Efficient Processing of Skyline Queries Using MapReduce Considering Medical field as Case Study

[1] Chethana C R, [2] Dr.Mohamed Rafi
[1] PG Student, [2] Professor,
[1,2] Department Of Studies in Computer Science & Engineering
[1,2] University BDT College Of Engineering (A Constituent College of VTU Belagavi), Davanagere,Karnataka,India,

Abstract: The skyline operator has attracted considerable attention recently due to its broad applications. However, computing a skyline is challenging today since we have to deal with big data. For data-intensive applications, the MapReduce framework has been widely used recently. In this paper, we propose the efficient parallel algorithm SKY-MR+ for processing skyline queries using MapReduce. We first build a quadtree-based histogram for space partitioning by deciding whether to split each leaf node judiciously based on the benefit of splitting in terms of the estimated execution time. In addition, we apply the dominance power filtering method toeffectively prune non-skyline points in advance. We next partition data based on the regions divided by the quadtree and compute candidate skyline points for each partition using MapReduce. Finally, we check whether each skyline candidate point is actually a skyline point in every partition using MapReduce. We also develop the workload balancing methods to make the estimated execution times of all available machines to be similar.

Key words: Data Mining MapReduce Prediction SKY-MR

I.INTRODUCTION

The skyline operator and its variants have attracted considerable attention recently due to their broad applications such as product recommendations, review evaluations with user ratings, querying wireless sensor networks and graph analysis. However, computing a skyline is challenging today since we have to deal with big data. For data-intensive applications including similarity joins and top-k substring matching, the MapReduce framework has been considered as a de facto standard. Thus, several skyline processing algorithms using MapReduce have been proposed.

MR-GPMRS in consists of the partitioning and global skyline phases. The partitioning phase of MR-GPMRS divides the data space into grid partitions and prunes the partitions that cannot contain any skyline point by utilizing the dominance relationships between grid partitions. In the global skyline phase, in every unpruned partition P, the points which are located in other unpruned partitions and may dominate a point in P are first collected and each point in the partition P is compared with the collected points to determine whether it is a global skyline point in parallel. To compute the skyline efficiently, an additional localskyline phase is involved between the partitioning and global skyline phases in MR-BNL, PPF-PGP and SKY-MR. They compute the local skyline for each partition and use them to compute the skyline in the global skyline phase. The benefit of the additional phase is that the overheads of computing the skyline as well as distributing the points via the network in the global skyline phase areReduced since the number of local skyline points in eachpartition is much less than that of all points in the partition.

II. Existing System

Computing a skyline is challenging today since we have todeal with big data. For data-intensive applications including similarity joins and top-k substring matching, the MapReduce framework has been considered as a de facto standard. Thus, several skyline processing algorithms using MapReduce have been proposed. MR-GPMRS in consists of the partitioning and global skyline phases. The partitioning phase of MR-GPMRS divides the data space into grid partitions and prunes the partitions that cannot contain any skyline point by utilizing the dominance relationships between grid partitions. In the global skyline phase, in every unpruned partition P, the points which are located in other unpruned partitions and may dominate a point in P are first collected and each point in the partition P is compared with the collected points to determine whether it is a global skyline point in parallel.

Drawbacks of SKY-MR

SKY-MR builds a sky-quadtree from a sample of data based on the user-defined parameter split threshold which is the maximum number of points in each leaf node. As the split threshold decreases, the number of leaf nodes in the quadtree tends to increase and more points are allowed to be pruned by the dominance relationships between leaf nodes in the local skyline phase. In contrast,

decreasing split threshold has an adverse effect on the network overhead by transmitting more duplicates of local skyline points to other leaf nodes in the global skyline phase. Since there is a trade-off between the costs of the local and global skyline phases, when a reasonable split threshold is not provided, its performance suffers. Furthermore, since SKY-MR as well as the other MapReduce skyline algorithmsdo not consider workload balancing, the performances of the algorithms could degrade. Finally, there is still a lot of room for improvement to reduce the communication and computation costs of the local skyline phase.

III.Proposed System

The MapReduce Framework

MapReduce or its open-source equivalent Hadoopis a widely used framework for data-intensive parallel computation in shared-nothing clusters of machines. In Hadoop, data is represented as key-value pairs. Hadoop divides the input data to a MapReduce job into fixed-size pieces called chunks and spawns a mapper task for each chunk. The mapper task invokes a map function for each key-value pair in the chunk and the map function may output several key-value pairs. The key-value pairs emitted by all map functions are grouped by keys in the shuffling phase andpassed to reducer tasks to generate the final output.Userscancontrol which key goes to which reducer task by modifying a Partitioner class. For each distinct key, the reduce task invokes a reduce function with the key and the list of all values sharing the key as input. A reduce function may generate several key-value pairs.
Each mapper (or reducer) task can execute a setup function before invoking map (or reduce) functions and a cleanup function after executing all map (or reduce) functions. Hadoop executes the main function on a single master machine

SKY-MR: The state-of-the-art Algorithm

(1) **Sky-quadtree building phase:** SKY-MR builds a skyquadtreewith a sample of data to split the data space into several partitions. The d-dimensional data space is subdivided recursively into 2d equi-sized sub-regions each of which is associated with a node of the sky-quadtree until each sub-region contains at most a predefined number of points called the split threshold. According to the dominance relationships between the regions represented by nodes, every node without any skyline point is marked as “pruned”.

(2) **Local skyline phase:** For each unpruned leaf node n of the sky-quadtree, the local skyline of P(n), denoted by SL(P(n)), is computed where P(n) is all points in the region represented by n. To reduce the number of checking dominance relationships between points in the next phase, virtual max points and sky-filter points are computed after the local skylines are obtained. The virtual max point vpn of a leaf node n is an artificial d-dimensional point such that vpn(k)= maxp2SL(P(n))p(k) with 1_k_d. In each leaf node n, we also select a single local skyline point, called a sky-filter point, which has the minimum value for every dimension.

(3) **Global skyline phase:** Each local skyline point in every unpruned leaf node is checked whether it is a global skyline point or not by comparing it with the local skyline points in the region of the other unpruned leaf nodes. When the total number of local skyline points is less than the size threshold _, a single machine is used to speed up. SKY-MR first collects all virtual

max and sky-filter points of every leaf node. If a local skyline point p located in the region of a leaf node is dominated by any sky-filter point, p is discarded without comparing to the local skyline points of the other unpruned leaf nodes. The number of checking dominance relationships between a pair of points can be even more reduced by utilizing the virtual max points. For an unpruned leaf node n, it is shown in [14] that if a local skyline point p0 in another leaf node n0 does not dominate the virtual max point of the leaf node n (i.e., p0 6_ vpn), the point p0 does not dominate every local skyline point in SL(P(n)). Thus, in every unpruned leaf node n, each local skyline point p in the region of n becomes a skyline point if p is not dominated by every local skyline point p0 which dominates vpn and is in the region of the other unpruned leaf nodes.

Related Works

After skyline processing was introduced in [1], several serial algorithms for computing skylines and its variants were introduced in [2], [3], [18], [19], [20], [21], [22], [23], [24], [25]. However, existing serial skyline algorithms utilizing centralized indexing structures such as B+-trees and R-trees are not suitable to be parallelized using MapReduce since the MapReduce framework does not provide the functionality for building and accessing centralized indexing structures. Although we focus on computing the skyline using MapReduce, we still need a serial skyline algorithm to calculate the local skyline for each partition. Thus, among the serial skyline
algorithms [1], [18], [24], [25] without using centralized indexes, we adopt the state-of-the-art algorithm BSkyTree-P [24]. To split the data space into 2d partitions, BSkyTree-P first selects a pivot point. Then, every point dominated by the pivot point is removed and BSkyTree-P recursively divides the partitions into sub-partitions until each partition contains at most one point. It next merges the partitions and computes the local skyline of the merged partition repeatedly until there is a single partition and then the global skyline is obtained.

Recently, skyline processing algorithms in distributed environments such as MapReduce [11], [12], [13], [14], sensor networks [6] and other distributed systems [26], [27], [28], [29] have been proposed. The MapReduce algorithms for probabilistic skyline queries [16], [30] and subspace skyline queries [31] are also proposed. Among the above works, we next illustrate MR-GPMRS [11], MR-BNL [12], PPF-PGPS [13] and SKY-MR [14] briefly since they are the most relevant works to ours. The details of the state-of-theart algorithm SKY-MR. While MR-GPMRS [11] consists of the partitioning and global skyline phases only, MR-BNL [12], PPF-PGPS [13] and SKY-MR [14] are composed of the partitioning, local skyline and global skyline phases. In the partitioning phase, the space is split into partitions by using sky-quadtree partitioning in SKY-MR, angle-based partitioning [32] in PPFPGPS or grid partitioning in MR-GPMRS and MR-BNL. In contrast to MR-GPMRS using two phases, MR-BNL, PPFPGPS and SKY-MR compute the local skyline for each partition in the additional localskyline phase. Then, in the global skyline phase, MR-GPMRS, MR-BNL, PPF-PGPS and SKY-MR compute the global skyline. Since MR-BNL uses only up to 2d machines in the local skyline phase where d is the number of dimensions, the machines participating in the MapReduce framework could not be fully utilized. In addition, since a single machine computes the global skyline, MR-BNL is inefficient when a large number of local skyline points are produced. On the contrary, SKY-MR utilizes all available machines at the local and global skyline phases. Furthermore, in the local skyline phase, SKY-MR performs additional pruning by utilizing the dominance relationships between partitions. It is shown in [14] that SKY-MR outperforms MR-BNL. Since PPF-PGPS uses a single machine to compute the global skyline, SKYMR also shows better performance than PPF-PGPS. In addition, SKY-MR is generally more efficient than MR-GPMRS since MR-GPMRS does not have the local skyline phase. Although the works in [27] and [28] are not proposed for MapReduce, since they can be processed with MapReduce, we present them here.

The 1-step and 2-step algorithms in [27] split the data space into grid partitions. They next prune the partitions with no skyline point and compute the global skyline for every unpruned partition in parallel. While both algorithms in [27] split the data space into a fixed number of grids, SKY-MR varies the number of partitions adaptively based on the data distribution.

The algorithm PPPS in [28] for multi-core machines utilizes the angle-based space partitioning [32]. PPPS recursively splits each partition into two partitions until the number of the partitions becomes the desired number of CPU cores c. The local skyline is next computed for every partition in parallel. Finally, PPPS performs a bottom-up merge in O(log(c)) iterations until there remains a single partition only. Since PPPS can utilize c=2i cores only in the i-th merging iteration, multi-cores are not fully utilized. However, SKY-MR computes the global skyline by considering each partition independently and utilizing all available machines simultaneously. As expected, it is shown in [14] that SKY-MR is superior to the MapReduce implementations of the algorithms in [27] and [28].

[1] S. B¨orzs¨onyi, D. Kossmann, and K. Stocker, "The skyline operator," in ICDE, 2001, pp. 421–430.
They propose to extend database systems by a Skyline operation. This operation $filters out a set of interesting points from a potentially large set of data points. A point is interesting if it is not dominated by any other point. For example, a hotel might be interesting for somebody traveling to Nassau if no other hotel is both cheaper and closer to the beach. We show how SQL can be extended to pose Skyline queries, present and evaluate alternative algorithms to implement the Skyline operation, and show how this operation can be combined with other database operations, e.g., join.

[2] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimal and progressive algorithm for skyline queries," in SIGMOD, 2003.
The skyline of a set of d-dimensional points contains the points that are not dominated by any other point on all dimensions. Skyline computation has recently received considerable attention in the database community, especially for progressive (or online) algorithms that can quickly return the first skyline points without having to read the entire data file. Currently, the most efficient algorithm is NN (nearest neighbors), which applies the divideand-conquer framework on datasets indexed by R-trees. Although NN has some desirable features (such as high speed for returning the initial skyline points, applicability to arbitrary data distributions and dimensions), it also presents several inherent disadvantages (need for duplicate elimination if d>2, multiple accesses of the same node, large space overhead). In this paper we develop BBS (branch-and-bound skyline), a progressive algorithm also based on nearest neighbor search, which is IO optimal, i.e., it performs a single access only to those R-tree nodes that may contain skyline points. Furthermore, it does not retrieve duplicates and its space overhead is significantly smaller than that of NN. Finally, BBS is simple to implement and can be efficiently applied to a variety of alternative skyline queries. An analytical and experimental comparison shows that BBS outperforms NN (usually by orders of magnitude) under all problem instances.

[3] E. Dellis and B. Seeger, "Efficient computation of reverse skyline queries," in VLDB, 2007, pp. 291–302.
In this paper, for the first time, we introduce the concept of Reverse Skyline Queries. At first, we consider for a multidimensional data set P the problem of dynamic skyline queries according to a query point q. This kind of dynamic skyline corresponds to the skyline of a transformed data space where point q becomes the origin and all points of P are represented by their distance vector to q. The reverse skyline query returns the objects whose dynamic skyline contains the query object q. In order to compute the reverse skyline of an arbitrary query point, we first propose a Branch and Bound algorithm (called BBRS), which is an improved customization of the original BBS algorithm. Furthermore, we identify a super set of the reverse skyline that is used to bound the search space while computing the reverse skyline. To further reduce the computational cost of determining if a point belongs to

the reverse skyline, we propose an enhanced algorithm (called RSSA) that is based on accurate pre-computed approximations of the skylines. These approximations are used to identify whether a point belongs to the reverse skyline or not. Through extensive experiments with both real-world and synthetic datasets, we show that our algorithms can efficiently support reverse skyline queries. Our enhanced approach improves reversed skyline processing by up to an order of magnitude compared to the algorithm without the usage of pre-computed approximations.

[4] J. Lee, S. won Hwang, Z. Nie, and J.-R. Wen, "Navigation system for product search," in ICDE, 2010.
We demonstrate Product EntityCube, a product recommendation and navigation system. While the unprecedented scale of a product search portal enables to satisfy users with diverse needs, this scale also complicates product recommendation. Specifically, our target application poses a unique challenge of overcoming insufficient user profiles and feedbacks. To address this problem, we organize query results into clusters representing different user perceptions of similarity, and provide a navigational UI to handle personal interests. Specifically, we first discuss hybrid object clustering capturing diverse user perception from millions of Web pages and disambiguating different senses using feature-based similarity. We then discuss skyline object ranking to highlight interesting items at each cluster. Our demonstration illustrates how Product EntityCube can enrich user product shopping experiences.

[5] T. Lappas and D. Gunopulos, "Efficient confident search in large review corpora," in ECML/PKDD (2), 2010.
Given an extensive corpus of reviews on an item, a potential customer goes through the expressed opinions and collects information, in order to form an educated
opinion and, ultimately, make a purchase decision. This task is often hindered by false reviews, that fail to capture the true quality of the item's attributes. These reviews may be based on insufficient information or may even be fraudulent, submitted to manipulate the item's reputation. In this paper, we formalize the Confident Search paradigm for review corpora. We then present a complete search framework which, given a set of item attributes, is able to efficiently search through a large corpus and select a compact set of high-quality reviews that accurately captures the overall consensus of the reviewers on the specified attributes.We also introduce CREST (Confident REview Search Tool), a user-friendly implementation of our framework and a valuable tool for any person dealing with large review corpora. The efficacy of our framework is demonstrated through a rigorous experimental evaluation.

IV.CONCLUSION

The parallel skyline computation using MapReduce and develop the algorithm SKY-MR+. We first build a sky-qtree+ with an adaptive quadtree building technique to utilize the dominance relationships between regions and apply the dominance power filtering method to effectively prune out non-skyline points in advance. SKY-MR+ partitions the data based on the regions split by the sky-qtree+ and computes the candidate skyline points independentlyfor each partition. Finally, we check whether each skylinecandidate point is actually a skyline point in every partition independently. To make the estimated execution times of all available machines to be similar, we develop workload balancing techniques.

REFERENCES

[1] S. B¨orzs¨onyi, D. Kossmann, and K. Stocker, "The skyline operator,"in ICDE, 2001, pp. 421–430.
[2] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimal and progressive algorithm for skyline queries," in SIGMOD, 2003.
[3] E. Dellis and B. Seeger, "Efficient computation of reverse skyline queries," in VLDB, 2007, pp. 291–302.
[4] J. Lee and S.-w. Hwang, "Scalable skyline computation using abalanced pivot selection technique," Information Systems, vol. 39, pp. 1–21, 2014.
[5] F. N. Afrati, P. Koutris, D. Suciu, and J. D. Ullman, "Parallel skylinequeries," in ICDT, 2012, pp. 274–284.
[6] H. K¨ohler, J. Yang, and X. Zhou, "Efficient parallel skyline processing using hyperplaneprojections," in SIGMOD, 2011, pp. 85–96.