Ad Hoc Networks 35 (2015) 105–115

Contents lists available at ScienceDirect

Ad Hoc Networks

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

Efficient subspace skyline query based on user preference using MapReduce

Yuanyuan Li[a,b], Zhiyang Li[a,*], Mianxiong Dong[c], Wenyu Qu[a], Changqing Ji[d], Junfeng Wu[a,e]

[a] School of Information Science and Technology, Dalian Maritime University, Dalian, China
[b] School of Software, Dalian Jiaotong University, Dalian, China
[c] Department of Information and Electronic Engineering, Muroran Institute of Technology, Hokkaido, Japan
[d] College of Physical Science and Technology, Dalian University, Dalian, China
[e] School of Information Engineering, Dalian Ocean University, Dalian, China

ARTICLE INFO

Article history:
Received 27 November 2014
Revised 15 June 2015
Accepted 10 July 2015
Available online 23 July 2015

Keywords:
Subspace skyline query
MapReduce
Pruning strategy
Grid
User preference

ABSTRACT

Subspace skyline, as an important variant of skyline, has been widely applied for multiple-criteria decisions, business planning. With the development of mobile internet, subspace skyline query in mobile distributed environments has recently attracted considerable attention. However, efficiently obtaining the meaningful subset of skyline points in any subspace remains a challenging task in the current mobile internet. For more and more mobile applications, subspace skyline query on mobile units is usually limited by big data and wireless bandwidth. To address this issue, in this paper, we propose a system model that can support subspace skyline query in mobile distributed environment. An efficient algorithm for processing the Subspace Skyline Query using MapReduce (SSQ) is also presented which can obtain the meaningful subset of points from the full set of skyline points in any subspace. The SSQ algorithm divides a subspace skyline query into two processing phases: the preprocess phase and the query phase. The preprocess phase includes the pruning process and constructing index process which is designed to reduce network delay and response time. Additionally, the query phase provides two filtering methods, SQM-filtering and ε-filtering, to filter the skyline points according to user preference and reduce network cost. Extensive experiments on real and synthetic data are conducted and the experimental results indicate that our algorithm is much efficient, meanwhile, the pruning strategy can further improve the efficiency of the algorithm.

1. Introduction

Skyline query has attracted an increasing amount of attention over the past few years. Skyline query has been widely applied in highly mobile distributed environments for multiple-objective decisions. For example, a tourist may search for a suitable hotel using a mobile phone network when he/she arrives at the airport of a city.

Due to large-scale data and high processing costs, it is not possible to compute the skyline points using terminals such as mobile phones. The number of skyline points quickly increases when the amount of data increases, particularly for increasing trend of applications to deal with big data, for example, big data on the social network. To address this problem, skyline computation in a distributed environment is the best option. The authors in [1–4] proposed some approaches for skyline processing in P2P network. These methods are

* Corresponding author. Tel.: +86 13904265439.
E-mail address: lizy0205@dlmu.edu.cn (Z. Li).

http://dx.doi.org/10.1016/j.adhoc.2015.07.006
1570-8705/© 2015 Elsevier B.V. All rights reserved.

Table 1
Dataset of hotels.

ID	Price (RMB)	Mileage (KM)	Occupancy rate
p_1	200	7	0.5
p_2	200	3	0.6
p_3	250	5	0.8
p_4	500	3	0.7
p_5	150	9	0.6
p_6	200	9	0.5
p_7	300	6	0.7
p_8	500	5	0.8
p_9	300	7	0.9

restricted to their own specific scenarios, such as overlay network, and cannot be adapted for the aforementioned scenario.

Furthermore, skyline query can be computationally expensive when it provides numerous candidate attributes. In many applications, various users may focus their attention on different subsets of the attributes according to their own interests. Pei et al. and Tao [5,6] proposed efficient skyline algorithms in subspaces that can retrieve the skylines in a user-defined subset of attributes. MapReduce framework [7] is a programming model for processing large datasets using a distributed, parallel algorithm on a cluster. A considerable amount of work has been conducted to migrate traditional skyline algorithms to the MapReduce framework, such as [8–11], but they cannot support subspace skyline query.

The subspace skyline query assumes every attribute to be of equal importance, and the difference in the importance of attributes is not considered. Bias to some attributes of greater interest was considered in [12], in which the authors proposed the Weighted Dominant Skyline. However, it is not possible for users to provide the weight assignment without any initial knowledge of the dataset. In this paper, we only require users to provide the ranks of their attributes of interest.

Taking Table 1 as an example, a dataset $P = \{p_1, p_2, \ldots, p_9\}$ about hotels contains 3 attributes: the price, the distance to the airport (Mileage) and the occupancy rate. Assuming that the attributes are of equal importance, there are 4 skyline points in P: p_1, p_2, p_5 and p_6. However, in many cases, the relative importance of attributes is often different. For example, when a tourist is very tired, the distance to the airport is crucial for him/her, and the other attributes are secondary. In this case, although p_5 and p_6 are skyline points, the tourist does not consider these points because the distance to the airport is too far. In the above example, the interesting subset of skyline points is $\{p_1, p_2\}$.

The aim of a skyline query is to help users manually make a decision. As the size of the dataset increases, the size of skyline points becomes too large. There are many meaningless skyline points to report, such as p_5 and p_6 in the above example. In this paper, we present a new algorithm which we call Subset Skyline Query or SSQ for short to return the meaningful subset of subspace skyline points.

The major contributions of this work are summarized as follows: (1) We present a system model for implementing the skyline operator using MapReduce in any subspace, in which the requirements of mobile internet and big data are carefully considered. (2) We propose the algorithm of the Subspace Skyline Query (SSQ) which can obtain the meaningful subset skyline of skyline points. In the first phase, we design a pruning strategy based on grid to reduce the network delay and response time. It can effectively prune out non-skyline points in advance. The constructing index process is also used to support subspace skyline query. In the second phase, we provide two filtering methods, called SQM-filtering and ε-filtering, to filter the skyline points according to user preference and reduce network communication. (3) We conduct experiments on real and synthetic data. Experimental evaluations show that SSQ can significantly improve the efficiency of the subspace skyline query over big data.

The remainder of this paper is organized as follows. In Section 2, we review the previous work related to skyline query processing. Section 3 provides the useful preliminaries. In Section 4, a system model is presented. Section 5 describes the implementation of the SSQ algorithm in the parallel programming framework of MapReduce. Section 6 presents experimental evaluations that demonstrate the efficiency of the proposed algorithm. Finally, we conclude the paper with a summary of our results in Section 7.

2. Related work

The skyline operator is useful for extracting interesting data points from a dataset. Over the past decades, a considerable number of research works have reported on the skyline operator and its variants. Börzsönyi et al. [13] first introduced the skyline operator into the relational database and proposed two algorithms: Block Nested Loop (BNL) and Divide-and-Conquer (D&C). Chomicki et al. [14] presented Sort-Filter-Skyline (SFS) as a variant of BNL, which can immediately eliminate objects dominated by others in the presorted dataset. The primary shortcoming of the above algorithms is their dependence on memory capacity. Many skyline query algorithms based on indexing have been proposed, such as the nearest neighbor [15] and branch-and-bound skyline [16] algorithms. Due to the dimension curse, the size of the skyline is typically large. Consequently, many variants of skylines have been proposed. Subspace skyline query were first discussed by Pei et al. [5] and subsequently elaborated in later works [6]. The key concept of a subspace skyline query is to implement the skyline operator in the most interesting subset of all dimensions by the user. Clearly, skyline query processing using the index structure mentioned in [15,16] over all dimensions, such as R-tree, cannot support subspace skyline query. All of these works assumed that skyline query is performed on centralized systems. Unfortunately, a large amount of data can result in low efficiency for skyline query performed in a centralized setting.

Deviating from skyline query on centralized systems, significant research effort has been devoted to implementing the skyline operator in distributed environments. Skyline queries in P2P networks were discussed in [2–4], in which unstructured peers or routing indexes were used to identify relevant peers. Skyline processing has also been studied in other distributed environments, such as web information systems [17–19]. They are not adapted for our scenario because skyline processing over big data cannot be performed in lightweight terminals. Motivated by the fact