Efficient Processing of Skyline Queries Using MapReduce

ABSTRACT

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.

EXISTING SYSTEM

Skyline processing was introduced in several serial algorithms for computing skylines and its variants were introduced. 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 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.

DRAWBACKS

- Skyline algorithms utilizing centralized indexing structures such as B+-trees and R_-trees are not suitable to be parallelized using MapReduce.
- MapReduce framework does not provide the functionality for building and accessing centralized indexing structures.

PROPOSED SYSTEM

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 to effectively 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. We did experiments to compare SKY-MR+ with the state-of-the-art algorithms using MapReduce and confirmed the effectiveness as well as the scalability of SKY-MR+.

ADVANTAGES

- The dominance power filtering method to effectively prune non-skyline points in advance.
- The workload balancing methods to make the estimated execution times of all available machines to be similar.
- We check whether each skyline candidate point is actually a skyline point in every partition using MapReduce.

SYSTEM REQUIREMENTS

- **H/W System Configuration:-**

- Processor - Pentium –IV
- RAM - 4 GB (min)
- Hard Disk - 20 GB
- Key Board - Standard Windows Keyboard
- Mouse - Two or Three Button Mouse
- Monitor - SVGA

- **S/W System Configuration:-**
- Operating System : Windows 7 or 8 32 bit
- Application Server : Tomcat5.0/6.X
- Backend coding : Java
- Tool : Virtual Box
- Environment : Ubuntu
- Technology : Hadoop