Querying RDF data in Networks of Web Browsers

Arnaud Grall[1,2], Hala Skaf-Molli[1], and Pascal Molli[1]

[1] LS2N – University of Nantes, France
{arnaud.grall,hala.skaf, pascal.molli}@univ-nantes.fr
[2] GFI Informatique - IS/CIE, Nantes, France {arnaud.grall@gfi.fr}

Abstract. Web browsers represent an under-exploited data deposit. In this paper, we propose Snob, a decentralized query execution engine for SPARQL query execution over RDF data hosted in a P2P network of web browsers.

1 Introduction

Web Browsers are certainly the most deployed execution environment in the world and currently represent an under-exploited data deposit. Browsers are in direct contact with end-users, they are able to capture their location, their history of browsing and their perceptions of the real world. However, *querying data over a large-scale network of browsers is challenging.*

Many works address the problems of data management in Peer-to-Peer networks [6]. However, decentralizing data in browsers raises some specific issues. First, a network of browsers gathers a very large number of browsers with few RDF data hosted in browsers. This raises the problem of source selection when executing a SPARQL query over a very large number of relevant sources [3]. Second, a large number of browsers are now running on mobile phones. The query execution has to save bandwidth and battery. Finally, connections between browsers rely on the WebRTC[3] standard that does not have routing. Consequently, communication costs with distant neighbors are expansive. Therefore, during query execution direct neighbors must be considered as privileged relevant data sources.

In this paper [4], we propose Snob, an approach for executing SPARQL queries over a network of browsers[5]. We assume that browsers are connected in an unstructured network based on periodic peer-sampling [5], *i.e.*, each browser is connected to a bounded random subset of the network that is renewed periodically during a shuffling. A SPARQL query running in a browser can be seen as a federated SPARQL query executed after each shuffling where neighbors are the

[3] https://www.w3.org/TR/webrtc/

[4] This work was partially funded by the French ANR project O'Browser (ANR-16-CE25-0005-01). Mr. Grall is funded by the GFI Informatique compagny (145 Boulevard Victor Hugo, 93400, Saint Ouen, France).

[5] Snob is described with more details in [2]

data sources. As data sources are renewed every shuffling, the query execution will eventually continue to produce new results after each shuffling. This ensures that the number of messages exchanged with neighbors is constant per shuffling and every query makes progress at each shuffling. However, as progression can be slow, we build a second overlay network where browsers are connected to a fixed number of browsers that process similar queries.

This paper presents the following contributions: (i) Snob a SPARQL query execution model over RDF data hosted in a network of browsers. (ii) A semantic overlay network based on query containment. (iii) Experimentation shows that the number of results produced by queries grows with the number of running queries in the network. The semantic overlay network is able to speed up the number of produced results when fewer queries are running simultaneously.

2 The Snob Approach

Snob relies on four key ideas:

1. Browsers are connected through an unstructured network (RPS network) based on periodic peer-sampling [5], *i.e.*, periodically, each browser shuffles its local view on the network with the local view of its neighbors. This prevents the network to be partitioned. Compared to structured networks, unstructured network tolerates high churn and high expressiveness of queries [6][chapter 16].

2. A browser evaluates its queries as federated queries using only its direct neighbors as data sources. Later, it waits for the next shuffling that will bring new data sources. Such approach regulates the network traffic, *i.e.*, a browser can only send a number of messages bounded to the number of direct neighbors per shuffling.

3. To speed up query execution, we promote the sharing of intermediate results. Sharing intermediate results replicates and aggregates data. Consequently, the probability that a browser meets another one with relevant intermediate results after l shuffling is improved [4].

4. A Semantic Overlay Network (SON) [1] built on top of the unstructured network that selects as neighbors, for each browser, the k-best browsers processing similar queries. The profile of a browser is used for ranking them. A profile is the set of the triple patterns of the SPARQL queries under processing. The similarity of two profiles is based on triple patterns containment relationships ($\sqsubseteq$). Given 2 triple patterns tp_i, tp_j, we define a scoring function $S_t(tpi, tpj) : \mathbb{N}$ such that : $S_t(tpi \sqsubseteq tpj \wedge tpj \sqsubseteq tpi) \succeq S_t(tpi \sqsubseteq tpj) \succ S_t(tpi \sqsupseteq tpj)$. So given two browsers profiles B_i, B_j, we define a scoring function $S_b(B_i, B_j) = \sum_{tp_i \in B_i, tp_j \in B_j} S_t(tp_i, tp_j)$. After each shuffling, each browser recomputes its k-best neighbors with new neighbors discovered by the RPS.

Figure 1 shows browsers hosting data from Diseasome and Linked MDB. Browsers $B1 - B4$ executes queries over Diseasome, $B6 - B9$ over LinkedMDB and $B5$ over both. The RPS network ensures that all browsers are connected through a random graph, browsers profiles make the clustered network converging towards two communities. $B1 - B4$ will be highly connected because they

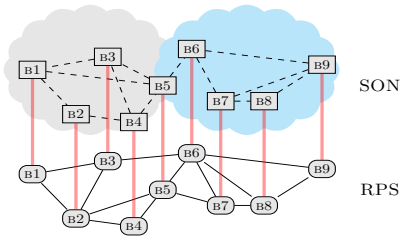

Fig. 1: Random Peer Sampling and Semantic Overlay Networks

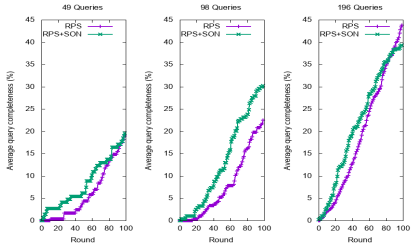

Fig. 2: Average query completeness.

execute queries over Diseasome, while $B6 - B9$ will be grouped together due their interest in LinkedMDB. $B5$ is connected to both communities.

Query Execution At a given time, a browser B_i has a fixed number of neighbors: k neighbors in the random peer sampling layer, and l neighbors in the semantic overlay network. Typical values of k and l are logarithmic to the size of the whole network. Each browser exposes to its neighbors a triple pattern interface. The browser is able to process an incoming triple pattern query and return results to the triple pattern query originator. Incoming triple pattern queries are processed over local data and intermediate results.

Suppose that the browser B_i processes the query Q_i. Q_i is processed *like* a federated query over its $k+l$ neighbors considered as data sources. In this setting, there are few differences with federated query processing: (i) The federation is incomplete, *i.e.*, the federation will change at the next shuffling. Therefore, B_i has to keep intermediate results for continuing processing when new sources are discovered. (ii) Shuffling could bring already visited data sources, even these browsers have been already visited, queries have to be re-executed. Because intermediate results hosted by visited browsers could have been changed.

As the size of local data is small in our context, after each shuffling, B_i processes as following: (i) for each triple pattern tp_j of query Q_i, evaluates tp_j over all its neighbors and updates its datasets. (ii) Execute Q_i and eventually produces new results. (iii) Sleep until next shuffling and only answers to triple pattern requests from direct neighbors.

3 Experimental Study

Snob source code is available at [6]. Snob uses RDFStore-js as a local data store and local query engine. We use the real datasets Diseasome[7] and Linked Movies Database[8] (LinkedMDB). We generated 100 queries per dataset using PATH and STAR shaped templates with two to eight triple patterns that are instantiated with random values from the dataset. We removed the queries that caused the

[6] `https://github.com/folkvir/webrtc-dequenpeda`
[7] `https://old.datahub.io/dataset/fu-berlin-diseasome`
[8] `http://data.linkedmdb.org/`

query engine to abort execution. This results in 100 queries from LinkedMDB and 96 queries from Diseasome. For these queries, we extracted triple patterns, each triple pattern of the query is executed as a SPARQL construct query and produces a fragment that we split into two sub-fragments. Sub-fragments are randomly distributed across the clients, each client hosts at least one fragment.

We set up a network of 196 clients in two configurations: one with only RPS where each client has 10 random neighbors, and another with RPS+SON where each browser has 5 random neighbors and 5 neighbors in SON. Results presented are the average of three successive executions of 100 rounds. A round corresponds to a query execution after a periodic shuffle where the shuffling time is set to 10 minutes. Figure 2 presents the pourcentage of completeness of query answers, i.e. the pourcentage of produced answers w.r.t the total number of answers, per round for three different configurations. The total number of queries answers is computed in a centralized setting where all data are loaded. Firstly for 49 queries (quarter), secondly for 98 queries (half) and finally for 196 queries (all). As we can see, RPS+SON provides better query completeness when less queries are running in the network. When every browser runs a query, then proportional replication of intermediate results is enough.

4 Conclusion

In this paper, we proposed Snob, a query execution model for SPARQL query over RDF data hosted in a network of browsers. Snob allows semantic application developpers to exploit RDF stored in browsers. This work opens several perspectives. First, data exchange between browsers has to be optimized when a data source is revisited to optimize the traffic. Second, intermediate results could be streamed on the semantic overlay network. Finally, a DHT service could be implemented in browsers to speed up the discovery of similar queries in the network.

References

1. Crespo, A., Garcia-Molina, H.: Semantic overlay networks for p2p systems. In: International Workshop on Agents and P2P Computing. pp. 1–13. Springer (2004)
2. Grall, A., Skaf-Molli, H., Molli, P.: SPARQL Query Execution in Networks of Web Browsers (Jun 2018), `http://hal.univ-nantes.fr/hal-01805154`
3. Grubenmann, T., Bernstein, A., Moor, D., Seuken, S.: Challenges of source selection in the wod. In: International Semantic Web Conference. pp. 313–328. Springer (2017)
4. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: Proceedings of the 16th international conference on Supercomputing. pp. 84–95. ACM (2002)
5. Nédelec, B., Tanke, J., Frey, D., Molli, P., Mostéfaoui, A.: An adaptive peer-sampling protocol for building networks of browsers. World Wide Web pp. 1–33 (2017)
6. Özsu, M.T., Valduriez, P.: Principles of distributed database systems. Springer (2011)