Estimating the Number of Answers with Guarantees for Structured Queries in P2P Databases*

Marcel Karnstedt
Kai-Uwe Sattler
Michael Haß
TU Ilmenau, Germany
{*first.last*}@tu-ilmenau.de

Manfred Hauswirth
Brahmananda Sapkota
DERI, NUI Galway, Ireland
{*first.last*}@deri.org

Roman Schmidt
EPFL, Switzerland
{*first.last*}@epfl.ch

ABSTRACT

Structured P2P overlays supporting standard database functionalities are a popular choice for building large-scale distributed data management systems. In such systems, estimating the number of answers for structured queries can help approximating query completeness, but is especially challenging. In this paper, we propose to use routing graphs in order to achieve this. We introduce the general approach and briefly discuss further aspects like overhead and guarantees.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed Databases; H.3.3 [**Information Search and Retrieval**]: Search process

General Terms

Algorithms, Management, Performance

1. INTRODUCTION

A key challenge for large-scale distributed data management based on distributed hash tables (DHT) is that the advantages of overlays – no global knowledge required, no single-point of failure, superior resilience against partial failure – incur new problems when it comes to supporting advanced queries. Due to the lack of global knowledge, unpredictable changes in the peer population (churn), lack of global control, etc., query processing techniques from classical database systems can only be applied to a limited extent: The classical database view is based on a closed world assumption and thus a query will always return all matching answers. In contrast to this, P2P data management is inherently open world: While processing a query, peers can fail, leave or join the network, or simply send no or a delayed answer. Though this can be mitigated by replication and delay-tolerant query techniques, there is no guarantee that all answers which potentially exist can be returned.

*The work presented in this paper was (partly) carried out in the frameworks of the EU project TripCom (FP6-IST-4-027324-STP), the Líon project supported by the Science Foundation Ireland under Grant No. SFI/02/CE1/I131 and the EPFL Center for Global Computing and supported by the Swiss National Funding Agency OFES as part of the European project NEPOMUK No FP6-027705.

CIKM'08, October 26–30, 2008, Napa Valley, California, USA.
ACM 978-1-59593-991-3/08/10.

Estimating the number of answers expected for a query can help to estimate the completeness of query results. This is not in order to guarantee completeness and availability. Rather, it helps to assess the quality of results received by "in situ querying", where no constant and repeatable results can be expected. This is in contrast to [4], where the authors propose to wait for results getting available (again). They estimate query completeness in order to argue on the delays resulting from this. We dissociate from the classical understanding of completeness, which should rather be referred as availability in the introduced context. This is a rather static aspect and due to the DHT.

In this paper, we introduce the notion of routing graphs. The routing graphs are used to trace the routes a query and its sub-queries take through the network. The main contribution of our work is the development of lightweight techniques for providing reliable information about the result completeness of complex database-like queries in a dynamic and unreliable P2P environment. From a conceptual view, this is on peer level, rather than an estimation on data level with the need for precomputed data summaries or detailed information about neighbor peers (like in [4]). The peer-level approach is much cheaper and better suited for large-scale and dynamic networks than the data-level approach. The underlying assumption is that data is balanced among the peers and that a linear correlation between the number of failed peers and the resulting miss of data from the expected answers exists. This assumption holds in DHTs providing load balancing features such as P-Grid [1]. Furthermore, we aim at an incremental and online refinement of the estimation. We give a rather high level description of our approach and briefly refer to further aspects like the resulting overhead and possible guarantees. We present more details on the methodology and further issues and evaluate the introduced algorithms in [2].

2. GENERAL APPROACH

We implemented our approach in the UniStore [3] data management system. Query processing is, like in traditional database systems, based on query plans composed of plan operators. The processing of an initial query plan may result in multiple sub-query plans that travel independently through the network.

The processing of a single query plan q is started at the query initiating peer. Processing this query results in parallel sub-plans $q_i, i = 1, \ldots, \text{R}$, where R is the number of all plans generated. Each single plan is processed separately and results in a reply sent to the initiating peer, even if the

final query result in one of these plans should be empty. To estimate the query completeness, we have to estimate R̃.

The path of a query and all of its sub-plans through the network can be illustrated as a directed tree, with the leafs corresponding to the set of peers sending final replies. Thus, the number of leafs in that tree is R̃. We call such a tree a *query routing graph* $RG(V, E)$. A vertex in the graph, called *routing point*, represents a peer responsible for processing a part of a q_i, i.e., the peer is involved in the processing of at least one operator in q. Edges are called *routing connections*. For each sub-plan that is routed from one routing point to another we have one such edge in RG. Thus, each routing level corresponds to the processing of a single plan operator. A peer may represent multiple vertices of a routing graph because it may be contacted several times for processing different parts of q.

As a consequence, the query routing graph is a topology overlaid on the topology of the overlay (ring, tree, etc.). The graph also differs from its query plan q as the processing of one operator may span multiple routing points and multiple operators may be processed at one routing point. The latter is true when multiple operators can be processed locally on one peer without routing. Figure 1 gives an illustrating example.

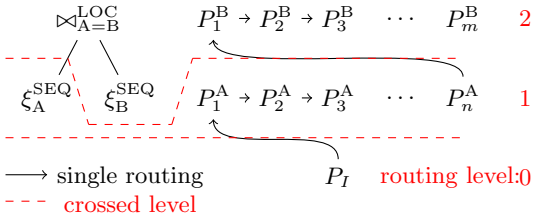

Figure 1: Sequential processing

ξ_{A} extracts all triples corresponding to attribute `A`. Assuming post-order processing, after extracting all data for attribute `B` the final join $\bowtie_{\text{A=B}}$ can be processed on P^B_m. The right side of the figure shows the routing graph corresponding to a strict sequential processing of the query plan. The query is initiated at P_I, which sends the plan to the first peer responsible for `A`. Each peer responsible for a part of `A` extracts local data, includes it into the query plan, and forwards the query accordingly. After all n peers in this sequence were contacted, the nth peer crosses the next routing level by routing the plan to the first peer responsible for processing ξ_{B}. The join can be processed finally at P^{B}_m, which sends the final reply to the initiator. Intuitively, estimating the number of replies for strict sequential processing is trivial, as R̃ = 1 always holds. Modifications like intermediate replies can be integrated into completeness estimation similar to the parallel processing strategies discussed in the following. Figure 2 shows the routing graph for the same query plan, but with intra-operator parallelism.

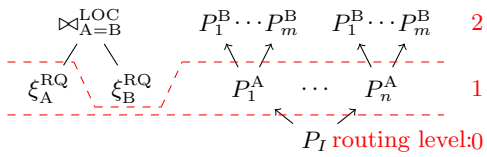

Figure 2: Intra-operator parallel processing

First, all peers responsible for `A` are contacted in parallel. Each of the peers $P^{\text{A}}_i, i = 1, \ldots, n$ independently contact all peers responsible for attribute `B` in order to finalize processing. Due to the parallelism, the number of generated plans and thus replies is multiplied while processing the separate levels of the query plan. For strict intra-operator parallel processing the number of replies is $\tilde{R} = \prod_{l=0}^{L-1} d_l$, where L is the maximal routing level, and d_l is the fanout at each level l. Note that this only holds if the number of generated plans is equal for each forwarding peer, which means that it is independent from the actual determined (intermediate) data. This is true, for instance, when using intra-operator parallelism based on range queries as shown in Figure 2. In the more general case, the number of leafs can be described by a recursive formula:

$$p_0 = 1, \quad p_l = \sum_{k=1}^{p_{l-1}} d_{(l-1),k} \quad (l = 1, \ldots, L), \quad \tilde{R} = p_L$$

where p_l is the number of vertices at level l, and $d_{l,k}$ is the fanout of the kth vertex at level l.

Algorithm 1 Basic algorithm: new-reply()

```
1: update-overlay();
2: update-RG();
3: r = r + 1;
4: R = 1;
5: for all routings ρ ∈ RG → get-levels() do
6:     R = R + ρ → estimate(R);
7: end for
8: return r/R;
```

Based on the notion of routing graphs, the basic idea of estimating query completeness works as shown in Algorithm 1. This procedure is called every time a new reply arrives at the initiator. After updating necessary information in the estimated overlay structure and the routing graph RG (one for each initiated query), the fraction of received results r and estimated final replies R is returned. Thus, a currently estimated completeness (on peer level) is determined in an online fashion. R is estimated iterating over all levels of RG. We have to identify routing methods and define equations for estimating the number of expected sub-query plans for each. We use a query trace contained in each reply to collect the necessary information. In its default implementation, this involves no overhead but some marginal bandwidth consumption. Further, we are able to give probabilistic guarantees for the estimations. Details on this are presented in [2].

3. REFERENCES

[1] K. Aberer, M. Hauswirth, M. Punceva, and R. Schmidt. Improving Data Access in P2P Systems. *IEEE Internet Computing*, 6(1), 2002.

[2] M. Karnstedt, K. Sattler, M. Haß, M. Hauswirth, B. Sapkota, and R. Schmidt. Approximating Query Completeness by Predicting the Number of Answers in DHT-based Web Applications. In *Int. Workshop on Web Information and Data Management, WIDM icw CIKM'08*, 2008. to appear.

[3] M. Karnstedt, K. Sattler, M. Richtarsky, J. Müller, M. Hauswirth, R. Schmidt, and R. John. UniStore: Querying a DHT-based Universal Storage. In *ICDE'07 Demonstrations Program*, pages 1503–1504, 2007.

[4] D. Narayanan, A. Donnelly, R. Mortier, and A. Rowstron. Delay aware querying with seaweed. In *VLDB'06*, pages 727–738, 2006.