Query Processing in Self-Organized Storage Systems

Hannes Mühleisen

Supervisor: Robert Tolksdorf - PhD Research Phase 1
Department of Computer Science - AG Networked Information Systems
Freie Universität Berlin - Königin-Luise-Str. 24/26, 14195 Berlin, Germany
muehleis@inf.fu-berlin.de

Abstract. Storage systems are increasingly approaching their limits regarding system response to node overload and failure as well as overall scalability. Self-organized systems can be a solution to those issues. However, query processing research has not yet evolved to this area. This research proposal aims at extending distributed query processing to self-organized systems. The different components are discussed, and evaluation criteria for a emerging solution are laid out.

1 Problem statement

Database management systems and storage systems in general have evolved over time to not only efficiently and securely store data on one computer, but also to store this data in a organized network of many computers, with different sharing architectures. Once data can be stored and - crudely - accessed, the need for more sophisticated query patterns arises quickly. These are then formulated using a variety of mostly declarative query languages. Well-known examples for such languages include SQL for relational databases, and SPARQL for RDF storage systems. While evaluating these queries on a single computer represents a significant challenge, performing the same task in a distributed environment is even more complex.

One central component of any storage system supporting complex query patterns is the Query Processor (QP), which takes the declarative complex queries issued by the user and transforms them over various steps into an Query Execution Plan (QEP). The QEP then contains a detailed plan how to access, reorder, and deliver the requested data to the client. This component relies on a map or other lookup mechanism to determine which data elements are stored on which node of the storage network [6, 10, 5]. If no such map or efficient lookup mechanism is available, query execution is severely impaired. The node handling the query evaluation can no longer efficiently compute a QEP due to a lack of knowledge of the topology of the data stored in the network. While the Peer-to-Peer (P2P) community has developed the concept of mutable query plans, these approaches are only viable in a structured P2P system so far [9, 3, 1].

A different approach to distributed storage systems are tuple-based self-organizing storage systems using swarm algorithms. Starting from an approach implementing the Linda coordination language, this method makes use of nature-inspired algorithms to efficiently route storage and retrieval requests in an unstructured network consisting of a potentially large amount of nodes. Systems built with these algorithms have the potential to scale without inherent limits, and are thus very interesting for further research. The lookup facility in this system is currently limited to simple key-value requests [7]. The RDF data storage concept with its unified triple model makes tuple storage systems interesting for a large number of applications, and research is progressing on the subject [4]. However, complex query processing has not been addressed at this point, due to the unstructured and undefined nature of the storage networks involved.

The research problem will therefore be the design, implementation and evaluation of a efficient query processor for a self-organized storage system. If these systems would contain such a component, it would enable them to replace systems using conventional methods while improving the scalability of the storage system. Significant adaptations of the methods previously used to achieve distributed query processing are expected to be required in the scope of this work, but the overall established structure can be maintained. It is however not yet clear, if there are efficient solutions to this issue at all.

The remainder of this paper is structured as follows: Section 2 formulates the research questions and clarifies relevant assumptions, 3 shows the approach followed in order to identify a solution, which is presented in Section 4. Section 5 shows the steps to be taken in order to evaluate the solution. Finally, Section 6 concludes this paper.

2 Main questions of the thesis

Given a self-organized storage system with no global data structures as they are present in other self-organizing tuple storage systems [3], the main challenge of this work is to determine whether it is possible to efficiently evaluate complex data queries for those systems. An example for self-organizing systems without global data structures are swarm-based approaches [2]. The ongoing implementation efforts for storage systems using these approaches are currently only considering simple lookup techniques [4, 8], thus the planned work will yield considerable progress for this area of research.

The combination of a swarm-based self-organizing tuple storage system with a efficient query processing system could provide a big improvement for tuple storage systems such as RDF stores. With its conceptually unlimited scalability, the combination is able to handle the enormous amount of data generated

by Future Internet applications and users while still being able to retrieve any information using complex query languages. Compared to structured Peer-to-Peer approaches, swarm-based systems are better able to cope with ever-changing network structure and load shifts.

A number of prerequisites are assumed to be fulfilled, in order to focus entirely on query processing. It is assumed, that there exists a system, which can be run on a large number of connected computer systems (nodes). This system will form an overlay network consisting of a neighborhood of known nodes for each node. The entirety of nodes is able to store and locate tuples of arbitrary length. Client computer systems can connect to any node using a specialized client API. Clients can write tuples into the storage network by specifying them explicitly. Clients can read tuples from the storage network by specifying fixed parts of tuples (templates). All tuples matching the fixed parts given in the template are then returned. A single node does not recognize more than its immediate neighbor nodes. There is no method to retrieve all tuples stored in the network. While retrieving data for a specific key, nodes can give an approximation, which neighbor node may store or lead to matching tuples. Issues regarding the handling of single key lookup operations or data storage are not within the scope of the planned work.

3 General approach

In accordance with previous work in the area of distributed query processing, the structure defined in that area is re-used in order to make the different approaches comparable. The following components involved in query processing [6] have been determined to be affected:

1. **Query Optimizer and Plan Refinement** Using a cost model considering either calculated values from static methods, general query heuristics, or statistical accumulation of knowledge, the query optimizer selects a execution plan for the query. In a system without global knowledge, it may not be possible to converge on the optimal plan immediately. Instead, approaches like mutable query plans may be employed, where the plan is adapted and refined during actual execution [1, 9].
2. **Catalog** The catalog maintains a collection of metadata describing the distribution and organization of data, their indicies, cardinalities, and further information all intended to assist the other components in performing their tasks. For a distributed system, the catalog may also be distributed. For a distributed relational database, this catalog defines all relations stored in the system, for an RDF store with its unified data model this is not required. In

our case, this information is not available to the single nodes, hence only heuristical and statistical methods are applicable to support the other components.

The listed components have to be adapted fundamentally in order to function in a self-organized environment. The remaining components are expected to remain largely untouched. The new query processing system will then be compared with the current key-only lookup technique using a performance analysis. A significant statistical improvement over naive approaches is expected for complex queries.

4 Proposed solution

In order to build a complex query processing system on top of a self-organized storage system, the base functionality of that system, in particular its ability to retrieve data using a single lookup key, has to be verified first. For any tuple stored in the system, the success rate for read operations requesting that tuple has to be shown to be high enough to support reliable operations. In a second step, existing query processing paradigms will be researched and adapted for our self-organizing system. Considerable amount of theoretical work will precede the implementation of one or various possible solutions on top of our existing self-organizing storage system. The last step will be a comparison between the new solution and the naive approach as well as existing solutions, where applicable.

5 Evaluation

As we have shown, static methods for query processing may not be applicable in our case. Hence, in order to evaluate our query processor, the emerging prototype will be integrated into a self-organized storage system and tested against a naive implementation. To make results comparable, an traditional distributed query processing engine will also be tested against our approach while paying attention to the design goals for self-organized systems. A test data set as close as possible to a real-world setup will be chosen and stored in the storage network. A set of complex queries involving this data set will also be chosen and executed. The average time to complete each query in multiple runs from different nodes will be measured, and the average results compared.

6 Future work

The work described in this paper will continue with an exhaustive screening of research regarding query processing in distributed environments. Using this

information, the road map to identify a set of possible solutions to the issue of efficient query evaluation in an unstructured distributed storage systems will be set up. After first phase of theoretical calculations and simulations on the set of possible solutions, the most promising solutions will be selected for implementation into a existing self-organized distributed storage system. Benchmarking runs and comparison with naive approaches such as flooding and random walk will show the environment where the selected solutions yield an improvement. It is hoped, that one of the selected solutions will cover a significant number of environments. This solution will then be refined and evaluated furthermore.

Acknowledgments This work has been supported by the "DigiPolis" project funded by the German Federal Ministry of Education and Research (BMBF), support code 03WKP07B.

References

1. Philip A. Bernstein, Nathan Goodman, Eugene Wong, Christopher L. Reeve, and James B. Rothnie, Jr. Query processing in a system for distributed databases (sdd-1). *ACM Trans. Database Syst.*, 6(4):602–625, 1981.
2. Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Santa Fe Institute Studies in the Sciences of Complexity Series. Oxford Press, July 1999.
3. Min Cai. RDFPeers: A scalable distributed RDF repository based on A structured peer-to-peer network. Technical report, University of Southern California, Computer Science Department, April 02 2008.
4. Marko Harasic, Anne Augustin, Philipp Obermeier, and Robert Tolksdorf. RDFSwarms - selforganized distributed RDF triple store. In *Proceedings of the 2010 ACM Symposium on applied computing, ACM SAC 2010*, 2010.
5. George Kokkinidis, Lefteris Sidirourgos, and Vassilis Christophides. Query processing in RDF/S-based P2P database systems, 2008.
6. Donald Kossmann. The state of the art in distributed query processing. *ACM Comput. Surv*, 32(4):422–469, 2000.
7. Ronaldo Menezes and Robert Tolksdorf. A new approach to scalable linda-systems based on swarms. In *Proceedings of ACM SAC 2003*, pages 375–379, 2003.
8. Hannes Mühleisen, Kia Teymourian, and Robert Tolksdorf. A swarm-based semantic storage service. Poster Session at the 7th Extended Semantic Web Conference, ESWC10, 2010.
9. Vassilis Papadimos, David Maier, and Kristin Tufte. Distributed query processing and catalogs for peer-to-peer systems. In *CIDR*, 2003.
10. Bastian Quilitz and Ulf Leser. Querying distributed RDF data sources with SPARQL. In *ESWC 2008, Proceedings*, volume 5021 of *Lecture Notes in Computer Science*, pages 524–538. Springer, 2008.