The Journal of Systems and Software 93 (2014) 69–83

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Efficient distributed skyline computation using dependency-based data partitioning

Bo Yin [a,b], Siwang Zhou [a,*], Yaping Lin [a], Yonghe Liu [c], Yupeng Hu [a]

[a] College of Information Science and Engineering, Hunan University, Hunan, Changsha 410082, China
[b] School of Computer and Communication Engineering, ChangSha University of Science and Technology, Changsha 410114, China
[c] Department of Computer Science and Engineering, University of Texas at Arlington, Arlington, TX 76019, USA

ARTICLE INFO

Article history:
Received 14 July 2013
Received in revised form
24 December 2013
Accepted 6 March 2014
Available online 13 March 2014

Keywords:
Skyline query
Distributed systems
Data partitioning

ABSTRACT

Skyline queries, together with other advanced query operators, are essential in order to help identify sets of interesting data points buried within huge amount of data readily available these days. A skyline query retrieves sets of non-dominated data points in a multi-dimensional dataset. As computing infrastructures become increasingly pervasive, connected by readily available network services, data storage and management have become inevitably more distributed. Under these distributed environments, designing efficient skyline querying with desirable quick response time and progressive returning of answers faces new challenges. To address this, in this paper, we propose a novel skyline query scheme termed MpSky. MpSky is based on a novel space partitioning scheme, employing the dependency relationships among data points on different servers. By grouping points of each server using dependencies, we are able to qualify a skyline point by only comparing it with data on dependent servers, and parallelize the skyline computation among non-dependent partitions that are from different servers or individual servers. By controlling the query propagation among partitions, we are able to generate skyline results progressively and prune partitions and points efficiently. Analytical and extensive simulation results show the effectiveness of the proposed scheme.

1. Introduction

Recently skyline queries (Börzsönyi et al., 2001) have attracted tremendous amount of attention as they can help users to make intelligent decisions over complex data when multiple criteria are considered, without requiring user-defined scoring functions. The *Skyline* of a data set composed of *d*-dimensional points returns those points that are not dominated by any other point. A point dominates another point if it is as good or better in all dimensions and better in at least one dimension, with respect to some specific preference (e.g., min, max) on each dimension. A classical example is where a tourist may want to minimize both the price of the room and the distance from the hotel to the beach. The skyline operator provides a viable solution to identify all potentially "interesting" hotels for which there is none cheaper and closer to the beach. This is illustrated in Fig. 1, where each point represents a hotel with price and distance as coordinates. Here hotels *a*, *b*, *c*, and *d* are in the skyline set, representing the best possible tradeoffs between price and distance from the beach.

As computing infrastructures become increasingly pervasive, connected by readily available network services, data storage and management have become inevitably more distributed. A plethora of applications now collect data from distributed sites and compute query results based on the collective view of all the data. Lacking global knowledge, skyline query processing over these data in the distributed environments faces inherent challenges. Furthermore, as sizes of data sets become larger and larger, distributed skyline computation can be prohibitively complex and expensive. Efficient algorithms are demanded to retrieve the results in reasonable response time.

One of the desirable properties of a skyline algorithm is progressiveness, where the query algorithm reports partial results back as early as possible rather than all in a flock (Kossmann et al., 2002; Papadias et al., 2003). Progressiveness is critical when users need to react and formulate competitive decisions in near real-time. In distributed skyline computation where the query time may be relatively long, progressive returning of answers could alleviate the user's inconvenience to some certain.

In this paper, we focus on minimizing the response time and progressively generating skylines in distributed skyline computation.

* Corresponding author. Tel.: +86 139 73185941.
E-mail addresses: yinbo@hnu.edu.cn (B. Yin), swzhou@hnu.edu.cn (S. Zhou), yplin@hnu.edu.cn (Y. Lin), yonghe@cse.uta.edu (Y. Liu), yphu@hnu.edu.cn (Y. Hu).

http://dx.doi.org/10.1016/j.jss.2014.03.021
0164-1212/

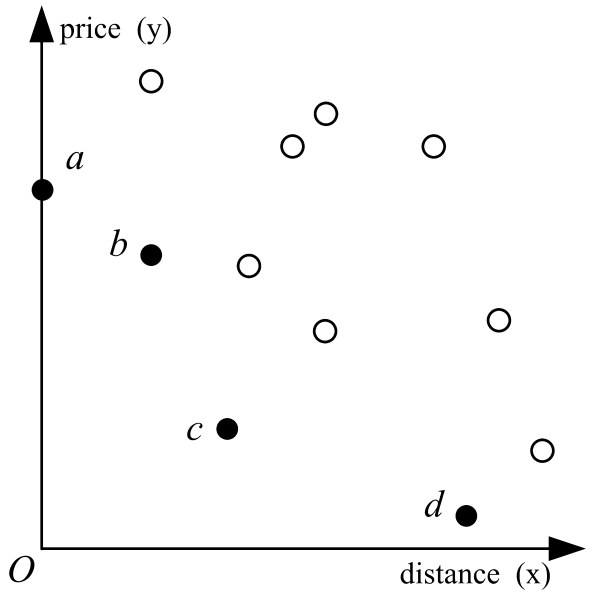

Fig. 1. Skyline example.

In our setting, the coordinator server, which can directly communicate with all participating servers, distributes the query request to all servers and collects their results (Cui et al., 2008; Zhang et al., 2009; Junior et al., 2011; Zhu et al., 2009). We allow arbitrary horizontal partitioning, i.e., each server stores autonomously a fraction of the data and a server has all the attributes.

Several approaches have been suggested for distributed skyline query processing. Nevertheless, most of these approaches have been focused on the initial data partitioning, i.e., partitioning the dataset to a set of servers, for parallel processing and fair workload (Vlachou et al., 2008; Wu et al., 2005; Valkanas and Papadopoulos, 2010; Wang et al., 2007, 2009; Chen et al., 2008; Köhler et al., 2011; Li et al., 2006; Afraiti and Koutris, 2012). The most related works to ours are Cui et al. (2008), Junior et al. (2011), which employ an arbitrary horizontal partitioning, and accelerate skyline queries by controlling the query forwarding order among servers. Their major principles are two-fold. First, incomparable servers (i.e., servers do not depend on each other) are processed in parallel. Second, for depending servers, they are queried in the order such that the gain that can be achieved by filtering is maximized, consequently reducing the amount of transferred data (also the response time). In distributed data sets with arbitrary horizontal partitioning, the probability that two servers are not incomparable is high and increases with dimensionality. Therefore, solutions proposed in Cui et al. (2008), Junior et al. (2011) have limited parallelism and their progressiveness for incomparable server groups is poor.

In this paper, to address the above problems, we first observe that data points of a server may have different "dominance likelihoods" with data on other servers. A "dominance likelihood" between two servers denotes that data of one server S_j must dominates at least some points of the other server S_i, that is, S_i depends on S_j. Suppose that the server set that S_i depends on is V. If we group points of S_i according to the dependency relationships, skyline computations can be parallelized among any server in V and data subset (of S_i) that does not depend on this server. Furthermore, we are able to instantly output the local skyline points of partitions that do not depend on any other server since these local skyline points must belong to the global results, and qualify a rest skyline point by only comparing it with points on dependent servers, thereby progressively producing skyline results without compromising correctness.

Based on these observations, we propose a novel dependency-based portioning scheme, which divides data on each server into partitions according to their dependency relationships. In addition to progressiveness, this approach results in the parallelism not only among non-dependent/incomparable servers but also among dependent servers, thereby speeding up the query processing. Our partitioning scheme also enhances the pruning power of the filtering point, when we enforce a query forwarding order among partitions by exploiting dependencies. For any pair of resulted partitions, they are either incomparable, or data of one partition must be dominated by at least one point of the other partition. Thus, given a partition P, we can select from dependent partitions which usually precedes P the filtering point to discard dominated points located in P effectively. Additionally, as the query propagates, our approach can progressively generate partial results as a data point only needs to be compared with those from dependent partitions.

The main contributions of this paper are the following: (1) for distributed skyline computation, we propose a data partitioning-based approach termed MpSky that is optimized for response time while maintaining progressiveness; (2) we propose a novel data partitioning scheme, which groups points of each server using dependencies, such that a skyline point can be identified by only comparing it with data on dependent servers, and parallel computation can be performed on inter-server and intra-server partitions; (3) we present an ordering method to generate skyline results progressively and benefit effective pruning. Our analysis shows that the execution plan has a bounded value of height; (4) we perform a comprehensive experimental evaluation to show that the superiority of our proposed MpSky over the state-of-the-art algorithm (Cui et al., 2008).

The rest of the paper is organized as follows. Section 2 surveys the previous work. We present necessary preliminaries in Section 3 followed by overview of MpSky in Section 4. In Section 5, we present the dependency-based data partitioning scheme, while in Section 6 we describe the query plan among partitions. Section 7 presents the query execution guided by the query plan. Section 8 presents an extensive experimental evaluation that demonstrates the superiority of MpSky. We conclude in Section 9.

2. Related work

Recently skyline computation (Börzsönyi et al., 2001) has attracted tremendous attention in a variety of distributed systems including web information systems (Balke et al., 2004), parallel systems (Vlachou et al., 2008; Köhler et al., 2011), peer-to-peer systems (Wu et al., 2005; Valkanas and Papadopoulos, 2010; Wang et al., 2007, 2009; Chen et al., 2008; Hose et al., 2006; Vlachou et al., 2007, 2010; Fotiadou and Pitoura, 2008), mobile ad-hoc networks (Huang et al., 2006; Vlachou and Norvåg, 2009), as well as more generic distributed systems (Cui et al., 2008; Zhang et al., 2009; Junior et al., 2011; Zhu et al., 2009). In the following, we provide an overview of existing approaches for distributed skyline computation. One of the first algorithms in the distributed domain, by Balke et al. (2004), address skyline operation over web databases, where different dimensions are stored in different data sites (vertical data distribution). Later, most of the related work has focused on highly distributed environments, such as peer-to-peer networks, assuming that sources store common attributes (horizontal data distribution). Such approaches can be classified into two categories.

In the first category, space partitioning is assumed among peers and each peer is responsible for a disjoint partition of the data space. Toward this, a structured peer-to-peer or tree-based network overlay is employed. The system splits the data in a way that the system can first visit the servers with higher probability of possessing skyline points. The first work for skyline query processing over disjoint