Processing Skyline Queries in P2P Systems

Katja Hose

Department of Computer Science and Automation, TU Ilmenau
P.O. Box 100565, D-98684 Ilmenau, Germany
katja.hose@tu-ilmenau.de

Abstract

Efficient query processing in P2P systems poses a variety of challenges mainly resulting from the strict decentralization and limited knowledge. Particularly with regard to queries involving ranking, top-N or skylines, existing approaches for centralized systems cannot be applied easily to P2P environments. In this paper, we focus on the problem of efficiently processing skyline queries in large-scale P2P systems, where it is nearly impossible to guarantee complete and exact query answers without exhaustive search, i.e., flooding the network. Thus, applying approximate query answering techniques, that are also typical for processing top-N queries in centralized database environments, seems to be the natural choice. We address this problem by presenting an approach that allows for reducing the number of queried peers as well as for giving probabilistic guarantees for the correctness of the answer.

1 Introduction

Schema-based Peer-to-Peer (P2P) systems, also called Peer Data Management Systems (PDMS), are a natural extension of federated database systems. In addition to the characteristics evolving from the P2P paradigm (namely autonomous peers with equal rights and opportunities, self-organization as well as avoiding global knowledge), each peer in a PDMS provides its own data with its own schema. All peers can answer and process queries and are linked to a small number of neighbors via mappings representing schema correspondences.

The expected advantages of such a structure like robustness, scalability and self-organization come not for free: In a large-scale, highly dynamic P2P system it is nearly impossible to guarantee a complete and exact query answer. The reasons for this are among others possibly incomplete or incorrect mappings, data heterogeneities, incomplete information about data placement and distribution and the impracticality of an exhaustive flooding. Therefore, best effort query techniques such as similarity selection and join, nearest neighbor search, top-N operators and skyline queries are most appropriate. By ”best effort” we mean, that we do not aim for exact results or guarantees but instead try to find the best possible solution w.r.t. the available local knowledge. However, even if we relax exactness or completeness requirements we still need estimations or predictions about the error rate. This means for example giving a probabilistic guarantee that none of those peers that have not been asked could contribute to the result.

For illustration purposes we use an example of a virtual astronomical observation scenario, where each participating observatory represents a peer offering data about sky observations. A simplified query in this scenario might ask for a set of astronomical objects that are both, as bright as possible and situated as close as possible to a specified point in space that is defined by a set of coordinates. In most situations it is not obvious, whether the user would prefer (i) an object that is situated very close to the given coordinates but that is not as bright as others, or (ii) a rather bright object that is situated farther away than others. As such, it is important to present all interesting answers that might fulfill the user's needs, so that he or she can choose the most promising one. This set of interesting answers is called the skyline. Note that in contrast to distributed information retrieval we have to deal with search in structured data instead of distributed keyword search in text data.

Formally speaking, given a set of data items the skyline comprises all those data items that are not dominated by any other item. One data item dominates another if it is *as good as or better* in all dimensions and *better* in at least one dimension. In the example stated above this means, *object1* is dominated by *object2* if *object2* is brighter and situated at least as close to the given coordinates as *object1*.

We assume XML to be the underlying data format and XQuery to be the common query language. Based on a possible extension of XQuery the example skyline query introduced above could be formulated as follows:

```
for $s in fn:doc("sky.xml")//objects
skyline of MIN distance($s/rascension,
                        $s/declination, 160, 20)
            MAX $s/brightness
return ...
```

The remainder of this paper is structured as follows: at first, in section 2 we have a closer look at the state-of-the-art algorithms revealing why they cannot be applied to P2P systems. After having presented a new probabilistic approach for processing skyline queries in section 3, we finally give

a short summary of our current state of research and an outlook on our future work in section 4.

2 Why Do We Need New Algorithms for Computing Skylines in P2P systems?

As stated in [3], which was the first paper to introduce the skyline operator into the field of database research, computing the skyline was known as the maximum vector problem [9, 12] before. Along with a corresponding SQL extension, which allowed MIN, MAX, and DIFF as annotations, [3] presented some basic algorithms for computing the skyline in centralized databases. Before discussing their applicability to P2P systems, let us summarize the main problems that we have to deal with:

- Working with massively distributed environments with large numbers of peers that are autonomous and can leave or join the network at any time.
- There is no central instance that provides information about data distribution and localization. A peer only has limited knowledge about neighboring peers in a horizon defined by a finite hop count.
- Qll peers are equal: Each of them has the ability to initiate and process queries.
- Additionally for PDMS: There is no global schema. Thus, when forwarding a query it has to be translated into the receiving peer's local schema.

Application of Existing Algorithms

BNL (Block Nested Loops) and D&C (Divide & Conquer) were two of the first algorithms proposed for application in database environments [3]. They were designed for reducing main memory consumption and CPU cost in centralized RDBMS, where one peer processes the entire skyline with all required data being available locally. Obviously, comparing each data item to all the others is only feasible when collecting all data at one central unit, e.g., at the initiating peer. Then, just like in centralized databases, the BNL or D&C algorithm can be applied on the collected data. Apparently, this naive approach is not efficient because collecting data locally requires high network bandwidth and is likely to exceed local memory.

Applying the D&C paradigm on P2P systems results in a strategy where the network is flooded and each peer processes the query locally. The results are merged on their way to the initiator. Although the algorithm only needs one round-trip it has a major drawback that consists in the need for flooding the network, which is not acceptable with regard to large-scale P2P systems.

Index based algorithms as proposed in [13] (Bitmap and B-tree) require specialized index structures for processing skyline queries. These pre-computed indexes are designed for indexing data that is available locally and describe every queried dimension of each single data item. Due to the characteristics of P2P systems, dynamic behavior in particular, such index structures are not feasible. The reasons for this are among others the need for global knowledge and high maintenance costs.

The main advantage of algorithms like NN-search (Nearest Neighbor) [8] and BBS (Branch and Bound Skyline) [11] is that they can give an overview describing the final result after short time of processing. In principle, the basic idea of partitioning the data space and looking recursively for a nearest neighbor in each of these partitions would also work in P2P systems. However, applying this basic approach without modifications results in one round-trip for each nearest neighbor lookup. Developing efficient techniques for reducing a round-trip's costs, e.g., based on query refinement techniques, as well as reducing the number of round-trips (especially considering duplicate elimination) might be worth investigating in future work.

The first algorithm that considered processing skyline queries in distributed environments was proposed in [2] and enhanced in [1]. Principally, both algorithms are based on the TA algorithm by Fagin [10]. The algorithms are optimized for use in Web Information Systems and based on the concept that each Web source provides a globally ordered score list, that can be accessed by sorted and random access. This concept is hardly applicable to P2P systems because instead of dealing with only a few peers, that the central processing unit has direct access to, we have to deal with many peers that are situated in a distance of several hops and that might only be accessible indirectly using other peers for message forwarding. Moreover, providing a list in globally sorted order requires global knowledge that is not available in P2P systems.

None of the existing algorithms provides an efficient solution for processing skyline queries in P2P systems. In summary, the main problems consist in:

- the necessity of a central unit that processes the query with all data being available locally.
- the dependency on specialized index structures that represent global knowledge.
- the need for several round-trips that result in rather large execution costs in P2P systems.

Keeping these problems of existing approaches in mind, our research focuses on developing an approximative strategy for use in P2P systems in general and PDMS in particular. This strategy only needs one round-trip and gives probabilistic guarantees for the result's correctness and reduces costs based on decentralized routing optimization and index structures with limited horizons. We present this strategy more detailedly in the following section.

3 New Approaches for Efficiently Processing Skyline Queries in P2P Systems

After having presented why existing algorithms cannot (or at least cannot easily) be applied to P2P systems, this section comprises new approaches for processing skyline queries. These approaches are based on two main demands that we make on an efficient skyline algorithm for application in P2P systems: (i) no reliance on global knowledge and (ii) efficient processing of skyline queries in terms of execution cost, i.e., ask only some peers, and ask those only once.

3.1 Strategies

Our research on computing skylines in P2P systems led us to the following classes of processing strategies, each of them describes one main principle and thus one general class of more detailed processing techniques. The first three of them can be directly derived from existing approaches and have been introduced in section 2.

1. **Naive**: Collecting all data at the initiator and computing skylines locally using any algorithm for centralized processing.
2. **Nearest Neighbor**: Computing a nearest neighbor (NN) by flooding the network (first round-trip), partitioning the data space according to NN, recursively looking for NN in each partition (several round-trips).
3. **D&C**: Flooding the network, each peer processes the skyline query locally, merging skylines when sending answers to the initiator.
4. **Probabilistic**: Using routing filters for determining which peers to forward the query to in a probabilistic manner, computing local skyline points, checking answer data for dominance.

A general approach for reducing costs is not to ask all those peers that cannot contribute to the final result. This requires an index structure that describes the data of neighboring peers. We will present such an index structure in subsection 3.2. Though index structures like DHTs could be used for processing skyline queries, an algorithm based on such structures can be applied only in a few P2P systems. Since we are aiming at a more general approach, we will not pursue this aspect any further, but concentrate on a local index structure that is designed for dynamic environments (having a limited horizon and being maintained using query feedback).

In order to decide which neighboring peers cannot contribute to the final result, we assume that each peer possesses indexes that are defined on the queried attributes (dimensions respectively). Note that this approach is also applicable to super peer architectures, where we can regard the backbone network (formed by the super peers) as P2P network such that each super peer and its clients are treated as one unit with regard to index creation and maintenance. According to such indexes queries are routed only to those peers that might provide relevant data. Considering the fact that indexes always represent an approximation of reality a peer might not exactly know whether its neighbor can actually contribute to the result or not. Forwarding the query to all such peers results in being on the safe side but in most cases ends up in flooding the network. Consequently, in addition to reducing message volume and the number of round-trips further cost reduction can be achieved by taking the risk of missing relevant data by not forwarding the query to questionable peers. This, of course, means a tradeoff between correctness and efficiency.

Our strategy is based on this consideration and quantifies the risk of having missed relevant data in a probabilistic manner. This results in statements like "the output meets the exact result with a probability of 96%", i.e., with a probability of 96% there exists no data point residing at any non-involved peer that dominates any point of the algorithm's result. In order to be able to make routing decisions that guarantee for example a user-specified input probability of 90%, we need indexes that allow for calculating such probabilistic guarantees. Based on the concept of routing filters [7] that we developed for indexing XML data on both index and schema level, the following subsection briefly presents how to enhance that concept for giving probabilistic guarantees.

3.2 Advanced Routing Filters

Routing filters mainly represent a combination of local index structures, XML synopsis, and histograms. Local index structures have been shown to be suitable for efficient routing in P2P environments [4]. Histograms are successfully used in a wide variety of optimization and estimation techniques. So we decided to combine them resulting in the concept of routing filters that allow for approximating the distribution of attribute values. At each peer one filter is maintained for each established connection to a neighbor. The filter describes all XML data (schema and instance level) that can be accessed by forwarding a query to the neighbor. Routing filters are built and maintained using a query feedback approach. If they are not limited to any finite horizon (e.g., by applying the concept of hop counts [4]) and if we assume that no peers leave the network, they will converge to global knowledge as time passes by.

Since the problem of indexing data on schema level is not the focus of this paper, we will only consider how to index data on instance level and how to give probabilistic guarantees. Histograms approximate data distributions by partitioning a sort parameter into intervals (buckets) and approximating the source parameter in each of them. In case of routing filters this means that for each bucket B_i (defined by lower and upper boundaries l_i and u_i) we keep the average frequency h_i of all its attribute values. Being inspired by *V-Optimal histograms* [5] that try to minimize the variance of source parameter values, i.e., frequencies, within each bucket, we additionally store a characteristical value (maximum error e_i) per bucket that describes the error of the assumed frequency distribution. h_i and e_i are calculated according to the following equations:

$$h_i := \frac{\sum_{k=l_i}^{u_i} F(k)}{u_i - l_i + 1} \quad , \quad e_i := \max_{k=l_i..u_i} \{|F(k) - h_i|\}$$

where given n data values, represented by function $R : \{1..N\} \rightarrow \mathbb{N}$, we can determine the frequency $F(k)$ for any sort parameter value k with $F(k) = |\{e|R(e) = k\}|$. For simplicity we only consider discrete attribute values.

Figure 1 shows a part of a routing filter that only contains one-dimensional histograms. The figure shows simple histograms defined on attribute 'x' for two neighbors of the filter owning peer. The solid line represents the filters' assumed frequency (h_i) and the dashed lines the maximum and minimum frequencies ($h_i \pm e_i$) that the neighbors' data might actually meet.

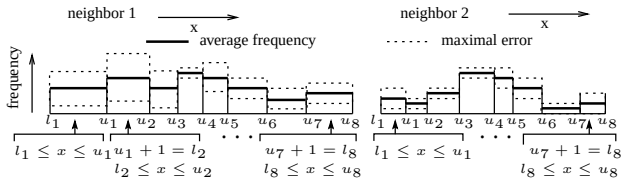

Figure 1: Histograms on 'x' for two neighbors

When applying our strategy, each peer according to its local routing filters has to decide independently from other peers which neighbors to forward the query to. Thus, facing the question whether a neighbor possesses relevant data in a specified range of interest, we have to adopt one independent random variable for each relevant data point and thus, a probability distribution for each of these random variables and calculate the convolution. As described in [14] this calculation is expensive in general, but rather efficient in case of normal distributions. For this purpose we keep the following information per bucket: average frequency h_i, maximum absolute error e_i, and standard deviation σ_i.

3.3 Probabilistic Approach

Considering the two demands mentioned in the beginning of this section (efficiency and no dependency on global knowledge), we developed a probabilistic strategy that can be characterized as follows:

- only one round-trip for answering a query: reduces execution costs and the impact of the network's dynamic behavior
- working incrementally: first results can be output at an early stage, the impact of the network's dynamic behavior is reduced (as shown in [6])
- reducing costs: minimizing the number of queried peers, pruning query paths that cannot contribute or at least are not likely to contribute to the final result (using routing filters)
- giving probabilistic guarantees for describing the result's correctness (using routing filters)

Input: QD: query definition, P_{in}: probability, $pred$ predecessor
RF: routing filters, S_{pred}: predecessor's skyline points
1 S_L = process-local-skyline(QD);
2 S_A, S_G = check-for-dominance(QD, S_L, S_{pred});
3 RN, P_L = calc-relevant-neighbors(QD, RF, S_G, P_{in});
4 send-answer($pred, S_A, P_L$);
5 P_C = divide-probability($P_L, |RN|$);
6 forward-query(RN, QD, P_C, S_G);

Figure 2: Basic algorithm for the probabilistic approach

Figure 2 depicts the algorithm's basic workflow. A query does not only contain the query definition QD, i.e., which attributes to minimize and maximize, but also the user's input probability P_{in}, i.e., (1 - *the risk of missing skyline points* that the user is willing to take). Furthermore, a query contains those skyline points S_{pred} that have already been determined by the peer that the query has been received from (called *predecessor* in the following).

At first, each peer that receives a query processes its local skyline S_L based on its local data (line 1). Afterwards, all points in S_L are checked for dominance by S_{pred} (line 2) resulting in: $S_A = \{p \in S_L | \neg dominated(p, S_{pred})\}$, $S_G = \{p \in S_L \cup S_{pred} | \neg dominated(p, S_L \cup S_{pred})\}$. $dominated(x, y)$ is true when point x is dominated by any point in y. In line 3 a set of relevant neighbors RN is computed according to the routing filters RF. A peer is in RN when it is likely to contribute to the final result, i.e., possesses points p_{dom}, so that $\exists p \in S_G : dominated(p, p_{dom})$. Trying to reduce costs by minimizing the size of RN the algorithm takes the risk of missing such dominating points (by not asking a neighbor) as long as P_{in} can be guaranteed. As mentioned above, this risk arises from the approximative nature of histograms that form the basis of routing filters. P_{in} represents the minimum guarantee and P_L represents the actual guarantee. Inequation $P_L \leq P_{in}$ always holds. In order to present first results at an early stage an answer containing S_A and P_L is sent to the predecessor $pred$ (line 4). The remaining allowed risk is divided among all peers in RN (line 5). In the last step (line 6) the query is forwarded to all peers in RN.

When receiving an answer the received probability and result values are combined with the local ones and either forwarded to the next peer or output to the user.

3.4 Preliminary Results: Processing Top-N Queries

Skylines can be regarded as a multidimensional extension of the top-N paradigm. Thus, we applied the idea of giving probabilistic guarantees on processing top-N queries. P_{out} describes the probability that at most C_{out} percent (specified correctness) of the result belongs to the global result that we would retrieve when we asked all the peers in the system. While P_{out} and C_{out} describe the quality of the result, P_{in} and C_{in} describe the boundaries that the algorithm must not exceed. The algorithm applies routing filters based on one-dimensional histograms as presented in subsection 3.2.

Due to limited space we will only present the algorithm's basic workflow and omit details: (1) According to the routing filters calculate a range [$x - dist$,$x + dist$] that is expected to contain the top N result items. (2) Calculate the set of neighbors that are most likely contribute to the final result. (3) Forward the query to these neighbors. Since answers are routed to the initiator in an incremental manner, first results can be output at an early stage. Every time a peer receives an answer it merges its local result R_L with the one received with the answer message R_A: $R_L = R_L \cup R_A$. Afterwards, R_L is pruned to N elements and only those elements that were received with the answer message and that are still contained in R_L are routed to the initiator.

Along with an answer comes a probability P_A as well as a correctness C_A. P_A describes how likely the answer fulfills the correctness of C_A. A peer combines P_A and C_A with its local values P_L (the value of P_L is only 1 when asking all neighbors) and C_L. At the initiator P_L and C_L might now be output to the user for describing the correctness of the currently output result items.

The benefits of this strategy are quite obvious: incrementality, restriction to one round-trip, minimizing the

number of queried peers, and giving probabilistic guarantees for the result. These are the four characteristics that our skyline strategy also complies with.

Based on the averages of several test runs in a network of 100 peers figure 3 shows the influence of the specified correctness C_{in} on the number of queried peers, on the correctness and on the probabilistic guarantee (C-act represents the ratio of global result items that are contained in the output result). The y-axis not only represents the correctness and probability but also the number of queried peers (scaled between 0 and 1, 1 meaning that all peers have been asked).

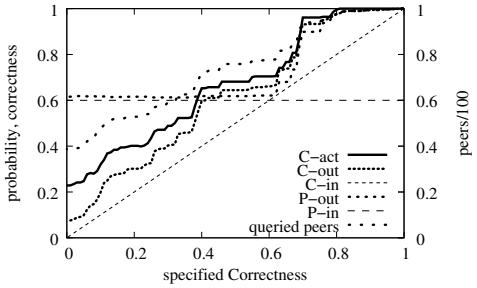

Figure 3: Influence of C_{in} on top-N query processing

Having applied routing filters for processing top-N queries we proved the applicability of our approach for describing result quality by means of probabilistic guarantees. So far, we have only considered one-dimensional top-N queries on the basis of routing filters with one-dimensional histograms. As a natural consequence, there is need for more sophisticated routing filters that can be applied for multidimensional top-N queries and skyline queries. More aspects of our future work are presented in section 4.

4 Conclusion and Outlook

Most existing algorithms for processing skyline queries have been developed and optimized for use in RDBMS and cannot be applied to P2P systems. They try to reduce main memory consumption, IO cost and CPU time at a central processing unit with global knowledge. Since we are developing an efficient strategy for computing skylines distributedly, the problem we are facing resists at a higher level. Dealing with peculiarities of P2P systems, e.g., dynamics and limited knowledge, our focus is developing an efficient strategy for processing skyline queries distributedly over many peers in a network. The basis of that strategy is built by a top-N strategy whose applicability we could already show and whose results we shortly outlined in this paper.

Having presented our current state of work, we now enumerate the main aspects of our future work that will concentrate on improving and implementing our basic probabilistic approach:

- ensuring that data points are not to be removed once they have been output to the user (this might be possible due to the incrementality)
- developing either routing filters on the basis of multidimensional histograms or adapting the algorithm for work with the existing variants
- modifying routing filters for indexing text data
- applying more sophisticated cost models
- for PDMS: taking the goodness of mappings into consideration
- finding optimal rules for sharing the allowed input risk P_{in} among those peers that the query is forwarded to
- examining the impact of dynamics and limited knowledge

Considering all these elements of future work as well as the basic strategy we are expecting our approach to work quite efficiently in P2P systems in general and in PDMS in particular.

References

[1] W.-T. Balke and U. Güntzer. Supporting skyline queries on categorical data in web information systems. In *IMSA 2004*, 2004.

[2] W.-T. Balke, U. Güntzer, and J. Xin Zheng. Efficient distributed skylining for web information systems. In *EDBT*, pages 256–273, 2004.

[3] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE 2001*, pages 421–432, 2001.

[4] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *22nd Int. Conf. on Distributed Computing Systems*, pages 23–32, July 2002.

[5] Y. E. Ioannidis and V. Poosala. Balancing histogram optimality and practicality for query result size estimation. In *ACM SIGMOD'95*, pages 233–244, 1995.

[6] M. Karnstedt, K. Hose, and K.-U. Sattler. Query Routing and Processing in Schema-Based P2P Systems. In *DEXA'04 Workshops*, pages 544–548, 2004.

[7] M. Karnstedt, K. Hose, E.-A. Stehr, and K.-U. Sattler. Adaptive Routing Filters for Robust Query Processing in Schema-Based P2P Systems. In *IDEAS 2005*, July 2005.

[8] D. Kossmann, F. Ramsak, and S. Rost. Shooting stars in the sky: An online algorithm for skyline queries. In *VLDB 2002*, pages 275–286, August 2002.

[9] H. T. Kung, F. Luccio, and F. P. Preparata. On finding the maxima of a set of vectors. *Journal of the ACM*, 22(4):469–476, October 1975.

[10] A. Lotem, M. Naor, and R. Fagin. Optimal aggregation algorithms for middleware. In *PODS'01*, May 2001.

[11] D. Papadias, Y. Tao, G. Fu, and B. Seeger. An optimal and progressive algorithm for skyline queries. In *ACM SIGMOD 2003*, pages 467–478, 2003.

[12] F. P. Preparata and M. I. Shamos. *Computational Geometry - An Introduction*. Springer, 1985.

[13] K.-L. Tan, P.-K. Eng, and B. Chin Ooi. Efficient progressive skyline computation. In *VLDB 2001*, pages 301–310, 2001.

[14] M. Theobald, G. Weikum, and R. Schenkel. Top-k query evaluation with probabilistic guarantees. In *VLDB 2004*, pages 648–659, 2004.