Skyline-based Peer-to-Peer Top-k Query Processing

Akrivi Vlachou [#1], Christos Doulkeridis [#2], Kjetil Nørvåg [*3], Michalis Vazirgiannis [#4]

[#]*Dept. of Informatics, Athens University of Economics and Business, Greece*
[1]avlachou@aueb.gr
[2]cdoulk@aueb.gr
[4]mvazirg@aueb.gr

[*]*Dept. of Computer Science, NTNU, Trondheim, Norway*
[3]Kjetil.Norvag@idi.ntnu.no

***Abstract*— Due to applications and systems such as sensor networks, data streams, and peer-to-peer (P2P) networks, data generation and storage become increasingly distributed. Therefore a challenging problem is to support best-match query processing in highly distributed environments. In this paper, we present a novel framework for top-k query processing in large-scale P2P networks, where the dataset is horizontally distributed to peers. Our proposed framework returns the exact results to the user, while minimizing the number of queried super-peers and transferred data. Through simulations we demonstrate the feasibility of our approach in terms of overall response time.**

I. Introduction

Recently there has been an increased interest to support more flexible query operators than ordinary query types, such as top-k queries. Top-k queries retrieve the objects that best match the user requirements by employing user-specified scoring functions that result in an ordered set of objects containing the best k objects only [1], [2]. Since data generation and storage become increasingly distributed, in this paper we address the efficient computation of top-k queries in large scale peer-to-peer (P2P) networks.

Assuming horizontal data partitioning, the challenge is to provide efficient algorithms for processing top-k queries, i.e. queries that return only the best k results to the user. Users are allowed to specify a monotone function for each query that aggregates some attributes of the objects into a single score value that defines a total ordering, and enables the retrieval of top-k results. There is only limited previous work on supporting top-k queries in P2P systems, and those approaches either assume vertical data partitioning to peers [3], use caching techniques [4], [5], [6], or deliver approximate query result sets [7]. In contrast to these approaches, our work assumes horizontal data partitioning among peers (which is the case for independent sources) and focuses on query processing, rather than caching. This is due to the fact that each user may define his own arbitrary preferences on a query, therefore top-k queries are dynamic and not necessarily re-occurring.

In this paper we present, a framework that supports top-k query processing over horizontally partitioned data stored on peers organized in a super-peer network. In our approach, a super-peer is responsible for gathering all necessary data from its associated peers, in order to be able to answer top-k queries. For a maximum value of K, denoting an upper bound on the number of results requested by any top-k query ($k \leq K$)[1], each peer computes its K-skyband [8] as a pre-processing step. Then, super-peers aggregate the K-skyband sets from their peers, and maintain this aggregated data to answer any incoming top-k query. By exchanging skyline sets [9] (which are a subset of the K-skyband sets) at super-peer level, we are able to route queries to those super-peers that actually contribute to the top-k result. Thus our approach always provides the exact and complete result set.

To summarize, our framework utilizes a peer selection mechanism at super-peer level based on the skyline of each super-peer. Although the skyline operator has received recently considerable attention, its usage for answering top-k queries has not been explored yet. Therefore, we investigate the applicability of the skyline operator for efficiently answering top-k queries for a large class of scoring functions, indicating user-specified preferences, in large P2P networks. Our experimental evaluation shows that our approach performs efficiently and provides a viable solution. Sec. II overviews the related work. In Sec. III we describe our framework. The experimental evaluation is presented in Sec. IV, and we conclude in Sec. V.

II. Related Work

Previous work on top-k query processing in distributed environments [10], [11] has focused on vertically distributed data over multiple sources. Most approaches initially tried to improve some limitations of the Threshold Algorithm [12]. There exists some previous work for top-k queries in P2P over vertically distributed data. In [13], Cao and Wang propose an algorithm called "Three-Phase Uniform Threshold" (TPUT) that was later improved by KLEE [3].

For horizontally distributed data among peers, P2P top-k query processing has been studied in only a few works so far. Balke *et al.* [5] try to minimize the data object traffic induced by top-k processing. However, this approach requires that each query is processed by all super-peers, unless the exact same query has reoccurred before, which is unlikely as there is an infinite number of potential queries posed by different users. A similar approach for unstructured P2P systems is presented in [4], where the main technique is a variant of flooding, followed by a merging score-list step at intermediate peers. In [6], the authors rely on result caching to prune

[1]In the rest of this paper we assume that $k \leq K$.

network paths and answer queries without contacting all peers. Their approach relies on caching techniques, therefore the performance is dependent on the query distribution. Even more important, they assume acyclic networks, which is restrictive for dynamic peer-to-peer networks. Hose *et al.* [7] construct routing filters in the form of histograms, in order to prune query paths and return approximate results. However this approach is not suitable for the case of more than one ranking attributes, since multi-dimensional histograms should be used.

Finally, the skyline operator [9] has recently received considerable attention, but its usage for answering top-k queries has not been explored yet. In [14] the authors improve the performance of ranked join indices based on the concept of dominating sets.

III. P2P Top-k Query Processing

The overall aim is to provide an infrastructure for answering top-k queries in P2P networks, assuming a super-peer architecture. Super-peers SP_i ($1 \leq i \leq N_{sp}$) constitute only a small fraction of the peers P_j ($1 \leq j \leq N_p$) in the network, i.e. $N_{sp} << N_p$. Peers that join the network directly connect to one of the super-peers. Each super-peer maintains links to simple peers, based on the value of its degree parameter DEG_p. In addition, a super-peer is initially connected to a limited set of at most DEG_{sp} other super-peers ($DEG_{sp} < DEG_p$).

A. Preliminaries

Given a data collection O of n objects o_i ($1 \leq i \leq n$), we assume d features $s_j(o_i)$ ($1 \leq j \leq d$) that describe an object $o_i \in O$. We assume that the features s_j are numerical scoring functions with non-negative values that evaluate certain features of database objects. The feature space is defined by the d scoring functions s_j, therefore it is a d-dimensional space. An object $o_i \in O$ can be represented as a point p in the feature space: $p = \{p[1], ..., p[d]\}$, where $p[j] = s_j(o_i)$, is a value on dimension d_j. In the rest of this paper we use the terms object and data point interchangeably. Furthermore, without loss of generality, we assume that smaller score values are more preferable.

In our approach we assume an aggregation function f that is increasingly monotone, i.e. if for every i:$p[i] \leq p'[i]$, then $f(p) = f(p[1], ..., p[d]) \leq f(p'[1], ..., p'[d]) = f(p')$. The restriction of monotonicity is a common property [1], [12] and it conveys the meaning that whenever the score of all dimensions of the point p is at least as good as another point p', then we expect that the overall score of p is as least good as p'. The result of a top-k query is the ranked list of the k objects with lowest $score$ values. In our setting a top-k query $q_k(f)$ takes two parameters: a user specified monotone function f and the number of requested objects k. Notice that both the scoring function and the parameter k may differ for each query and we are interested in retrieving the k objects with the best (minimum) values of the scoring function.

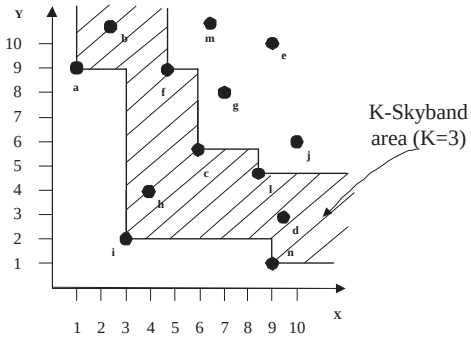


Fig. 1. Skyline and K-skyband

B. Query Processing

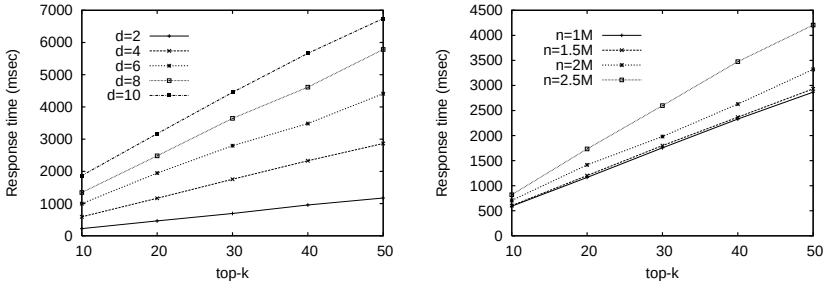
Each peer P_i holds n_i d-dimensional points, denoted as a set O_i ($1 \leq i \leq N_p$). Obviously the size of the complete set of points is $n = \sum_{i=1}^{N_p} n_i$ and the dataset O is the union of all peers' datasets O_i: $O = \cup O_i$. This is the case of horizontal data distribution and each peer maintains its own data objects, such as images or documents. Only the feature values of few selected objects, namely the K-skyband [8] points that represent a subset of each peer's data, are published to the respective super-peer, while the original data is stored at the peer.

Assuming a space D defined by d dimensions $\{d_1, d_2, .., d_d\}$ and given a set of points O, a point $p \in O$ with $p = \{p[1], ..., p[d]\}$ is said to *dominate* another point $q \in O$, if on each dimension $d_i \in D$, $p[i] \leq q[i]$; and on at least one dimension $d_j \in D$, $p[j] < q[j]$. The *skyline* is a set of points $SKY \subseteq O$ which are not dominated by any other point [9]. The points in SKY are called *skyline points*, and in the example of Figure 1, a, i and n are the skyline points. Notice that for example the top-1 result of a query with $f = 0.5 * x + 0.5 * y$ belongs to the skyline, namely the point i is the best match for this query.

Motivated by the fact that the top-1 always belongs to the skyline for any monotone function, we use skyline as a preprocessing step to answer top-k queries. In order to collect the points necessary to answer exact top-k queries, we adopt the concept of K-skyband. The K-skyband query reports the set of points which are dominated by at most $K-1$ other ones. Thus, the conventional skyline is a special instance of the K-skyband, where $K = 1$. In Figure 1, the K-skyband for $K = 3$ includes all points that lie in the line-shadowed area.

Each peer is capable to answer a top-k query by examining only the points that belong to its K-skyband. Each super-peer SP_A maintains the aggregated K-skyband points of its associated peers. In order to keep the information in a manageable size, SP_A computes the K-skyband over the aggregated peer points. This results in a new set of points, also referred to as super-peer's K-skyband, which 1) summarizes the data objects of all peers connected to the super-peer, and most importantly 2) is capable to answer any top-k query.

The remaining challenge is to answer top-k queries over



(a) Scalability with dimensionality (b) Scalability with cardinality

Fig. 2. Response time for uniform data distribution, $N_{sp} = 200$, $N_p = 2000$, $K = 50$.

the entire super-peer network. Instead of flooding queries at super-peer level, we build routing indices based on the skyline points of super-peers. This enables selective querying of those super-peers that are actually responsible for peers with relevant results to the query.

Concluding, our framework supports efficiently top-k queries over distributed data in a super-peer network, utilizing routing indexes based on the skylines sets of super-peers. In the next section we evaluate our framework experimentally.

IV. Experimental Evaluation

We studied the performance of our framework using simulations. The simulator was implemented in Java and all experiments run on a 3.8GHz Dual Core AMD processor with 2GB RAM. The P2P network topology used in the experiments consists of N_{sp} interconnected super-peers in a random graph topology. In our experiments we used synthetic data collections. Each peer generates its own dataset uniformly, which includes random points in a space $[0, L]^d$. We conduct experiments varying the dimensionality (2-10) and the total cardinality (1M-2.5M points) of the dataset. Each time we generate 20 queries with random weightings.

We measure the average response time for the k first results. We assume 50KB/sec as network transfer bandwidth on connections between super-peers, in order to model network delays. The response time is measured as the sum of processing time and network transfer time required for the objects transferred in the network. In Figure 2, a network of $N_p = 2000$ peers is studied, in terms of response time. In Figure 2(a), the response time is presented for different values of d. We increase the dimensionality d from 2 to 10, and we study top-10 to top-50 queries, assuming $K = 50$. The chart shows the response time, for varying dimensionality and shows that the time is increasing with d. We also study the scalability of our framework with respect to the cardinality n of the dataset (see Figure 2(b)). The slightly increasing response time with cardinality is mainly due to the increasing size of K-skyband, and this leads to higher processing time.

V. Conclusions

In this paper we presented a framework for answering top-k queries in a P2P network where data is horizontally distributed to peers. Relying on a super-peer architecture, a top-k query is forwarded among super-peers, in such a way that the amount of transferred data is minimized. To achieve this goal, we use the skyline and K-skyband sets to efficiently support top-k query processing. We provide an experimental evaluation through simulations, showing that our framework performs efficiently and provides a viable solution when a large degree of distribution is required.

Acknowledgment

This research project is co-financed by E.U.-European Social Fund (75%) and the Greek Ministry of Development-GSRT (25%). Prof. Dr. Vazirgiannis was supported by the Marie Curie Intra-European Fellowship NGWeMiS: Next Generation Web Mining and Searching (MEIF-CT-2005-011549).

References

[1] S. Chaudhuri and L. Gravano, "Evaluating top-k selection queries." in *Proc. of VLDB*, 1999.

[2] V. Hristidis, N. Koudas, and Y. Papakonstantinou, "PREFER: A system for the efficient execution of multi-parametric ranked queries," in *Proc. of SIGMOD*, 2001.

[3] S. Michel, P. Triantafillou, and G. Weikum, "KLEE: a framework for distributed top-k query algorithms," in *Proc. of VLDB*, 2005.

[4] R. Akbarinia, E. Pacitti, and P. Valduriez, "Reducing network traffic in unstructured P2P systems using top-k queries." *Distributed and Parallel Databases*, vol. 19, no. 2-3, pp. 67–86, 2006.

[5] W.-T. Balke, W. Nejdl, W. Siberski, and U. Thaden, "Progressive distributed top-k retrieval in peer-to-peer networks," in *Proc. of ICDE*, 2005.

[6] K. Zhao, Y. Tao, and S. Zhou, "Efficient top-k processing in large-scaled distributed environments," *Data Knowl. Eng.*, vol. 63, no. 2, pp. 315–335, 2007.

[7] K. Hose, M. Karnstedt, K.-U. Sattler, and D. Zinn, "Processing top-N queries in P2P-based web integration systems with probabilistic guarantees." in *Proc. of WebDB*, 2005.

[8] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "Progressive skyline computation in database systems." *ACM Trans. Database Syst.*, vol. 30, no. 1, pp. 41–82, 2005.

[9] S. Börzsönyi, D. Kossmann, and K. Stocker, "The skyline operator," in *Proc. of ICDE*, 2001.

[10] S. Chaudhuri, L. Gravano, and A. Marian, "Optimizing top-k selection queries over multimedia repositories," *IEEE Trans. on Knowledge and Data Engineering*, vol. 16, no. 8, pp. 992–1009, 2004.

[11] U. Güntzer, W.-T. Balke, and W. Kießling, "Optimizing multi-feature queries for image databases," in *Proc. of VLDB*, 2000.

[12] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," in *Proc. of PODS*, 2001.

[13] P. Cao and Z. Wang, "Efficient top-k query calculation in distributed networks," in *Proc. of PODC*, 2004.

[14] P. Tsaparas, T. Palpanas, Y. Kotidis, N. Koudas, and D. Srivastava, "Ranked join indices." in *Proc. of ICDE*, 2003.