Introducing Collaborative Link Traversal Query Processing in the Context of Structured Decentralized Environments

Bryan-Elliott Tam

IDLab, Department of Electronics and Information Systems, Ghent University – imec

Abstract

Decentralized web environments aim to give users data autonomy and control. Data sovereignty focus on two aspects: privacy and provider choice. However, the concept remains incomplete if it fails to incorporate the actual utilization of the data. Specifically, in the context of application functionality, data sovereignty can be relinquished to the owner of the computational units or applications. The exploration and retrieval of information are core functionalities of web-based social applications because it is from those mechanisms that shared experiences to foster interactions are created. A promising example of web discovery techniques is Link Traversal Query Processing (LTQP), a SPARQL query paradigm that aims at exploring the web to answer queries by following the links between documents. In my doctoral research, I introduce Collaborative Link Data Query Processing, a paradigm where multiple query engines collaborate to improve query result completeness and execution performance in LTQP. I divide the research on the cooperation of query engines into two parts: 1) Improving the completeness of results, by exploring more of the search space, and 2) reducing the potentially long query execution time by caching results. To validate this proposal, I will develop a prototype and evaluate it using existing benchmarks. Based on my analysis of the state of the art, previous studies have made contributions to collaborative SPARQL query execution and RDF peer-to-peer caching. However, there is currently a research gap regarding the investigation of such systems in the context of LTQP within a structured decentralized environment.

1. Introduction

Decentralized web initiatives give users more control over their data. It can be formalized in a concept called data sovereignty. The authors of [1] defined this concept as “the power an individual has over their data”[1]. It can also be interpreted as “the self-determination of individuals and organizations concerning to the use of their data”[1], in practice it is the power of the user to choose “where [their] data is stored and who is granted access to it”[1].

In this PhD program, I attempt to move the definition of power and control, in the context of decentralized web environments, from a more consumer choice [2, 3, 4], the right to choose who will use my data, to a definition that includes the right of the users as a collective to make use of their data without relying on a third-party distributor with substantial computational power. In the context of data usage, sovereignty, and control are largely vested in the owner of the computational unit [5]. This ownership empowers them to make decisions that may diverge from the desires or interests of the users [4]. Additionally, it facilitates the utilization of users as products to be sold or as an unpaid

source of revenue [5], even when the data is anonymized [2, 4] and contribute to the centralization of wealth, increased surveillance and the non-representative distribution of contents online (for example in languages and type of content) [3, 4, 5]. To improve this sovereignty, I focus on data querying of social applications, applications that are driven by the interaction between users. In social applications, requesting and discovering new information, while taking into consideration complex concepts like serendipity [6], are the core functionalities creating the shared context that enable sociability. Concretely, I propose Collaborative Link Traversal Query Processing (CLTQP), a Link Traversal Query Processing (LTQP) paradigm where every user can share their computational power when querying using SPARQL. This collaboration aims to increase query result completeness and to reduce execution time for all users. With this collective participation, it would be possible at little cost to provide a more democratic economic base from which users could have more power to choose features for social applications that cater to their desire and interest and start social applications with a lesser monetary barrier. This paper is divided as follows, first, related work is presented, after the research proposal is made, then the methodology is explained and there is a short conclusion.

2. Related Work

2.1. Link Traversal Query Processing

LTQP is a technique that consists of recursively looking up URLs from dereferenced URIs acquired by the query engine to explore the surrounding information around the original response [7] using the follow-your-nose principle of Linked Data. In contrast, traditional SPARQL queries only query one document, and in Federated Query Processing the data sources need to be known beforehand. The query first starts with a small set of URIs called seed URIs [7] that form the initial data sources for the execution. The engine then uses a predefined lookup policy to discover new URIs inside the documents obtained from the seed URIs. Link traversal has a great exploratory potential by discovering unknown data sources during the querying execution with no input from the client [7]. The exploitation of this potential can extend the knowledge available for the engine to query or give more context to the data. In its traditional form, it consists of following, more or less naively, the links inside the response payload and dereferencing them to get new data sources. However, Link Traversal comes with some difficulties, such as the open-endedness of the web and the complexity of query planning [8]. Reachability criteria can be defined to restrict the links that are followed based on conditions. Classical examples are `cAll`, which follows every link, and `cMatch`, which follows links that match the query pattern [9].

2.2. Collaborative SPARQL Querying

Collaborative SPARQL Querying consists of using multiple agents to facilitate querying by diminishing the computation load of the execution or the discovery of data sources [10]. Snob [11] proposes a mechanism for collaborative query based on the continuous execution of queries over rotating browser data sources. The browsers form an unstructured peer-to-peer (P2P) network where each peer has a random and a profile-based

(browsers with similar profiles execute similar queries) partial view of the network. The browsers can share their intermediary results and at specific intervals, the peers are randomly shuffled by requesting new peers from known browsers. With this technique, it becomes possible to enhance the completeness of query results over time without the need to query every individual data source. ColChain [12] has a different approach. The query engines still have a partial view of the network but it is based on communities instead of being random or profile-based. A community, in the ColChain context, is a set of "nodes that participate in and observe [each other] and the fragments published [between them]" [12]. So the division of the network is made intentionally by the users. For the query process, the engine examines its own data sources and then expands the search to include the data sources of known communities. To discover new communities, the engine inquires peers to identify unknown communities. The collaborative aspect lies in the partition of a "global" knowledge graph into intentional semantic units. Other academic contributions have also aimed to leverage the social links between data sources to diminish the query execution time by not flooding the network when querying, such as in the article [13]. Another approach is to use the structure of the object modeled, such as academic papers as in the contribution of the authors of [14].

2.3. P2P Caching In The Context Of The Web

I define P2P caching as a particular case of collaborative querying where the query engine shares their already computed and valid results. Squirrel [15] proposed a P2P caching mechanism, where the URLs are mapped to keys inside a distributed hash table (DHT). If the user does not have the desired content in its local cache, it first queries the P2P network before requesting the URL. Squirrel does not propose a mechanism to take into consideration the distance between the client and the node's acting. Flower-CDN [16] proposes to modify the keys of the DHT to consider the locality. The content of the websites is distributed to peers of a locality, inside this locality a super-peer knows where to locate every content of all the websites of the locality. When a client makes a query to the DHT, the DHT directs to the super peer closest to the locality of the client and the super peer finds the content requested by the client. Behave [17] proposed another paradigm instead of using a structured network with a potentially slow DHT, it relies on an unstructured network where each peer has a partial view of the whole web. Each peer's view of the network is partially random and partially based on the websites visited to create a "behavioral locality" [17]. It uses a gossip protocol, at certain times the peers exchange randomly the nodes they know to change their view on the network. CyCLaDEs [18] adapted the concept of Behave for the use case of SPARQL query of RDF documents.

3. Problem Statement

3.1. Proposal

This paper aims to create a SPARQL query paradigm called Collaborative Link Traversal Query Processing. It consists of using multiple SPARQL query engines with the aim of improving the completeness of queries by exploring more of the search space and

reducing the query execution time through the means of exchange of results. Both problems have been engaged in the academic literature, but not in the case of LTQP and considering the distributed SPARQL querying domain [19]. In this PhD project, I will apply this query paradigm in the context of Solid, which implies that there is a strong consideration for privacy during querying [20] and a structured environment that can be leveraged to speed up the query [21].

The first problem I aim to solve is to increase the query completeness. In this context, I define the domain and the search space as a subset of a graph containing all the triple of the web of linked data that can be explore by the query engine. To increase the query completeness, CLTQP attempts to explore more of the search space by having multiple query engines engaging in non-overlapping partitions of the huge or pseudo-infinite search domain and executing the query. Hence, in the same amount of time, having more triples processed compared to an approach with just a single query engine. An important property emerging from the traversal of links is a structural proximity bias of the query results, which means that from the link traversal method, some data sources tend to be discovered more easily regardless of their potential influence on the query completeness and the interpretation of the query results. The bias has two interconnected sources: a sensitivity to the initial conditions induced by the seed URLs and the structure of the web, which is not a fully connected graph. Hence, a data source that takes more steps to be accessed, in regards to the seed URLs, can be more difficult to discover. Corollary, there is a bias based on the popularity of the data source, as it is easier to find a data source that is referenced more times and in a wide range of data source types (by data source types, I mean data sources that focus on specific topics) than data sources having the reverse properties. Hence by exploring more of the search space, there is more chance to discover those data sources. The second problem is to reduce the execution time, and I explore the method of P2P caching to alleviate this issue. Hartig in [22] demonstrates that caching in LTQP can help improve the completeness of results, however, in some conditions the query execution time can be increased. In the literature, there are contributions on the topic of P2P caching, but none engage with the problem of LTQP and its particularities, such as long execution time, exploration of multiple sources, and difficulty in attaining completeness which may change the conclusion of the caching and network strategy. Additionally, in environments like Solid, privacy is an additional consideration for caching [23]. For both problems, it has to be considered that a mechanism to incentivize reciprocity is necessary to ensure fairness and the good functioning of the P2P system. It can be implemented in multiple ways, for example, as an obligation to participate in a minimum percentage of queries or a number of links to provide. This enforcement could be managed by a community-specific structure with policies in that regard. Given that a user does not respect the policy, they cannot access the results of that community, so it is a form of social contract.

3.2. Research Questions And Hypotheses

Building on the proposal and the related work of Section 2, I formalize my research questions and hypotheses below:

- **Question 1**: Can we achieve better query result completeness and lower global query execution time in the context of LTQP by making multiple SPARQL query engines col-

laborate?

- **Question 2**: Does the CPU usages and the number of HTTP request for each engine diminish with the increase of engine collaborating?
- **Question 3**: How can we prevent multiple query engines from repeating identical calculations over the same data sources?
- **Question 4**: How can we reduce query execution time using P2P caching in the context of CLTQP?
- **Hypothesis 1**: Given a large enough search space, in which it is possible to split it between the engines, there is an inverse correlation between the number of engines collaborating and the execution time, and a direct correlation with the number of data sources explored.
- **Hypothesis 2**: It is possible to partition the search space in the context of CLTQP, in a way that the query processing time of overlapping data sources is less than the time to process distinct data sources.
- **Hypothesis 3**: It is possible to index a P2P cache and create a procedure for its usage in the context of CLTQP so that its lookup time is faster than the execution of the query.

Question 1 and 2, on the one hand, is the main question of my work, which aims at determining if CLTQP is a useful query paradigm. Questions 3 and 4, on the other hand, are asked to determine the efficiency of the specific method that will be employed to solve the two main problems. Those questions can be extended to consider the number of query engines and the types of scenarios (query, data sources, privacy policy, and so on) encountered. The hypotheses are the intuitive expected results and set a ground base for the development of my approaches.

4. Methodology

As discussed in the proposal, my work can be divided into two sub-problems; the division of the search space and caching. This implies two sets of potential solutions detailed below.

4.1. Search Domain Division Among Peers

My first aim is to increase the query completeness; For that purpose I will try to divide the search space among peers in a P2P network. It has to be considered first that the topology of the domain is not known, so we cannot divide the search space a priori. Below I present three strategies to divide the domain and process the query.

1. **Collect the seed URLs and divide them between the query engines**: The advantage of this strategy is the communication between the engines is minimal, at the start or at a moment when we have a large number of URLs we let the engines execute the query on their own and at the stopping condition, the engines share their results. The limitation of this strategy is that we don't consider if the data sources discoverable inside the seed URLs overlaps. Another important limitation is the loss of accuracy if part of the solution is present inside the documents processed by different engines. In that case, to find those results and retain the accuracy, the solution map would need to be joined.

By detecting those cases and using adaptative query planning [21, 24], it would be possible to change strategy to avoid the loss of accuracy.

2. **Set the reachability criteria of each engine so that they cannot or are less likely to have overlapping search field**: The advantage of this strategy is, also, the low communication between the engines. However, unlike the previous one, there is a mechanism to avoid redundant calculations. The query engines have a lookup policy that restricts links visited by others. For example, the engines might be responsible for a specific semantic section of the domain, e.g., cities in geospatial query. The limitations of this strategy are that the criteria might have to be changed depending on the executed query and the type of dataset from which we expect to find results ,and the loss of accuracy explained in the first strategy.
3. **Use a global link queue and solution map shared between all the query engines**: The advantage of this strategy is that it's a simple way to avoid redundant calculations as all engines have the same link queue. Also with the shared solution map the engines would avoid the problem of loss accuracy mention in the first strategy. Another possibility would be to let one peer do the join operation while the other peers handle the traversal and the execution of the query as inspired by the "slave-master" paradigm of the article [19]. The problem with this strategy is the necessary communication and the potential locking mechanism to avoid inconsistencies.

I am planning to build a prototype using the SPARQL meta query engine Comunica [25], because it already has LTQP algorithms implemented and it is a highly extensible software which will facilitate the implementation of those algorithms. I will evaluate it against the Solid social media benchmark; SolidBench [21] and compare the results with other LTQP approaches. I will evaluate those methods while varying the number of engines collaborating by increasing the number until the performance stagnates or diminishes. I propose to measure the following metrics:

- **Result accuracy**: The F-score; a fraction indicating the correctness and completeness of results
- **Query execution time**: The total time it takes to complete a query
- **The ratio of the execution time and the communication time between the engines**
- **Ability to access isolated documents**: Measured by analyzing the number of links leading to query-relevant data sources and evaluating their actual contribution
- **Overlapping of the search space**: The number of times a triple and data source has been queried
- **Query result arrival times**: The time it takes for each triple from the beginning of the query to be obtained [26]

4.2. Collaborative Caching

My second aim is to reduce the query execution time by using already computed results from a shared cache. The information cached could be the *data source URLs contributing to a query* and the *intermediary joint results with the associated solution map*, given some triple patterns to avoid their calculations. The cache could also be interpreted as a checkpoint for a longer execution or as a map of the data sources to explore. I

propose to investigate those two strategies:

1. **Unstructured network where peers are clustered based on their behavior**: The advantage of this strategy is that the lookup time to find information in the cache is constant and the peers have a high probability of possessing the knowledge desired. The clustering can be based on the engines that have engaged in a query collaboration with the subject engine. The disadvantage of that method is that it relies on a type of self-organization of the network of engines, and it does not consider that engines that have not collaborated might still have results in common.
2. **Distributed Hash Table to find the cached element**: The advantage of this approach is that we can get every cached element of the engines implementing the protocol. The disadvantage is that the lookup time is logarithmic with the number of elements cached, also the private information of the users will be dispersed in the DHT, with no regard to the will of the user [10], but there are strategies with the combination of a gossip protocol to keep privacy [23]. Also, an alternative would be to not share private information.

Building on the evaluation method of the first set of solutions, those metrics are added:

- **Cache access time**: The time it takes to retrieve information from the cache
- **Cache miss and cache hit rates**: The ratio of time the query engine get the information requested from the cache
- **Execution time reduction**: The ratio between the execution time of a query with and without the cache

5. Conclusion

In this article, I presented a proposal for my PhD research and the state of the art associated with it. My project aims at reducing the query execution time and increasing the query completeness when exploring the web of linked data. The solution proposed enables query engines to collaborate in the exploration of data sources and to share of results through a P2P cache. The potential of this proposal is to democratize the creation and usage of large-scale Web applications.

Supervisor: Pieter Colpaert, Ruben Taelman

Funding: Supported by SolidLab Vlaanderen (Flemish Government VV023/10) and the Research Foundation Flanders (FWO) under grant number S006323N

References

[1] Verstraete, M., Verbrugge, S., Colle, D.: Solid: Enabler of decentralized, digital platforms ecosystems. In: ITS (2022).
[2] Terranova, T.: Free Labor: Producing Culture for the Digital Economy. Social Text. (2000).
[3] Curran, J.: The internet of dreams Reinterpreting the internet. In: Misunderstanding the Internet (2016).
[4] Sevignani, S.: The commodification of privacy on the Internet. Science and Public Policy. (2013).
[5] Mechant, P., De Wolf, R., Van Compernolle, M., Joris, G., Evens, T., De Marez, L.:

Saving the web by decentralizing data networks? A socio-technical reflection on the promise of decentralization and personal data stores. In: 2021 14th CMI International Conference (2021).
[6] Smets, A., Michiels, L., Bogers, T., Björneborn, L.: Serendipity in Recommender Systems Beyond the Algorithm: a Feature Repository and Experimental Design. In: IntRS@RecSys (2022).
[7] Hartig, O., Özsu, M.T.: Walking Without a Map: Ranking-Based Traversal for Querying Linked Data. In: ISWC
[8] Hartig, O.: Linked Data Query Processing Based on Link Traversal. In: Linked Data Management
[9] Hartig, O., Freytag, J.-C.: Foundations of Traversal Based Query Execution over Linked Data. In: Conference on Hypertext and Social Media
[10] Grall, A., Skaf-Molli, H., Molli, P., Perrin, M.: Collaborative SPARQL Query Processing for Decentralized Semantic Data. In: Database and Expert Systems Applications. , Cham (2020).
[11] Grall, A., Skaf-Molli, H., Molli, P.: SPARQL Query Execution in Networks of Web Browsers. In: ISWC (2018).
[12] Aebeloe, C., Montoya, G., Hose, K.: ColChain: Collaborative Linked Data Networks. In: Proceedings of the Web Conference 2021 (2021).
[13] Shen, H., Lin, Y., Chandler, H.: An Interest-Based Per-Community P2P Hierarchical Structure for Short Video Sharing in the YouTube Social Network. IEEE. (2014).
[14] Jin, H., Yu, Y.: SemreX: a Semantic Peer-to-Peer Scientific References Sharing System. In: AICT-ICIW'06 (2006).
[15] Iyer, S., Rowstron, A.I.T., Druschel, P.: Squirrel: a decentralized peer-to-peer web cache. In: ACM SIGACT-SIGOPS (2002).
[16] Dick, M.E., Pacitti, E., Kemme, B.: Flower-CDN: a hybrid P2P overlay for efficient query processing in CDN. In: International Conference on Extending Database Technology (2009).
[17] Frey, D., Goessens, M., Kermarrec, A.-M.: Behave: Behavioral Cache for Web Content. In: IFIP (2014).
[18] Folz, P., Skaf-Molli, H., Molli, P.: CyCLaDEs: A Decentralized Cache for Triple Pattern Fragments. In: The Semantic Web (2016).
[19] Feng, J., Meng, C., Song, J., Zhang, X., Feng, Z., Zou, L.: SPARQL Query Parallel Processing: A Survey. In: IEEE International Congress on Big Data (2017).
[20] Taelman, R., Steyskal, S., Kirrane, S.: Towards Querying in Decentralized Environments with Privacy-Preserving Aggregation. ArXiv. (2020).
[21] Taelman, R., Verborgh, R.: Evaluation of Link Traversal Query Execution over Decentralized Environments with Structural Assumptions
[22] Hartig, O.: How Caching Improves Efficiency and Result Completeness for Querying Linked Data. In: LDOW (2011).
[23] Nilizadeh, S., Jahid, S., Mittal, P., Borisov, N., Kapadia, A.: Cachet: A Decentralized Architecture for Privacy Preserving Social Networking with Caching. In: Emerging Networking Experiments and Technologies (2012).
[24] Acosta, M., Vidal, M.-E., Lampo, T., Castillo, J., Ruckhaus, E.: ANAPSID: An Adaptive Query Processing Engine for SPARQL Endpoints. In: ISWC 2011. Springer Berlin Heidelberg (2011).
[25] Taelman, R., Van Herwegen, J., Vander Sande, M., Verborgh, R.: Comunica: a Modular SPARQL Query Engine for the Web. In: International Semantic Web Conference (2018).
[26] Acosta, M., Vidal, M.-E., Sure-Vetter, Y.: Diefficiency Metrics: Measuring the Continuous Efficiency of Query Processing Approaches. In: ISWC 2017 (2017).