197–212. Springer International Publishing, Cham (2014), `http://dx.doi.org/10.1007/978-3-319-11964-9_13`
3. Bernstein, P.A., Goodman, N., Wong, E., Reeve, C.L., Rothnie Jr, J.B.: Query processing in a system for distributed databases (sdd-1). ACM Transactions on Database Systems (TODS) 6(4), 602–625 (1981)
4. Bizer, C., Schultz, A.: Benchmarking the performance of storage systems that expose sparql endpoints. World Wide Web Internet And Web Information Systems (2008)
5. Bowman, I.T.: Hybrid shipping architectures: A survey. University of Waterloo February (2001)
6. Buil-Aranda, C., Hogan, A., Umbrich, J., Vandenbussche, P.Y.: SPARQL Web-querying infrastructure: Ready for action? In: 12th International Semantic Web Conference (Nov 2013)
7. Franklin, M.J., Jónsson, B.T., Kossmann, D.: Performance tradeoffs for client-server query processing. In: ACM SIGMOD Record. vol. 25, pp. 149–160. ACM (1996)
8. Görlitz, O., Staab, S.: SPLENDID: SPARQL endpoint federation exploiting VOID descriptions. In: Proceedings of the 2nd International Workshop on Consuming Linked Data. Bonn, Germany (2011), `http://uni-koblenz.de/~goerlitz/publications/GoerlitzAndStaab_COLD2011.pdf`

9. Hose, K., Schenkel, R.: Towards benefit-based RDF source selection for SPARQL queries. Proc. of the 4th International Workshop on Semantic Web Information Management pp. 1–8 (2012)
10. Morsey, M., Lehmann, J., Auer, S., Ngonga Ngomo, A.C.: Dbpedia sparql benchmark–performance assessment with real queries on real data. The Semantic Web–ISWC 2011 pp. 454–469 (2011)
11. Neumann, T., Weikum, G.: x-RDF-3x: Fast querying, high update rates, and consistency for RDF databases. In: Proceedings of the International Conference on Very Large Data Bases. vol. 3, pp. 256–263. VLDB Endowment (Sep 2010)
12. Rietveld, L., Beek, W., Schlobach, S.: Lod lab: Experiments at lod scale. In: The Semantic Web-ISWC 2015, pp. 339–355. Springer (2015)
13. Rietveld, L., Verborgh, R., Beek, W., Vander Sande, M., Schlobach, S.: Linked data-as-a-service: The Semantic Web redeployed. In: Proceedings of the 12th Extended Semantic Web Conference (Jun 2015), http://linkeddatafragments.org/publications/eswc2015-lodl.pdf
14. Saleem, M., Khan, Y., Hasnain, A., Ermilov, I., Ngonga Ngomo, A.C.: A fine-grained evaluation of SPARQL endpoint federation systems. Semantic Web Journal (2014), http://svn.aksw.org/papers/2014/fedeval-swj/public.pdf
15. Schmachtenberg, M., Bizer, C., Paulheim, H.: Adoption of the linked data best practices in different topical domains. In: International Semantic Web Conference, pp. 245–260 (2014)
16. Vander Sande, M., Verborgh, R., Van Herwegen, J., Mannens, E., Van de Walle, R.: Opportunistic Linked Data querying through approximate membership metadata. In: Arenas, M., Corcho, O., Simperl, E., Strohmaier, M., d'Aquin, M., Srinivas, K., Groth, P., Dumontier, M., Heflin, J., Thirunarayan, K., Staab, S. (eds.) The Semantic Web – ISWC 2015. Lecture Notes in Computer Science, vol. 9366, pp. 92–110. Springer (Oct 2015), http://linkeddatafragments.org/publications/iswc2015-amf.pdf
17. Verborgh, R., Hartig, O., De Meester, B., Haesendonck, G., De Vocht, L., Vander Sande, M., Cyganiak, R., Colpaert, P., Mannens, E., Van de Walle, R.: Querying datasets on the Web with high availability. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C. (eds.) Proceedings of the 13th International Semantic Web Conference. Lecture Notes in Computer Science, vol. 8796, pp. 180–196. Springer (Oct 2014)
18. 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, 184–206 (2016), http://linkeddatafragments.org/publications/jws2016.pdf