International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.3, Issue.3, May-June. 2013 pp-1400-1402 ISSN: 2249-6645

Skyline Query Processing using Filtering in Distributed Environment

Shaik Shafi, [1] Sayeed Yasin[2]
[1]M.Tech, Nimra College of Engineering & Technology, Vijayawada, A.P., India
[2]Asst.Professor, Dept.of CSE, Nimra College of Engineering & Technology, Vijayawada, A.P., India

Abstract: *Skyline is used in a distributed database, because the database will not be in one system. It will be stored in multiple systems reside at different locations, if it is connected using internet. A Query is called as "Skyline", which query works or execute based on data points. "Skyline" query returns many multidimensional points. It extracts the information from different places of distributed database at different sites. Skyline query returns all the interesting points that are not dominated by any other points. Skyline queries play an important role in multi criteria decision making and the user preference applications. For example, a tourist can issue a skyline query on a hotel relation to get those hotels with high stars and cheap prices. This paper presents the skyline query processing in distributed environment using filtering.*

Keywords: *Data sites, Distributed environment, Filtering, Query.*

I. INTRODUCTION

Developments in the past couple of years have revealed a trend towards distributed data management and the storage systems. In the presence of the huge amounts of data that today's systems are providing access to, it is a tedious task for a user to find the most interesting available data without using the advanced query types, such as skyline queries. Whereas the problem is known as the skylines in database research, in other areas it was already known before as the maximum vector problem or the Pareto optimum [1] [2]. The popularity of the the skyline operator is mainly due to its applicability for decision making applications; skyline queries help users make intelligent decisions over complex data, where different and often conflicting criteria are considered.

Skyline queries have originally been proposed for centralized environments [3], i.e., single-database environments. As now- a- days data is increasingly stored and processed in a distributed way, skyline processing over distributed data has attracted much attention recently. Skyline query processing in the distributed environments poses inherent challenges and requires non-traditional techniques due to the distribution of content and the lack of global knowledge. There are various different distributed systems with a different requirements and unique characteristics that have to be exploited for efficient skyline processing. Peer-to-peer (P2P) systems can be considered as an example of the distributed system architecture for which several distributed skyline approaches have been proposed. Other architectures, such as the Web information systems, parallel shared-nothing architectures, distributed data streams, or wireless sensor networks have different requirements.

The variety of existing distributed systems leads to the variety of existing distributed skyline approaches. Moreover, the fact that several skyline variants, beyond the traditional skyline operator, have also been proposed in the past decade [4] [5] leads to various distributed approaches that support different skyline variants. The most important variants are- subspace skylines (only some attributes of a tuple are considered for evaluation), constrained skylines, and dynamic skyline queries (the skyline is not executed in the original data space but the data points are transformed into another data space before evaluating the skyline). The characteristic of the skyline variants requires sophisticated and specialized algorithms for efficient processing. Depending on the underlying network and the communication architecture, these variants allow for different optimizations.

II. RELATED WORK

A distributed skyline query can be processed by evaluating multiple constrained skyline queries on the different servers. A framework, called SkyPlan [6] has been proposed that maps the dependencies between the queries into a graph and generates cost-aware execution plans. The one of the possible ways to deal with geographically scattered data has been studied using a framework called PadDSkyline [7]. The theme of incomparability for skyline computation has also been explored. The authors have proposed and compared skyline computation based on dominance and incomparability through algorithms BSkyTree-S and BSkyTree-P [8]. The progressive skyline computations using DSL [9] and other algorithms [10] have also been proposed for query load balancing. The advent of the multi-core processors is making a profound impact on software development.

In [11] , the authors have modified the basic skyline computation algorithms SFS (Sort Filter Skyline), BBS (Branch and Bound Skyline) and SSkyline (Simple Skyline) to induce the possible parallelism in them and have proposed a new algorithm PSkyline (Parallel Skyline). In [12], the authors have proposed an algorithm called as SSP (Search Space Partitioning) which exploits features of BATON -Balanced Tree Overlay network for indexing the dataset so that in structured peer to peer network, the peers will be accessed to the minimum and exact data sub space to compute the skyline can be searched efficiently. Other aspects of the parallel skyline computation like rank-aware queries, constrained skyline queries and progressive skyline computation in P2P networks have been proposed in [13] respectively. Proper data sub space partitioning is another aspect of the parallel skyline computation. The related approaches have been discussed in [14].

III. PROPOSED WORK

In our proposed work, given a distributed environment without any overlay structures, our main objective is efficient query processing strategies that shorten the overall query response time. We first speed up the overall query processing by achieving parallelism of the distributed query execution. Given a skyline query with some constraints, all relevant sites are partitioned into incomparable groups among which the query can be executed in parallel. Within each group, specific plans are proposed to further improve the query processing involving all intra- group sites. On a processing site, multiple filtering points are deliberately picked based on their overall dominating potential from the "local skyline". They are then sent to the other sites with the query request, where they help identify more unqualified points that would otherwise be reported as false positives, and thus, reducing the communication cost between data sites.

Filtering points are selected from the local skyline result that initially obtained. Suppose that the initial skyline result is SKinit = {s1; s2; . . . ; sl}, we need to select K (<l) points from it as the "multiple" filtering points. We study two heuristics that guide the selection of K filtering points from l(>k) skyline points. The first heuristic for selecting the multiple filtering points maximizes the sum of the values of all possible choices. To accomplish this, we need to sort points in SKinit in a non- ascending order and then pick the top-K ones. We call this heuristic MaxSum. It actually simplifies the computation by ignoring the overlapping between different "skyline" points dominating regions. The smaller the "overlapping" regions are, the more accurate the method will be. In the second heuristic, we intend to take into account the topology between the filtering points, to reduce the overlapping faced by the first heuristic. "Distance" is a simple metric to help consider this. Intuitively, the farther two "skyline" points are apart, the less their dominating regions overlap. Hence, we propose a greedy heuristic, called "MaxDist", which maximizes the distance between filtering points. The algorithm of this heuristic is shown in Algorithm 1.

Algorithm 1:
Maxdist (SKinit, K)

Input: SK_{init} is the initial skyline;
K is the number of filtering points needed;
Output: a set of multiple filtering points.

Step 1: $F_{flt} = \Phi$

Step 2: Pick S_i and S_j from SK_{init} satisfying $|S_i, S_j| > |S_i^1, S_j^1|$
$\forall$ 1<= S_i^1, S_j^1 <=l;

Step 3: F_{flt} = { S_i, S_j }; $SK^1 = SK_{init}$ -{ S_i, S_j };

Step 4: While $|F_{flt}|$<K do

Step 5: Pick S_i from SK^1 satisfying

$$\sum_{sj \in F_{flt}} |S_i, S_j| >= \sum_{sj \in F_{flt}} |S_i^1, S_j|, \forall\ S_i^1 \in SK^1$$

Step 6: $F_{flt} = F_{flt} \cup \{ S_i \}$; $SK^1 = SK_{init}$ -{ S_i };

Step 7: return F_{flt}

Initially, it picks from "SKinit" two points between which the distance is the largest among all pairs (line 2). Then, it incrementally selects points from "SKinit" and adds them to the filtering set, until K filtering points are obtained. In every incremental step, the point with the maximal sum of the distances to all current filtering points is selected (line 5).

The idea behind "MaxDist" heuristic is good, but it is difficult to implement strictly due to its computational complexity. Therefore, we count the sum of distance between a point and all the current filtering points in MaxDist, and then pick the one with the maximum sum as a new filtering point. The improved version of the basic heuristic "MaxDist", called "MaxDist2", is shown in Algorithm 2.

Algorithm 2:
Maxdist2(SKinit, K, Δ)

Input: SK_{init} is the initial skyline;
K is the number of filtering points needed;
Δ is the distance threshold
Output: a set of multiple filtering points.

Step 1: $F_{flt} = \Phi$

Step 2: Pick S_i and S_j from SK_{init} satisfying $|S_i, S_j| > |S_i^1, S_j^1|$

$\forall$ 1<= S_i^1, S_j^1 <=l;

Step 3: $F_{flt} = \{S_i, S_j\}$; $SK^1 = SK_{init} - \{S_i, S_j\}$;

Step 4: While $|F_{flt}|$<K do

Step 5: Pick S_i from SK^1 satisfying

$$\sum_{sj \in F_{flt}} |S_i, S_j| >= \sum_{sj \in F_{flt}} |S_i^1, S_j|, \forall\ S_i^1 \in SK^1$$

Step 6: and dist(S_i, S_j) > Δ;

Step 7: $F_{flt} = F_{flt} \cup \{S_i\}$; $SK^1 = SK_{init} - \{S_i\}$;

Step 8: return F_{flt}

IV. CONCLUSION

In this paper, we have addressed the problem of constrained skyline query processing in distributed environment. Given a skyline query with some constraints, all relevant sites are partitioned into incomparable groups among which the query can be executed in parallel. Within each group, specific plans are proposed to further improve the query processing involving all intra- group sites. On a processing site, multiple filtering points are deliberately picked based on their overall dominating potential from the "local skyline". They are then sent to the other sites with the query request, where they help identify more unqualified points that would otherwise be reported as false positives, and thus, reducing the communication cost between data sites.

REFERENCES

[1] 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, 1975
[2] F. P. Preparata and M. I. Shamos. Computational Geometry - An Introduction Springer, 1985
[3] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In ICDE, pages 421–432, 2001.
[4] B. Cui, H. Lu, Q. Xu, L. Chen, Y. Dai, and Y. Zhou. Parallel Distributed Processing of Constrained Skyline Queries by Filtering. In ICDE, pages 546–555, 2008.
[5] D. Papadias, Y. Tao, G. Fu, and B. Seeger. An Optimal and Progressive Algorithm for Skyline Queries In SIGMOD, pages 467–478, 2003.
[6] João B. Rocha-Junior, Akrivi Vlachou, Christos Doulkeridis, and Kjetil Nørvåg, "Efficient Execution Plans for Distributed Skyline Query Processing", Conf on EDBT, March 2011.
[7] Lijiang Chen, Bin Cui, Hua Lu, "Constrained Skyline Query Processing against Distributed Data Sites" IEEE TKDE, Vol. No 23 no 2, February 2011, pp. 204-217.
[8] Jngwuk Lee, Seung-won Hwang. "BskyTree: Scalable Skyline Computation Using A Balanced Pivot Selection.", EDBT 2010, March 22-26, Lausann, Switzerland, 2010, pp. 195-206.
[9] Ping Wu, Caijie Zhang, Ying Feng, Ben Y. Zhao, Divyakant Agrawal,, and Amr El Abbadi "Parallelizing Skyline Queries for Scalable Distribution", EDBT 2006, LNCS 3896, pp. 112–130.
[10] Lin Zhu, Shuigeng Zhou, Jihong Guan, "Efficient Skyline Retrieval on Peer-to-Peer Networks", Future Generation Communication and Networking (FGCN), 2007, pp. 309-314.
[11] Hyeonseung Im, Jonghyun Park, Sungwoo Park, "Parallel skyline computation on multi-core architectures", IEEE Int‟ l Conf. on Data Engineering (ICDE), 2009, pp. 808-823.
[12] Shiyuan Wang, Beng chin Ooi, Anthony Tung, Lizhen Xu, "Efficient Skyline Processing on Peer to Peer Networks", IEEE Int‟ l Conf. Data Eng. (ICDE), 2007.
[13] K. Hose, M. Karnstedt, A. Koch, K. Sattler, and D. Zinn, "Processing rank-aware queries in P2P systems" In Proceedings of DBISP2P, 205, pp. 238–249
[14] Akrivi Vlachou, Christos Doulkeridis, Yannis Kotidis, Michalis Vazirgiannis, "SKYPEER: Efficient Subspace Skyline Computation over Distributed Data",. IEEE Int‟ l Conf. Data Engineering (ICDE) 2007, pp. 416-425.