Optimisation of Link Traversal Query Processing over Distributed Linked Data through Adaptive Techniques

Jonni Hanski

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

Abstract. An increasing amount of distributed Linked Data is being made available at different locations, with varying formats, structures, interfaces and availability. Making use of that data through declarative query languages such as SPARQL requires query engines capable of executing queries over it. Efficiently executing queries over the data requires efficient query plans, yet prior access to the information for producing such plans may not be possible due to the distributed and dynamic nature of the data. Furthermore, the inability to be aware of all data sources at a given time, following links to discover data in the form of link traversal may be needed. Consequently, query planning and optimisation may need to be performed with limited information, and the initial plan may no longer be optimal. Discovering additional information and data sources during query execution and adjusting the execution based on such discoveries using adaptive query processing techniques therefore could help perform queries more efficiently. The aim of this work is to explore a variety of existing or potential new techniques and their combinations for query-relevant information acquisition and query plan adaptation within the context of distributed Linked Data. Already prior results from multiple studies have demonstrated the benefits of various such techniques within or beyond Linked Data and Link Traversal Query Processing, and this work seeks to build upon such results to realise the benefits of various techniques in practice to tackle performance-related challenges.

1 Introduction

An increasing amount of data [28] is being made available on the Web following the Linked Data principles [14, 13]. The data is distributed for a variety of reasons ranging from availability and performance [5] to privacy and personal data management [39]. Making use of that data via abstraction layers in the form of query languages such as SPARQL [1] requires query engines capable of executing declarative queries over distributed Linked Data.

However, without knowledge of all data sources beforehand, approaches such as Link Traversal Query Processing (LTQP) [21, 23, 22, 35] that rely on the Linked Data principles are needed to discover data sources relevant for answering a given query. Additionally, studies have drawn attention to the variance in the reliability [3], availability [37], contents [28], access options [28] and transfer

rates [9] of distributed Linked Data sources, as well as the volume of the data [9]. Such variance results in a variety of challenges such as result completeness when querying a potentially infinite Web of Linked Data [23], as well as a number of query planning challenges with cardinality and selectivity estimation [27] and potential order-dependent selectivities within the data [17].

While some studies have demonstrated the limited impact of local result construction within LTQP scenarios [25], others [24, 35] have demonstrated how, even in a distributed environment with network latency, the query plan plays a vital role in efficient data access, depending on the use case. Unfortunately, the traditional optimise-then-execute approach relies on pre-computed statistics to provide sufficient information for producing an optimal query plan, and such an approach may produce sub-optimal plans in more dynamic environments or with limited information available [18], resulting in inefficient data access during query execution.

Within the context of distributed Linked Data, collecting and maintaining sufficient up-to-date information may not be feasible due to the variance described earlier, and some information such as data transfer rates, latencies or source availability may not be possible to know beforehand. Furthermore, within access controlled distributed environments such as Solid [39], the data and the information about it may vary depending on the access rights at the time of execution, and the owners of the data may modify it or change access permissons to it at any time. Maintaining up-to-date privacy-preserving aggregations or precomputed results on top of such data therefore requires further investigation, but through data discovery via link traversal, it will be possible to access the up-to-date data itself at any given time while respecting the data owner's access control policies.

Addressing the problems of query planning with initially missing information or statistics, unexpected correlations, unpredictable transfer rates and dynamically changing data, the concept of Adaptive Query Processing (AQP) has been introduced as a category of techniques to adapt query execution as new information becomes available [18]. Building on prior work, I will seek to apply such techniques and explore new ones to enable more efficient use of LTQP over distributed Linked Data.

Following this introduction, section 2 seeks to provide a brief overview of AQP, LTQP and other existing work, upon which the problem statement is built in section 3. The research methodology is outlined in section 4, evaluation plan in section 5, preliminary results are described in section 6 and conclusions in section 7.

2 State of the Art

The concept of Adaptive Query Processing (AQP) involves the adaptation of query execution based on runtime feedback, in an effort to find an execution plan that is well-suited to runtime conditions [18]. Such feedback could include, for example, information about the data being queried over [17], network la-

tency [36], source availability [3] or quality-of-service metrics [29]. To achieve this adaptivity, a query engine can interleave planning and execution as opposed to first planning and then executing with no interleaving [18]. For implementing such feedback and adaptivity, the concept of an adaptivity loop [18] has been identified, described as having four phases [18], with a total of five when considering pre-plan optimisation [29]:

1. **Plan pre-optimisation** based on information available at the time.
2. **Runtime monitoring** of parameters relevant to the goals of the system, and collection of metrics relevant for the analysis of the current plan and execution.
3. **Plan analysis**, to evaluate whether the goals are being met and if anything should be changed, and to determine whether re-optimisation should be done, taking into consideration the cost of re-optimisation and plan migration.
4. **Plan re-optimisation**, to find the next optimal plan, taking into consideration the techniques available for adjusting the plan and the points of it they allow modifications at, such as materialisation points or scheduling of operations.
5. **Actuation**, to implement the necessary changes from re-optimisation, including the migration of accumulated state inside various operators such as joins.

The remainder of this section will provide a brief overview of LTQP, dataset information acquisition methods and techniques in AQP.

2.1 Link Traversal Query Processing

Within the distributed Web of Linked Data [14, 13], knowing all relevant data sources or their supported access methods beforehand may not be possible. Furthermore, initiatives such as Solid [39] seek to distribute data into personal storages with access control [15], making available and accessible data dependent on current access permissions. Tackling some of the challenges, Link Traversal Query Processing (LTQP) [21, 23, 22, 35] offers a query execution paradigm that relies on only the basic Linked Data principles, by following links during query execution to discover data to query over. Unfortunately, such an approach, in addition to posing limitations with result completeness [23], also relies on some form of heuristics [21] or guidance [35] to efficiently discover relevant data through chains of links, as opposed to following all links in an arbitrary order. With no information on the data necessarily being queried over available beforehand in LTQP, producing optimal query plans prior to query execution becomes potentially infeasible, calling for adaptive approaches to adjust the execution as data is discovered.

2.2 Dataset Information Discovery

With query planning taking advantage of information on the data to produce optimal query plans [18], queries over distributed Linked Data benefiting from

information on data location [5], and with LTQP benefiting from guidance in selecting links to follow [35], acquiring such information is needed for the purposes of evaluating whether one query plan or execution approach is closer to optimal than another one, or if one data source or link should be prioritised over another. Various approaches have been investigated for making information about datasets available:

- **Dataset summaries**, such as Vocabulary of Interlinked Datasets (VoID) [8] for describing the access methods or structural information such as the total number of triples, distinct subjects, occurrences of predicates and similar.
- **Characteristics sets** [30] for estimating the cardinalities of large numbers of joins in bottom-up query processing, with techniqus for sample-based characteristics estimation [26] for datasets.
- **Approximate Membership Functions (AMFs)** [38], such as Prefix-Partitioned Bloom Filters (PPBFs) [5] and the extended Semantically Partitioned Bloom Filters (SPBFs) [7], already applied to reduce access to datasets known to not contain data relevant for answering a query [7, 34].
- **Locational Indexing** [5, 6], for efficiently locating data relevant for answering a query in a distributed environment.
- **Shape Trees** [31], **Solid Type Index** [42] or other techniques of summarising locations of data within a data source, to help efficiently locate relevant data.

2.3 Query Adaptation Techniques

Through adaptation of the query execution, advantage can be taken of newfound information. While the techniques proposed for implementing adaptivity differ in their level of interleaving between query execution, plan exploration and plan modification, two main approaches have been identified [18]:

- **Inter-query adaptivity** as the adaptation of subsequent query executions. While such an approach could be a natural next step for systems following the optimise-then-execute paradigm, imposing limited runtime overhead, it may prove insufficient or of limited benefit in environments where subsequent queries have little in common, or where the costs of operations or the characteristics of the data change frequently.
- **Intra-query adaptivity**, where the execution of a query is adapted on-the-fly during execution, calling for techniques that can accommodate such adaptation at runtime. This will also be the main focus of this work.

While different categorisations exist [29, 2, 27] for implementations of intra-query adaptivity, five major approaches could be identified across them as a compromise:

- **Operator-internal** techniques, using the implementation of a logical operator to achieve adaptivity, without requiring changes to the logical query plan. Techniques such as Symmetric Hash Join (SHJ) [41, 28], XJoin [36], MJoin [19] and Adaptive Group Join (agjoin) [3] could fit this category.

- **Data partitioning** techniques, to process different parts of the data differently in a fully pipelined environment, effectively applying different query plans on them, such as on a tuple level. Techniques such as Eddies [12, 2], State Modules (SteMs) [32] and STAIR [17] could fit this category.
- **Plan partitioning** techniques, to alter the plan at blocking operators or materialisation points, to delegate subplan selection from planning to execution phase, if sufficient information to select one is unavailable prior to execution [16].
- **Scheduling-based** methods, attempting to hide delays or produce results faster by rescheduling parts of the logical query plan based on intermediate results or unexpected delays. Techniques such as query plan scrambling [9] could fall within this category.
- **Redundant-computation** methods, to execute multiple plans simultaneously until the best-performing one is found and the others are terminated [11].

3 Problem Statement and Contributions

Building upon the existing work in section 2, this thesis will aim to overcome the challenges of traditional query processing outlined in section 1 through the use of adaptive techniques to achieve efficient Link Traversal Query Processing over dynamic distributed Linked Data with no prior knowledge of the data available prior to execution, while also being able to take advantage of any information made available through dataset descriptions. The research questions of this work are the following:

- **Question 1:** Assuming no prior knowledge about the data being queried over, how and what type of information can be discovered during query execution?
- **Question 2:** Taking advantage of newfound information during query execution, what type of new or existing techniques and approaches improve the query execution?
- **Question 3:** What kind of impact can be achieved through the various techniques? How do they affect the execution of the query, on their own or combined?

These research questions have inspired the following hypotheses:

- **Hypothesis 1:** It is possible to obtain sufficient information during query execution to produce an improved version of the initial plan. On average, it will take less time to migrate to the new plan and execute it than it would to finish with the initial one.
- **Hypothesis 2:** The execution time can be reduced by an order of magnitude by introducing a minimal amount of information about the data, for example approximate cardinality estimates.

- **Hypothesis 3:** The information used to guide query planning need not be accurate, as long as it is a step towards more accurate information compared to having no information available at all. That is, Hypothesis 2 holds even with inaccurate information.
- **Hypothesis 4:** Through the use of information available on the data being queried over, together with techniques in adapting the query execution, results can be produced not only faster overall, but also at a more stable rate over the execution of the query. That is, the delay between subsequent results remains, on average, constant. Without the techniques applied, this is expected to not be the case.

The base assumption behind the hypotheses is that, without any prior knowledge of the data, the query engine will produce sub-optimal query plans with trivial errors such as inefficient join orders. Thus, provided but a mere inkling of information slightly indicative of the correct direction, the engine shall already succeed in avoiding such trivial errors and perform an order of magnitude faster. Additionally, applying adaptive techniques should help alleviate any issues within a distributed environment and provide more efficient access to relevant data, helping the engine produce results both faster and at a more constant rate over the duration of the execution.

4 Research Methodology and Approach

Following section 3 and section 1, the aim of this work is to build upon existing research to enable efficient querying with LTQP over distributed Linked Data in practice with no prior knowledge available. This is to be achieved by applying techniques for discovering information about datasets at runtime, together with those for adapting the query plan to take advantage of that information.

The approach will involve the following steps, with prototype implementations and their evaluation outlined in further detail later in section 5, as the overall evaluation revolves around them:

1. **Review of techniques and approaches**, to discover existing ones using publications freely available online. This is to include publications both within and beyond the context of LTQP or distributed Linked Data, as deemed relevant, to discover also techniques not previously applied to LTQP, if any. For example, there may be techniques in querying over traditional relational databases or streams that could be of interest. Furthermore, standards, standards drafts and prototype implementations involving techniques of interest are to be considered.
2. **Identification of approaches of interest**, both existing and novel new ones as they are dicovered, and their **prototype implementations**. The aim is to establish an understanding of applying such techniques in practice.
3. **Evaluation of the techniques** based on their prototype implementations, to establish an understanding of the impact they have on their own and in combination with each other.

4. **Summarisation of results** throughout the process as they are discovered, to contribute to the field of Semantic Web research by providing insights into adaptive LTQP over distributed Linked Data.

The value of the research is to be found in the exploration of a variety of AQP techniques within LTQP over distributed Linked Data in particular. Such investigation has been deemed interesting in existing work [35]. The contributions of this work should help take the next steps towards the use of LTQP in practice, by providing meaningful results and reusable implementations for practical applications.

5 Evaluation Plan

For evaluating the elements of this work, various techniques will be applied through prototype implementations that are evaluated both on their own when possible and in combination with each other as deemed appropriate:

- **Prototype implementations** will be built upon a modular open-source SPARQL query engine [33]. Such a query engine has previously been used [35] in LTQP for evaluating approaches, and should therefore prove adequate for this purpose.
- **Evaluation** will be done using established benchmarking datasets such as the LDBC Social Network Benchmark [20, 10], having also been previously adapted for and used with LTQP [35] within the context of Solid [39]. Other benchmarks or reproducible real world datasets will also be considered as deemed interesting, assuming they can be adapted for use with Linked Data.

Drawing inspiration from existing work on LTQP, Linked Data and AQP, metrics such as the following are to be considered to understand the impact of the techniques:

- **Result completeness**, to ensure fair evaluation between techniques, should they produce different results. While not expected, this will be controlled.
- **Number of network requests** done by the query engine to produce a set of results. Existing work has demonstrated how the same set of results can be produced with fewer network requests using specific techniques [34, 38].
- **Server and client load**, in the form of processor and memory utilisation. Existing work has demonstrated how decreases in resource consumption on one side can be attained at the expense of the other side [40], and therefore any tests should take into consideration both sides.
- **Delay before initial query results**, as some techniques or implementations may result in delays prior to producing initial results [34].
- **Rate of producing results**, to determine how different techniques affect the intervals between intermediary results or the total execution time.

The last two metrics can be explored via the diefficiency [4] approach, designed to help measure the efficiency of a query engine over time. Such a technique should prove interesting to compare the impact of various techniques over time, and has also been previously applied [35] within the context of LTQP.

Through the observation of these metrics, it should be possible to confirm or reject specific instantiations of the hypotheses presented in section 3.

6 Preliminary Results

Thus far, preliminary results from an initial review of approaches are available, laying the foundation upon which to build the rest of the work. While an overview of the those results has been outlined in section 1 and section 2, the aim is to produce a comprehensive summary at a later stage. Envisioned immediate future work includes the use of some adaptations of Prefix-Partitioned Bloom Filters (PPBFs) [5] within the context of LTQP and Solid, as well as exploratory prototyping with an Eddies-inspired query engine architecture. Additionally, experiments are ongoing on the use of VoID [8] descriptions of datasets for join order optimisation, with mixed initial results using a two-phase join ordering methodology of initial zero-knowledge order followed by order based on initial information.

7 Conclusions

Through the application of dataset information discovery and AQP techniques, I aim to enable efficient LTQP over distributed Linked Data, without relying on pre-made indexes or summaries, although such indexes and summaries will still provide added value for query processing. This will ultimately enable more efficient use of resources throughout the Web of Linked Data, while also making the data more accessible.

Acknowledgements. The research for this work has been supported by Solid-Lab Vlaanderen (Flemish Government, EWI and RRF project VV023/10).

References

1. Sparql 1.1 overview. W3c recommendation, W3C (2013), https://www.w3.org/TR/sparql11-overview/
2. Acosta, M., Vidal, M.E.: Networks of linked data eddies: An adaptive web query processing engine for rdf data. In: The Semantic Web: ISWC 2015, Proceedings, Part I 14. pp. 111–127 (2015)
3. Acosta, M., Vidal, M.E., Lampo, T., Castillo, J., Ruckhaus, E.: Anapsid: an adaptive query processing engine for sparql endpoints. In: The Semantic Web: ISWC 2011, Proceedings, Part I 10. pp. 18–34 (2011)

4. Acosta, M., Vidal, M.E., Sure-Vetter, Y.: Diefficiency metrics: measuring the continuous efficiency of query processing approaches. In: The Semantic Web: ISWC 2017, Proceedings, Part II 16. pp. 3–19 (2017)
5. Aebeloe, C., Montoya, G., Hose, K.: Decentralized indexing over a network of rdf peers. In: The Semantic Web: ISWC 2019, Proceedings, Part I 18. pp. 3–20 (2019)
6. Aebeloe, C., Montoya, G., Hose, K.: Colchain: Collaborative linked data networks. In: Proceedings of the Web Conference 2021. pp. 1385–1396 (2021)
7. Aebeloe, C., Montoya, G., Hose, K.: The lothbrok approach for sparql query optimization over decentralized knowledge graphs. arXiv preprint arXiv:2208.14692 (2022)
8. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing linked datasets with the void vocabulary (2011), https://www.w3.org/TR/void/
9. Amsaleg, L., Tomasic, A., Franklin, M., Urhan, T.: Scrambling query plans to cope with unexpected delays. In: Fourth International Conference on Parallel and Distributed Information Systems. pp. 208–219 (1996). https://doi.org/10.1109/PDIS.1996.568681
10. Angles, R., Antal, J.B., Averbuch, A., Birler, A., Boncz, P., Búr, M., Erling, O., Gubichev, A., Haprian, V., Kaufmann, M., et al.: The ldbc social network benchmark. arXiv preprint arXiv:2001.02299 (2020)
11. Antoshenkov, G., Ziauddin, M.: Query processing and optimization in oracle rdb. The VLDB Journal **5**, 229–237 (1996)
12. Avnur, R., Hellerstein, J.M.: Eddies: Continuously adaptive query processing. In: Proceedings of the 2000 ACM SIGMOD international conference on Management of data. pp. 261–272 (2000)
13. Berners-Lee, T.: Design issues: Linked data (2000), https://www.w3.org/DesignIssues/LinkedData.html
14. Berners-Lee, T., Hendler, J., Lassila, O., et al.: The semantic web. Scientific American **284**(5), 28–37 (2001)
15. Capadisli, S., Berners-Lee, T.: Web access control (2022), https://solidproject.org/TR/wac
16. Cole, R.L., Graefe, G.: Optimization of dynamic query evaluation plans. In: Proceedings of the 1994 ACM SIGMOD international conference on Management of data. pp. 150–160 (1994)
17. Deshpande, A., Hellerstein, J.M., et al.: Lifting the burden of history from adaptive query processing. In: VLDB. pp. 948–959 (2004)
18. Deshpande, A., Ives, Z., Raman, V.: Adaptive query processing. Found. Trends databases **1**, 1–140 (01 2007). https://doi.org/10.1561/1900000001
19. Ding, L., Rundensteiner, E.A., Heineman, G.T.: Mjoin: A metadata-aware stream join operator. In: Proceedings of the 2nd international workshop on Distributed event-based systems. pp. 1–8 (2003)
20. Erling, O., Averbuch, A., Larriba-Pey, J., Chafi, H., Gubichev, A., Prat, A., Pham, M.D., Boncz, P.: The ldbc social network benchmark: Interactive workload. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. pp. 619–630 (2015)
21. Hartig, O.: Zero-knowledge query planning for an iterator implementation of link traversal based query execution. In: The Semantic Web: ESWC 2011, Proceedings, Part I 8. pp. 154–169 (2011)
22. Hartig, O.: Linked data query processing based on link traversal. (2014)
23. Hartig, O., Freytag, J.C.: Foundations of traversal based query execution over linked data. In: Proceedings of the 23rd ACM conference on Hypertext and social media. pp. 43–52 (2012)

24. Hartig, O., Heese, R.: The sparql query graph model for query optimization. The Semantic Web: Research and Applications pp. 564–578 (2007)
25. Hartig, O., Özsu, M.T.: Walking without a map: Optimizing response times of traversal-based linked data queries (extended version) (2016)
26. Heling, L., Acosta, M.: Estimating characteristic sets for rdf dataset profiles based on sampling. In: The Semantic Web: ESWC 2020, Proceedings. pp. 157–175 (2020)
27. Ives, Z.G., Halevy, A.Y., Weld, D.S.: Adapting to source properties in processing data integration queries. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data. pp. 395–406. SIGMOD '04 (2004). https://doi.org/10.1145/1007568.1007613
28. Ladwig, G., Tran, T.: Linked data query processing strategies. In: ISWC (1). pp. 453–469 (2010)
29. Liu, M.: Cost-Based Efficient Adaptive Query Processing for Data Streams. Ph.D. thesis, University of Pennsylvania (2012)
30. Neumann, T., Moerkotte, G.: Characteristic sets: Accurate cardinality estimation for rdf queries with multiple joins. In: 2011 IEEE 27th International Conference on Data Engineering. pp. 984–994 (2011)
31. Prud'hommeaux, E., Bingham, J.: Shape trees specification. W3c editor's draft, W3C (2020), https://shapetrees.org/TR/specification/
32. Raman, V., Deshpande, A., Hellerstein, J.M.: Using state modules for adaptive query processing. In: Proceedings 19th International Conference on Data Engineering. pp. 353–364 (2003)
33. Taelman, R., Van Herwegen, J., Vander Sande, M., Verborgh, R.: Comunica: a modular sparql query engine for the web. In: ISWC 2018, Proceedings (Oct 2018), https://comunica.github.io/Article-ISWC2018-Resource/
34. Taelman, R., Van Herwegen, J., Vander Sande, M., Verborgh, R.: Optimizing approximate membership metadata in triple pattern fragments for clients and servers. In: SSWS2020. vol. 2757, pp. 1–16 (2020)
35. Taelman, R., Verborgh, R.: Evaluation of link traversal query execution over decentralized environments with structural assumptions (2023). https://doi.org/10.48550/ARXIV.2302.06933, https://arxiv.org/abs/2302.06933
36. Urhan, T., Franklin, M.J.: Xjoin: A reactively-scheduled pipelined join operator (2000)
37. Vandenbussche, P.Y., Umbrich, J., Matteis, L., Hogan, A., Buil-Aranda, C.: Sparqles: Monitoring public sparql endpoints. Semantic web **8**(6), 1049–1065 (2017)
38. Vander Sande, M., Verborgh, R., Van Herwegen, J., Mannens, E., Van de Walle, R.: Opportunistic linked data querying through approximate membership metadata. In: The Semantic Web: ISWC 2015, Proceedings, Part I 14. pp. 92–110 (2015)
39. Verborgh, R.: Re-decentralizing the Web, for good this time. In: Seneviratne, O., Hendler, J. (eds.) Linking the World's Information: A Collection of Essays on the Work of Sir Tim Berners-Lee. ACM (2022), https://ruben.verborgh.org/articles/redecentralizing-the-web/
40. 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**, 184–206 (2016)
41. Wilschut, A.N., Apers, P.M.: Dataflow query execution in a parallel main-memory environment. Distributed and Parallel Databases **1**, 103–128 (1993)
42. Zagidulin, D., Sambra, A., Carvalho, M., Pavlik, E.: Solid application data discovery (2022), https://github.com/solid/solid/blob/main/proposals/data-discovery.md