Cost-Aware Skyline Queries in Structured Overlays

[Work-in-Progress]

Marcel Karnstedt, Jessica Müller, Kai-Uwe Sattler
Department of Computer Science and Automation,
TU Ilmenau, Germany

Abstract

Recently, systems providing access to extremely large data collections, managed in a distributed manner, gain emerging attention. A promising approach to implement the physical layer of such systems are structured overlays based on the peer-to-peer paradigm. On the level of query expressiveness, ranking queries like skyline queries are predestinated for providing a fast but concise overview of the data. The problem is that structured overlays are able to handle dynamic and unreliable environments, but usually support only limited query processing capabilities, which are very improper for processing such sophisticated queries. In this work, we propose three variants of a skyline operator and two extensions, especially suitable for efficient determination of skylines in the before mentioned overlay systems. Additionally, we back the introduced approaches on an appropriate cost model, ready for implementing adaptive cost-based query optimization.

1. Skyline Queries in Distributed Data Systems

In recent time, there is a raising need in applications allowing for managing data provided by separate users of large communities. In most cases, these applications particularly demand for distributed data management following the peer-to-peer (P2P) paradigm. Examples of systems based on the idea of P2P on several different application levels are recently popular folksonomies, file-sharing systems, scientific knowledge bases, communication platforms and desktop data collections. The latter are an example of data systems where special needs for managing structured data, possibly enriched by semantic data, arise.

A foundation of such systems is to provide access to extremely large and often heterogeneous data collections. Based on this background, providing capabilities for processing ranking queries become especially important. Popular representatives are top-N queries and *skyline* queries [3]. Queries of these types allow for providing a fast and compact, but concise view over large data, optimized for one or preferential multiple ranking functions.

On the physical layer, structured overlay systems, e.g., Distributed Hash Table (DHT) based overlays like P-Grid [1], CAN [10] or CHORD [11], offer highly developed and well-performing functionalities for achieving scalability and robustness mandatory in the introduced applications. They provide structures ready to build loosely coupled and widely distributed, but stable environments. In general, these overlays provide capabilities for efficiently inserting and deleting data, as well as processing exact-match lookups and latterly range queries. But, usually these approaches lack capabilities for processing sophisticated queries like structured queries and the before mentioned ranking queries.

In this work, we analyze approaches for processing skyline queries on structured data in structured overlays. We specifically base our considerations on *UniStore*, a triple-based universal storage [6, 7] on top of the P-Grid DHT overlay system. We propose a skyline operator for UniStore and discuss several extensions tailor-made for highly distributed management of structured data as introduced before. Additionally, we introduce accurate cost formulas for the implemented algorithms, allowing for adaptive high-performance query processing. Note that, despite UniStore is based on P-Grid, the applied principles could also easily be implemented on top of other DHTs like CAN or CHORD. Following, the skyline algorithms presented in this work are applicable to structured overlays (and other suitable types of distributed systems) in general.

Skyline queries are intensively researched in the database community [3, 8, 9, 2, 12]. A well-known definition is based on the *dominance* relation: given two or more ranking functions, the skyline of a data set is formed by all points of the set not dominated by any other point, i.e., a skyline member is ranked higher than a non-member in at least one dimension of the skyline. In mathematics and economy this set is known as the *Pareto optimum*. Members of a skyline represent best possible trade-offs between all ranking goals, which is for instance extremely helpful in decision making tasks.

Skylines are easy to exemplify in recommender and

business-to-people systems, e.g., hotel recommender systems or economical applications like car trading. A popular example, easy to understand for the non-initiated reader, is to ask for all hotels which are preferably close to the beach (min(distance)) *and* preferably cheap (min(price)). In figure 1 we show (a part of) an alternative example schema for author data in a public data management scenario. The figure shows several entities and their attributes, as well as relations between them. On the physical level, entities could be represented by tuples holding the attribute values. An example query in this scenario could ask for the skyline of authors that reaches from the youngest authors to those authors published the most publications. In this work, we focus on the most popular ranking functions minimize and maximize. Note that, skyline queries become even more powerful in conjunction with advanced querying features like similarity queries. For example, a user could be interested only in authors published at ICDE workshops – considering possible data heterogeneities it could be useful to allow an edit distance of up to 2 to the term 'ICDE' in order to ignore typos and similar.

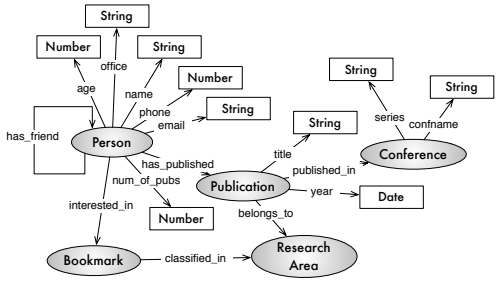

Figure 1: Example schema

UniStore is a universal storage platform particularly providing such query capabilities whilst implementing a highly scalable and robust high-performance distributed data system. Thus, in the next section we briefly introduce the general architecture our approach is based on by introducing the principles behind UniStore. Afterwards we discuss efficient skyline algorithms in section 3. A brief overview of a corresponding cost model is provided in section 4. We conclude our work by discussing current and open issues in section 5.

2. Data Management

We base our approach on the idea of vertically partitioned data, similar to the idea of the Resource Description Framework (RDF). Assume structured data, e.g., relational data, where tuples consist of multiple values corresponding to attributes defined in a schema. Each tuple t with attributes $a_1, \ldots, a_n$ is decomposed into a set of n triples $(OID_t, a_i, v_i), i = 1 \ldots n$, where OID_t is a system generated object ID unique for each tuple, a_i is an attribute identifier and v_i the corresponding value of that attribute. The resulting set of triples is distributed among all peers in the underlying overlay system. In DHT-based overlays, a peer is responsible for a partition of the key space generated by applying the hash function(s), which results in the responsibility for the hashed triples. We assume the existence of several indexes, built by applying hash function(s) h on chosen parts of all or some triples. Thus, a triple is stored (possibly multiple times) at a corresponding peer responsible for the resulting hash key or keys. We specifically assume two indexes: The first is based on $h(OID)$. Thus, different OIDs are distributed among different peers, but an original tuple can be completely materialized at one responsible peer. The second index is based on $h(a)$. In this way, different attributes reside on different peers. In several structured overlays all triples of one attribute will map to a closed range of hash keys and therefore reside in a partition of peers close to each other with respect to the constructed overlay. For example, in P-Grid, which constructs a binary prefix tree on the key space with peers located on the leaf level of this tree, all peers responsible for an attribute a are located in a common sub-tree. Consequently, we assume the support of efficient range queries in order to query for triples of a single attribute.

In a distributed data system following the general ideas of database systems, queries are represented by query plans consisting of different operators. For efficient processing in dynamic environments, an according optimizer should be able to decide between different implementations for one logical operator. Thus, an appropriate cost model is needed. Following this approach, we provide different skyline operators and define according cost formulas for each of them.

3. The Frame-Skyline Operator

Due to the triple-based storage and the special requirements of large-scaled distributed systems, centralized skyline algorithms like Block-Nested-Loops (BNL) [3], Divide-and-Conquer [3], Nearest-Neighbor [8] or Branch and Bound Skyline (BBS) approach [9] cannot be applied directly, at least not in a satisfyingly efficient manner. Research of skyline processing in distributed systems resulted in only a few approaches until now. [2] proposes an algorithm that exploits the TA principle of sorted lists for processing skylines for web information systems. There are three major drawbacks: first, a specialized network structure where the processing node has direct access to all other nodes. Second, this approach requires sorted lists of all objects. And third, it would not scale with large networks. Another work that considers skyline processing in distributed environments is DSL [12]. It progressively computes constrained skylines using a CAN built on top of the dimensions of later skyline queries. Thus, this approach is not general enough. Efficient skyline processing for unstructured P2P systems, enriched by the option to determine *relaxed* skylines, is introduced in [5]. This approach utilizes specialized spatial index structures and is based on different principles

and systems than the work presented here.

In order to reduce bandwidth consumption, network load and answer times the *Frame-Skyline* operator proposed in this work aims for minimizing the set of skyline candidates from the beginning of processing. In the following, we exemplary consider a two-dimensional skyline query looking for the minimum in each dimension. Elements from the data set that are minimal in one dimension are definitely part of the skyline – they cannot be dominated by any other item. In parallel, these elements narrow the search space for the second attribute. Figure 2 illustrates this: the point minimal in dimension y (marked with a circle) provides the maximal value for dimension x we have to consider. Any other point revealing a higher x-value cannot be part of the skyline, because it is definitely dominated in both dimensions. Based on the values of points minimal in one dimension, we can thereby narrow the search space for all attributes as illustrated by the pictured frame.

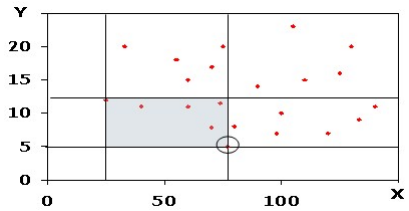

Figure 2: Frame-skyline approach

Based on this, we can use the overlay's query processing capabilities to efficiently contact only those peers responsible for elements included in the resulting frame. We developed three basic algorithms: central, distributed, and hybrid. Each of them starts by determining the minima, maxima respectively, for all queried dimensions and building the corresponding tuples. From these tuples ranges for each attribute are determined. From this point on, the three versions differ.

The central version queries these ranges using range queries. From the result those tuples are determined that occur in all dimensions, which build a first skyline. In multi-dimensional (> 2) skyline queries also projections to the search space may be part of the result. Thus, in this case all elements of the sub-spaces of the search space are determined from the range queries' results and the complete tuples are materialized. A central BNL algorithm finally computes the global skyline.

In the distributed variant we try to overcome problems resulting from the "get all data and compute locally" character of the central variant and the waiting states in it. After determining the frame borders separate local skyline queries are sent to all peers responsible for the resulting range of an attribute. In analogy to the central version, each contacted peer uses range and materialize queries to collect the remaining attributes and computes local skylines. The results are sent back to the initiating peer, which computes the final global skyline. In contrast to the central version, the distributed approach comes along with higher communication efforts, but provides, due to the distributed processing, better query answer times. See figure 3 for illustration of the approach.

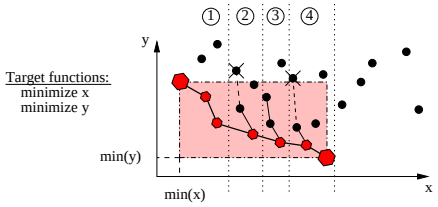

Figure 3: Distributed skyline processing

The hybrid processing is similar to the distributed version, but is aimed for avoiding the querying of one object ID multiple times. This occurs during the last state of processing when all local candidates are materialized, because triples with the same ID reside on different peers. Only the object IDs from all additional candidates of all sub-spaces are sent to the query initiator. There, duplicates are removed and the final tuples are materialized. Based on the local skylines and the materialized candidates, the final skyline is computed. This reduces the communication overhead at costs of answer times, because tuples are not materialized multiple times.

An important challenge in loosely coupled and in places unreliable systems is to avoid depending on waiting states. In the introduced operators, there are several situations where sub-queries are initiated and answered by multiple replies, e.g., determining the range of a dimension using range queries. This leads to the difficult question of how long to wait for replies, as there is no reliable knowledge about how many peers are contacted and how many will reply to these sub-queries. One idea is to use constant waiting times, which is not satisfyingly efficient. Another approach could be based on estimating result sizes and completeness of query results. But, such estimations are very hard and often inaccurate. We propose two extensions of each of the three, as introduced *blocking*, operators: *online* and *evolving*.

In the online extension parts of the skyline are transmitted to the user as soon as it is certain that they belong to the final skyline. This first skyline can be determined after receiving all range query answers. In the evolving version even elements are transmitted which could finally be replaced by dominating elements determined later – the skyline may evolve. This can be done after receiving any range query answer. Both extensions are tailor-made for dynamic and unreliable environments as P2P systems are.

4. Cost Estimation

It is not possible to state that one of the introduced approaches is best-suited in general for the aimed applications.

Each of them bears several advantages and disadvantages, and the best choice depends on the current environment, particularly on the data distribution, size of the overlay, number of peers responsible for a certain attribute etc. These circumstances are preferably covered by an optimizer which utilizes an appropriate cost model [6]. Using this cost model to estimate costs a priori, the query processor is able to adaptively decide at each peer which is the cost-optimal choice to continue. In this section, we briefly discuss cost formulas for the introduced skyline operators.

In a structured overlay system such as P-Grid, the most relevant cost factors are the number of messages and hops needed to process a query and the amount of bandwidth consumed during processing. Another important aspect are anticipated query answer times – but, times are predictable very hard in such environments and usually are reflected by the number of hops. As each sophisticated query is based on the routing and lookup mechanisms of the underlying overlay, we base cost estimations for skyline queries on statistical values and costs provided by it.

Due to the complexity of our considerations, we exemplify the developed cost estimations by listing the (shortened) formulas for determining the worst-case number of messages C^m for the central and the distributed variant of skyline processing:

- central: $C^m = 4 \cdot a + 2 + \sum_{i \in I} 2 \cdot r_i$
- distributed: $C^m = 4 \cdot a + 4 \cdot r_A + r_A \cdot \sum_{i \in I \wedge i \neq A} (2 \cdot r_i)$

The used variables symbolize the following values provided by P-Grid: a: number of skyline attributes, A: chosen skyline attribute queried by range, I: set of all skyline attributes, r_X: number of peers responsible for attribute X. An important message is, that the number of messages in the distributed variant depends on r_A (the number of peers processing local skyline queries), whereby the central variant only depends on the number of skyline attributes. This symbolizes the message overhead caused by the applied parallelism in the distributed approach. The formulas for the number of hops would show the gain of parallelism, but we omit them here due to space constraints.

In this form, the estimations are quite accurate. As not all needed factors will be available, some of them will have to be approximated. As in any other query optimizer, the point is not to accurately determine costs but to determine accurate relations between different alternatives. We developed such formulas for the best-case, worst-case and average-case for each operator and extension. A current issue is to integrate them into UniStore's cost model (among others the actual determination/approximation of needed cost factors).

5. Open Issues

In this work, we presented three different operators and sound extensions for efficiently processing skyline queries on structured data in structured overlay systems. The most current issue is to prepare an extensive evaluation, both in LANs and PlanetLab [4] as a testbed especially made for this purpose. The aim of this evaluation is to show the correctness of the introduced cost estimations and that all approaches are suitable for implementing scalable, dynamic and robust query processing. Special concern is placed on evaluating the online and evolving extensions, as they promise to be best-suited for dynamic and unreliable P2P environments.

Future issues include optimized processing of multi-dimensional skylines, possibly aided by specialized multi-dimensional index structures.

References

[1] K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, and R. Schmidt. P-Grid: A Self-organizing Structured P2P System. *ACM SIGMOD Record*, 32(3), 2003.

[2] W. Balke, U. Güntzer, and J. X. Zheng. Efficient Distributed Skylining for Web Information Systems. In *EDBT'04*, pages 256–273, 2004.

[3] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE'01*, pages 421–432, 2001.

[4] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. PlanetLab: An Overlay Testbed for Broad-Coverage Services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12, 2003.

[5] K. Hose, C. Lemke, and K. Sattler. Processing Relaxed Skylines in PDMS Using Distributed Data Summaries. In *CIKM*, 2006.

[6] M. Karnstedt, K. Sattler, M. Hauswirth, and R. Schmidt. Cost-Aware Processing of Similarity Queries in Structured Overlays. In *Sixth IEEE International Conference on Peer-to-Peer Computing*, 2006.

[7] 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*, 2007. To appear.

[8] D. Kossmann, F. Ramsak, and S. Rost. Shooting stars in the sky: An online algorithm for skyline queries. In *VLDB'02*, pages 275–286, August 2002.

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

[10] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. In *Proceedings of ACM SIGCOMM 2001*, 2001.

[11] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM*, pages 149–160. ACM Press, 2001.

[12] P. Wu, C. Zhan, Y. Feng, B. Zhao, D. Agrawal, and A. E. Abbadi. Parallelizing skyline queries for scalable distribution. In *EDBT'06*, pages 112–130, 2006.