Web-scale Range Queries in a Self-Organized Storage Service

Hannes Mühleisen, Tilman Walther and Robert Tolksdorf
Department of Computer Science – Networked Information Systems Group
Freie Universität Berlin
Königin-Luise-Str. 24/26, D-14195 Berlin, Germany
muehleis@inf.fu-berlin.de, tilman.walther@fu-berlin.de, tolk@ag-nbi.de, http://www.ag-nbi.de

***Abstract*—Distributed solutions for the storage and retrieval of large amounts of data are necessary to handle the growing amounts of knowledge expected from future applications. The support for range queries provides much-needed expressivity, for example for queries on data annotated with location and time. We present a novel and scalable way for range query evaluation in our distributed storage system based on behaviour found in ants. Routing decisions in this system are taken on the basis of virtual pheromone paths leading from one node to another node, distinct for different data items. Range queries for single ranges can be evaluated by checking the pheromone paths for the entire range and forking the operation to several nodes if necessary. An aggregation method ensures the efficiency of the routing operations. We have evaluated our approach using a synthetic data set stored on 100 nodes, and executed various range queries while recording the amount of nodes involved along with the size of the result set.**

***Keywords*-Distributed systems, Swarm Intelligence, Query Evaluation, Range Queries**

I. Introduction

The growing amounts of knowledge easily exceeds the limits of single affordable computer systems. Distributed solutions for the storage and retrieval of large amounts of data are thus necessary. However, applications can rightfully expect an expressivity comparable to what can be achieved on single systems for retrieval operations. A popular example for this expressivity are queries on data annotated with location and time information. One method to support simple reasoning on this type of data is the evaluation of range queries on these annotations, since both location and time information can be translated to a continuous numeric scale. This would for example allow to query all data items describing events between two dates or created within a specific area. While this evaluation can be easily performed on single stand-alone storage systems, accomplishing the same task on a distributed storage system is challenging. As part of our prior research, we have developed a distributed storage system which uses a tuple-based data model and a novel storage routing algorithm based on behaviour found in some ant species. The main goals of this algorithm are scalability and adaptability; changes in network infrastructure and shifts in system loads can be tolerated efficiently through adaption. In this paper, we discuss our research question as to how range queries can be evaluated efficiently in a distributed self-organized storage system.

On the path of answering our question, Section II presents our previous and related work on the matter. Section III introduces the basic operations of our distributed self-organized storage service "S4" and describes our concept for range query evaluation within this system. Based on our implementation, Section IV contains a short evaluation of the described concept. Finally, Section V concludes this paper, discusses its results, and gives an overview on future work.

II. Related and Previous Work

The different approaches for the distribution and management of data over several nodes lead to different concepts for the retrieval of a subset of the stored data which is defined by a range. In recent years, much research was focused on structured peer-to-peer networks such as CAN [1], Chord [2] or Tapestry [3]. These systems work as a distributed hash table (DHT) that maps values to a node which is responsible for a certain set of values. Together with the use of an order-preserving hash function, DHT based systems can be combined with overlay structures like tries or skip lists in order to allow range queries over the stored data. Examples for such approaches are proposed by Tanin et al. [4], Aspnes et al. [5] or Yalagandula [6]. Specialized stores like RDFPeers [7] or GridVine [8] employ comparable techniques for semantic data sets encoded using the Resource Description Format (RDF).

While DHT based peer-to-peer systems can be extended in order to support range queries, S4 offers a different approach which provides even higher scalability and adaptability. This is being realized through swarm algorithms, which have been identified as a powerful family of methods by Bonabeau et al. [9]. Basic concepts are the absence of global state and the use of simple, independently acting agents. In the case of S4, the probabilistic routing decisions of these agents lead to a highly adaptive storage network at the cost of a reduced routing efficiency.

Swarm intelligence was applied to build a distributed tuple space by Menezes and Tolksdorf [10] which led to the development of a distributed storage service based on swarm-algorithms, S4. A comprehensive overview of the concepts and methods employed in the system can be found in our previous work [11].

III. Range Query Routing in S4

As mentioned in the introduction, we have developed a distributed tuple storage system based on swarm intelligence, in particular the foraging method used in several ant species. Here, ants embark on a random walk from their nests to search food. As soon as they encounter food, a portion is picked up and carried back to the ant nest. On the way back, ants spread a pheromone, which can in turn be sensed by other ants. If an ant on its random walk in search for food detects a pheromone trail, it follows the trail. This represents a positive enforcement mechanism, and a small margin of error along with natural pheromone evaporation ensures the ability of the system to adapt to changing circumstances. The main trade-off of this approach are any guarantees towards completeness and execution time. However, our previous research showed this having no serious impact on its performance, with the majority of queries being resolved in near-optimal time delivering a full result set [11].

For distributed storage and retrieval, a network of storage computers is regarded as the landscape on which ants move, and the stored data items are considered to be the ant's food. Distinct pheromones for each stored data item create a network of pheromone paths, which can be used to efficiently locate data items. Each node stores a part of these paths that include himself, thereby creating a fully distributed data structure. We have implemented this concept for RDF triples in our *Self-Organized Semantic Storage Service* (S4) [11]. Based on an extensible tuple data model, simple storage and retrieval operations on an arbitrary number of nodes are supported. By using the pheromone paths, retrieval operations can be started for any stored data item from any node that is part of the storage network. Distinct from the S4-specific pheromone and networking layer, local storage is independent and may use any storage system at hand, for example a relational DBMS with range query support.

In order to extend this system to efficiently support temporal and spatial queries on the stored data, a distributed storage service has to support both multiple routing indices as well as range queries on these indices. For example, should data items for a particular period of time be retrieved, we can query the routing index for temporal values using the date range. A routing index is a specialized data structure built to route requests with constraints for a particular dimension of the stored data. We have discussed the handling of multiple routing indices within the S4 system in [12].

From a user's perspective, each index has to be configured using the corresponding tuple entry and an appropriate mapping function m, which maps the entries for the tuple index to a numerical value. With these numeric values pheromones are compressed using a lossy aggregation into so-called buckets, which are then used to support the routing decisions for the storage operations. These buckets can contain potentially large amounts of pheromone values grouped by the numeric values they have been mapped to. Within the context of this paper, we regard range queries on tuples as a set of three values: The index to be queried i, and the minimum and maximum value of the range r_{min} and r_{max}. Retrieval operations requesting multiple ranges can be easily split up into multiple retrieval operations containing only one range. For example, a retrieval operation on the temporal index for data items between December 2010 and March 2011 could be given as follows:

$$read(i = temp, r_{min} = m(12/2010), r_{max} = m(03/2011))$$

One of the main concepts of S4 is to store date items mapped to similar pheromone values in the same vicinity within the storage network. A special movement heuristic ensures the eventual consistency of this property [13]. Here, smaller sets of data items using the same pheromone values as larger sets are dissolved and added to the larger sets. The evaluation of a range query is now simplified, as the evaluation of the query is now mainly a question of identifying the part of the storage network, where the relevant data items are stored and searching this part more thoroughly. To evaluate range queries, we can thus follow pheromone paths through the network, as long as the pheromone values for both r_{min} and r_{max} are contained in exactly one pheromone bucket leading to another host.

As soon as there are more than one bucket containing these values, or if the minimum and maximum values are contained in different buckets, the retrieval operation forks itself into several sub-operations. For each pheromone bucket leading to another host overlapping or containing the retrieval operations' value range, a "child" retrieval operation is forked. Since read operations in S4 are limited by a step count, the "child" retrieval operations are only able to take as many steps as the "parent" operation had left, ensuring eventual termination. If the retrieval operations encounters data items on a node matching the carried range, these get directly forwarded to the node the main operation has originated from, which is then able to assemble the final result and deliver it to the waiting client application.

Fig. 1 gives an example for this method. A range query for $[r_{min}, r_{max}]$ is started on node 1. The query is forwarded to node 2 and 3, since both limits are exclusively contained in the same pheromone buckets for connections to node 2 and 3, respectively. On node 3, however, this is no longer the case and the range query is split up into three child range queries $[r_{min}, x]$, $[x, y]$ and $[y, r_{max}]$ and forwarded to nodes 4, 5 and 6.

The main advantage of this approach is the usage of the already existing pheromone trails to locate the part of the storage network where data items matching the range specification are present. Assuming correct operation of the re-organization heuristic, retrieval operations containing range specifications are only split up as they start encountering matching data. This is expected to considerably reduce the

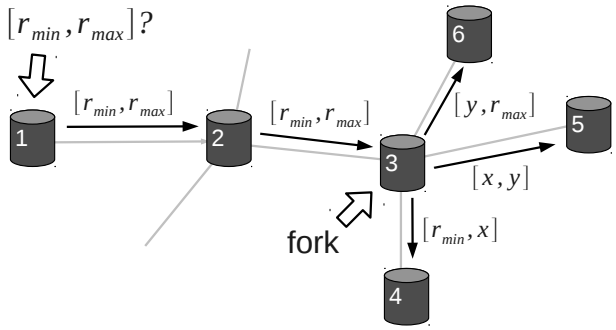

Figure 1. Forking example

amount of nodes to be queried in order to process a range query, which would allow our approach on evaluating those queries to scale to an arbitrary storage network size. The presented approach to range query evaluation could also be adapted to similar distributed storage systems, in particular systems using directional routing as well as key locality.

IV. Concept Evaluation

The main goal of our distributed storage service is to provide a very high level of scalability. Our method to support range queries was therefore designed under the same preconditions. Scalability for our distributed systems is mainly the ability of the system to handle requests using only a small subset of the nodes in the storage network. Consistent with our previous evaluation, the following evaluation is intended to describe the amount of nodes involved in creating an answer rather than the actual execution time since the latter is very dependent on the implementation. If the number of nodes participating in the generation of a result for a range query is limited to a very specific part of the network, our range query approach possesses the desired scalability.

We have implemented our concept into the S4 system and deployed it onto our test network containing 100 storage nodes. We have then written a synthetic test data set containing of 1 million data items to the network. These data items used evenly distributed numeric keys between 0 and 10000. Range queries matching an increasing number of data items stored on an increasing number of nodes were then performed multiple times. For each query run, the amount of nodes participating in the generation of a result and the amount of returned data items was measured. Range queries were repeated on every member node of the storage network, as our approach is fully distributed. For each test, range queries between $[0, 100]$ and $[0, 5000]$ were run on every node. These figures were then averaged for the test run.

Fig. 2 contains the results averaged over seven consecutive test runs. Two main results can be observed from this Figure: First, the amount of elements found increases consistently with the range size. This is to be expected, since the data items were distributed evenly over their numeric keys. This also shows the general feasibility of our approach for range queries in our distributed storage system. Second, the amount of nodes involved in the generation of the result is reduced as the range size is increased. This counter-intuitive behaviour can be explained easily, as larger ranges have a far higher possibility to find matching pheromone buckets on every node, thus finding the nodes storing matching data faster. We therefore conclude on the general feasibility and scalability of our approach on evaluating range queries on our self-organized distributed storage service.

V. Conclusion and Future Work

From the growing amounts of knowledge to be handled by applications, the need for distributed storage solutions became apparent. The support for range queries was introduced as a convenient way to support spatiotemporal queries. Range queries required the extension of retrieval operations within our swarm-based system to provide more expressivity than exact value matches. Within our context of a swarm-based distributed storage system aiming at scalability, the main focus of this paper was the question whether range queries can be evaluated efficiently.

Following a short introduction into the methods used to provide scalability in our S4 system, we have extended its support for single-key matches to range queries using only the system-inherent and already present data structures of the pheromone paths, which are formed on-demand by retrieval operations. We have evaluated our approach using a black-box evaluation methodology based on the amount of hops needed to retrieve a subset of a synthetic data set using range queries evaluated by our method. Our evaluation results showed the feasibility and scalability of our approach. To reiterate on our research question, we were indeed able to evaluate range queries on a self-organized distributed storage system efficiently.

A. Future Work

As future work, we want to extend our evaluation of our range query method to different storage network structures. So far, queries were run on the network created by our bootstrap algorithm, which exhibits many desirable features [11]. However, our approach is also to be evaluated on storage networks with more nodes, less connectivity or undesirable network characteristics such as a high average path length. Also, since this evaluation used a synthetic data set for conciseness, an extended evaluation is required to use a data set closer to reality, possibly including spatiotemporal data and corresponding queries. Another concern is the evaluation of the recall that can be achieved using our approach: The correct result set could be calculated beforehand on limited data sets, and the results from our approach can then be compared, yielding further insight in its practicability.

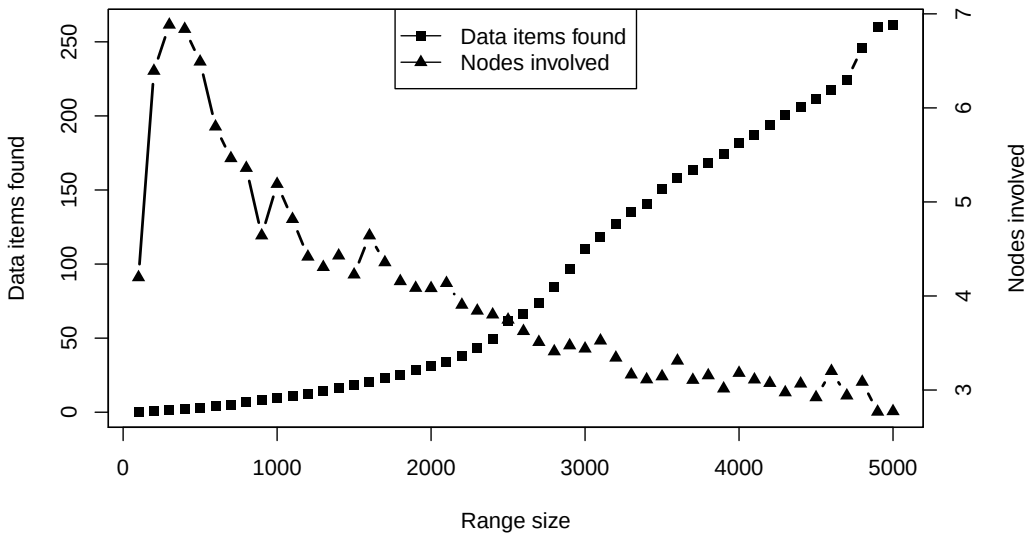

Figure 2. Range query evaluation results

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful comments and suggestions. Also, we would like to thank our student assistants Alexander Dümont, Sascha Gennrich, Dennie Kabul, and Kürşad Özarpat for their work on the S4 implementation. This research has been partially supported by the "DigiPolis" project funded by the German Federal Ministry of Education and Research (BMBF) under the grant number 03WKP07B.

REFERENCES

[1] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *IN PROC. ACM SIGCOMM 2001*, 2001, pp. 161–172.

[2] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for internet applications," in *ACM SIGCOMM*, 2001, pp. 149–160.

[3] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 41–53, 2004.

[4] E. Tanin, A. Harwood, and H. Samet, "Using a distributed quadtree index in peer-to-peer networks," *VLDB Journal*, vol. 16, pp. 165–178, 2007.

[5] J. Aspnes, J. Kirsch, and A. Krishnamurthy, "Load balancing and locality in range-queriable data," in *In Proceedings of the Symposium on Principles of Distributed Computing (PODC*, 2004, pp. 115–124.

[6] P. Yalagandula and J. C. Browne, "Solving range queries in a distributed system," Dept. of Computer Sciences, University of Texas at Austin, Tech. Rep. TR-04-18, 2004.

[7] M. Cai and M. Frank, "RDFPeers: a scalable distributed RDF repository based on a structured peer-to-peer network," in *WWW '04: Proceedings of the 13th international conference on World Wide Web*. New York, NY, USA: ACM, 2004, pp. 650–657.

[8] P. Cudré-Mauroux, S. Agarwal, and K. Aberer, "GridVine: An infrastructure for peer information management," *IEEE Internet Computing*, vol. 11, no. 5, pp. 36–44, 2007. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/MIC.2007.108

[9] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, ser. Santa Fe Institute Studies in the Sciences of Complexity Series. Oxford Press, July 1999.

[10] R. Menezes and R. Tolksdorf, "A new approach to scalable linda-systems based on swarms," in *Proceedings of ACM SAC 2003*, 2003, pp. 375–379.

[11] H. Mühleisen, A. Augustin, T. Walther, M. Harasic, K. Teymourian, and R. Tolksdorf, "A self-organized semantic storage service," in *Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services (iiWAS2010), preprint available at http://digipolis.ag-nbi.de/preprint/iiwas2010-s4-preprint.pdf*, 2010.

[12] H. Mühleisen, T. Walther, and R. Tolksdorf, "Multi-level indexing in a distributed self-organized storage system," in *Submitted to the 2011 IEEE Congress on Evolutionary Computation (CEC2011), preprint available at http://digipolis.ag-nbi.de/preprint/cec2011-s4.pdf*, 2011.

[13] ——, "Data location optimization for a self-organized distributed storage system," in *Submitted to the Ninth German Conference on Multi-Agent System Technologies, preprint available at http://digipolis.ag-nbi.de/preprint/mates2011-s4-preprint.pdf*, 2011.