CS615: Group 05
K-Dominant Skyline in Distributed System

Swapnil Mhamane
32
15111044
mswapnil@iitk.ac.in
Dept. of CSE

Ritika
31
15111036
ritigarg@iitk.ac.in
Dept. of CSE

Indian Institute of Technology, Kanpur

Final report

$13^{th} November, 2015$

Abstract

K dominant skylines are objects which are not dominated in more than k dimension by any other object in the interested set of dimensions. Finding such k dominant skyline over horizontally partitioned distributed data haven't acquired adequate attention. But it is quite natural requirement to get k attributed filtered data object in today's cloud based business application having their data distributed geographically. In such scenarios it costs a huge network overhead in bringing up entire data to query source site and then filtering entire data set as per user interest. For finding such k-dominant skyline we propose algorithm based on one-scan algorithm [1] with forest relation among the site in peer-to-peer network for execution of query, so as to reduce the network data traffic with some trade off with response time cost.

1 Introduction

"Given a d-dimensional data set, a point p dominates another point q if it is better than or equal to q in all dimensions and better than q in at least one dimension. A point is a skyline point if there does not exists any point that can dominate it. Skyline queries, which return skyline points, are useful in many decision making applications. Unfortunately, as the number of dimensions increases, the chance of one point dominating another point is very low. As such, the number of skyline points become too numerous to offer any interesting insights. To find more important and meaningful skyline points in high dimensional space, a concept , called k-dominant skyline had been proposed which relaxes the idea of dominance to k-dominance. A point p is said to k-dominate another point q if there are k ($\leq d$) dimensions in which p is better than or equal to q and is better in at least one of these k dimensions. A point that is not k-dominated by any other points is in the k-dominant skyline." [1] k-dominant Skyline queries have been studied in centralized. However, the execution of k-dominant skyline queries on different (potentially overlapping) data fractions, in order to obtain the k-dominant skyline set of the entire data set efficiently, has not received adequate attention.

1.1 Problem Statement

Our objective is to compute k-dominant skylines in large-scale Distributed Environment such as peer to peer system in which every site is connected to every other site. In such a setting, each server stores autonomously a fraction of the data(i.e. horizontal partitioning of data), thus all servers need to process the k-dominant skyline query. To find the k-dominant skyline, we introduce a novel framework, called kDomSkyAlgo, for processing distributed k-dominant skyline queries that returns the desired result.

1.2 System Overview

The underlying system is assumed to have a peer to peer network such that each site is connected to every other site in the network.Client can fire query at any site.

1.3 Motivation

Why the k-dominant skyline have gained so much importance in the multi-criteria decision making applications. For justifying this question , we will present some examples where , it is worth to compute the k-dominant skyline.

1. Suppose you are going to have a round trip around the world. In your trip you want to visit the popular places according to some criteria. We can clearly see that this system will have data to be distributed among the different sites. According to the preferences of the user we want to find some interesting locations or skyline points.

2. Suppose you are about to buy a product (say car) and suppose the database for storing the products information is distributed. Now, there could be a large number of attributes (like price ,color, mileage, green core) over which that product is being defined. There are too many cars at the website for the user to examine them all manually. So rather, You could select some or all of the attributes as your preference list. Computing the skyline over these cell phone features may remove a large number of cars whose features are worse than those in the skyline, hopefully leaving only a manageable number of candidates for manual evaluation.

The above examples illustrates that how useful it could be to compute the k-dominant skylines over a set of distributed databases.

The remaining of this report is organized as follows: Section 1.4 overviews the related work. Then, we present the necessary preliminaries in Section 2.1 and basic steps involved in Section 2.2. In sections 2.3,2.4,2.5 we present the first,second and third concrete algorithms.Then we present the evaluations and results in section 3.

1.4 Related Work

The basic idea of skyline by presented by Borzonyi et al, who proposed the very first algorithm to compute the skyline from a given dataset. This approach was extended to a concept called k-dominant skyline computation by Chee-Yong Chan [1] and his team. As the k-dominant skylines may be cyclic and non-transitive ,thus the already existing algorithm cant be applied to obtain the desired result. So They (Chee-Yong Chan and his team) proposed three new algorithms to get the k-dominant skylines. One scan algorithm [1] basically sequentially scans the complete data set once and uses the property that each nonk dominant skyline will be dominated by at least one full skyline. Two scan algorithm [1]firstly figures out a list of candidate set for the k-dominant skyline and then takes one more pass in the candidate set so as to remove the non-k dominant skyline. Sorted retrieval [1]which is based on the aggregation based skyline. The skyline computation in distributed environment was studied by Joo B. Rocha-Junior and his team [2]. This paper was basically organized to get a execution plan so as to obtain a balance between the latency time and the network overhead.
We are basically combining the k-dominant skyline computation with the distributed environment. We are extending the one scan algorithm to compute the k-dominant skyline at each site and the basic framework of the algorithm [2] proposed by Rocha.

2 Algorithm

The naive approach to solve the given problem is just collect the data from all other sites in the network and send it to originator. The originator will then merge the complete database and then apply any algorithm for finding the k-dominant skylines(one-scan in our case). If the size of datasets stored at different sites is small and the number of sites are also less then this may be a good approach but generally this is not the case. We have a huge amount of data and a large number of sites at our pool, hence flushing whole data over the network may even cause network overflow.
An improvement over the naive approach could be based on the answer to the question" why to send complete dataset of each site ? ".We rather send only the k-dominant skylines of each site and then the originator will merge the k-dominant skyline set of individual site to get the complete result. But this method have a drawback that we might not get exactly the complete k-dominant skyline set but some extra things may also be included (i.e. false positives). Because of non-transitive property of the k-dominant skylines. For finding the k-dominant skyline set it is enough to compare each tuple against the full skyline set [1]. So now we can use this result to find the k-dominant skyline set. Now instead of sending only the k-dominant skyline, the sites forward their full skyline set (i.e. (k-dominant skyline set) U (non-k dominant full skyline set)). It may be a case when large number of skylines which are not the part of final k-dominant result are

also forwarded to the originator but sending them to was a waste.So as to prevent this wastage, we are sending a filter point from the originator to all the sites in the network .Based on the filter point obtained each site will prune some points and then each site will compute its full skyline set and send it to the originator. The originator will then merge the results of different sites to get the complete result set.

2.1 Preliminaries

Given a dataset D on a data space defined by a set of d dimensions $\{d_1, ..., d_d\}$, a point p $\in$ D is represented as p=$\{p_1, ..., p_d\}$ where p_i is the value on dimension d_i. Without loss of generality, we assume that $\forall d_i : p_i \geq 0$, and that smaller values are preferable.

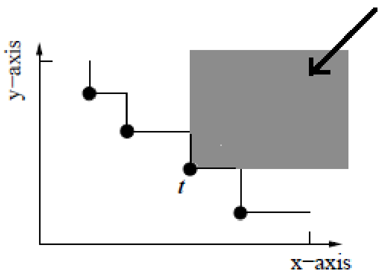

Figure 1: Dominance area of any point(t)

Let us consider m sites denoted by S_1,S_2,S_3......S_m connected as peer to peer system.The dataset at site S_i is denoted by the MBR $m_i(l_i, u_i)$; where l_i and u_i denote the lower bound,upper bound of site S_i respectively. Here l_i,u_i don't need to be the actual points in the site but defines the boundary of its data points.

In this section, we are defining the terms which will be used to get hold over the proposed algorithms.
The dominance area of any point is defined as:
The area under which all the points are being dominated by it i.e. The area that is enclosed by the hyper-rectangle that has as lower left corner the point p and as upper right corner the maximum corner of the universe.In Figure: 1 the dominance area for t[a 2-dimensional point] is shown.

The dominance area of any n-dimensional point t(t_i ;denote the value of t at i^{th} dimension and i varies from 1 to n) is given by:
Dominance area = π(u_i - t_i) :: $\forall$ i=1-n
where u_i is the upper bound till where the data could be there on the database. Given a hyper-rectangle m_i, we introduce the notion of enclosed dominance area of a point t, as the area within the hyper-rectangle mi that is dominated by p. Similarly, given two hyper-rectangles $m_i(l_i, u_i)$and $m_j(l_j, u_j)$, the enclosed dominance area of m_i on m_j is the volume of m_j that is dominated by the lower left corner l_i of m_i.
We define the relations between the datasets of two different sites so as to form a forest like structure , in which each edge will show some sort of dependency so that computing skylines in one site leads to pruning some of the data points of the other site.
For forest construction establish the relations between two sites. The relations[dependencies] among two sites S_i and S_j are defined as:

1. Site S_i is said to fully dominate site S_j if $u_i \succ l_j$.

2. Site S_i is said to partially dominate the site S_j if $u_i \succ u_j$ and $l_j \not\succ l_i$.

3. If none of the above two cases occurs then sites are said to be incomparable.

The dependency relations are the basis of our proposed algorithms. Given three approaches the basic algorithm remains the same , but differs by filter points , when the skyline points are calculated and on what criteria the tree is formed. Based on the relations between the sites we are going to create a forest of sites present in the network. Then make each forest as a child of the originator. In this forest structure, we will exploit the dependencies between the different sites. The following are the cases:

- If any site(say S_i) is fully dominating another site S_j then we can prune the S_j from any further computation.

- If any site (say S_i) is partially dominating another site (say S_j). Then the points that are being dominated by the u_i in S_j can be removed from the further computation.Hence, S_i will be taken as it's parent and pruning in S_j will be done on the basis of filter points from its parent S_i.It may be possible that any site (say S_k) is being partially dominated by two or more sites then we define different heuristics for approach 1 and 3.This heuristic is taken as maximum dominance area in approach 1,2 and maximum enclosed dominance area in approach 3.

- If two sites are incomparable then we will make separate trees for each site.

Classify each node according to above mentioned criteria and hence we will get the desired forest structure for further processing.

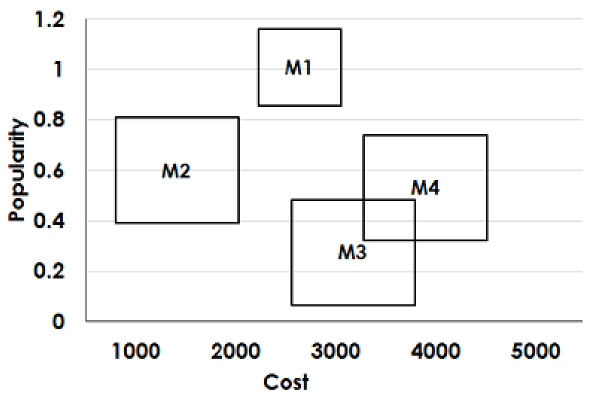

Figure 2: Dominance relation between MBR

In the figure 2, m_2 fully dominate m_1 and hence no point in m_1 can be in the k-dominant skyline and hence m_1 could be pruned off from the computation. m_3 partially dominates m_4 hence the points which lies strictly above the upper bound (u_3) of m_3 can be pruned off.Thus in general it would be beneficial to first compute the skylines in m_3 then in m_4. Hence , we make m_3 as parent of m_4 during the forest construction. If more than one MBR partially dominate another MBR then we will consider the MBR with maximum dominance area in approach 1 and 2(maximum enclosed dominance area in approach 3).

2.2 Basic algorithm

The basic steps involved in the both the approaches are:

1. Collect the summarized information (i.e. MBR) of data of each server to the originator.
2. The originator based on the information obtained will compute a forest like structure.The forest structure we are going to get will have the following basic structure:
   Originator of the query as the root node, this level will be named as level 0. The next level (i.e. level 1) will have incomparable sites .The criteria for building this forest like structure is specific to the algorithm.
3. Now , these parent child relations will be pushed to each server site.
4. After the forest formation phase, the originator and the all other sites will compute their local full skyline ((k-dominant skyline) U (non-k dominant full skyline)). The order of computation of the skyline set (i.e. child computing sky set parallely with the parent or waiting for their parents to complete the processing) is specific to approach.
5. The child sites will get some filter point from its parent and will prune according to that point before or after computation of the skyline set (depends on the approach).
6. After this process, the sites at the leaf of the forest will send its full skyline set ((k-dominant skyline) U (non k-dominant skyline set)) to its parent.Parent will merge its own result set with the result set obtained from the child. These nodes will then forward the merged set to its own parent.This process will continue till we finally hit the originator site in the forest.

For computing the k-dominant skyline at each site we are using the one-scan algorithm [1].

Algorithm 1 kDomSkyAlgo

```
Input: Database $D$, Query $Q$
Output: K-Dominant dataset kdom
1: for all sites do $m[i] \leftarrow RequestMBR()$;
2: $T[] \leftarrow TreeFormation(sitedetails, m[i])$;
3: Push each tree $\in$ T to its root site
4: kdom= kdomskyfinder( )
5: return $k - dominantdataset$
```

In step 1 of the algorithm 1, we use RequestMBR() function to compute the MBRs of each site and return to the originator. The originator server then forms the tree based on the dominance relation between different MBR. As mentioned earlier, some MBR may get fully pruned in this process. Then we forward the tree to respective to root node further find the k-dom skyline amongst node within that tree.

Algorithm 2 TreeFormation

```
Input: MBR $m[i]$, $sitedetails$
Output: forest $F$
1: For all sites $S_i$
2: For all sites $S_j$
3: If $S_j$ completely dominates $S_i$ then remove $S_i$ from the list.
4: Else
5: Find $S_j$ which maximizes the dominance area of $S_j$ under Si
6: Assign that $S_j$ to be parent of $S_i$.
```

In step 2 of algorithm 1, we call algorithm 2, We are forming the tree based on the dependencies. If any site

S_i is completely dominated by any other site S_j then the site S_i will be completely removed from our query computation else we find a site S_k which maximizes the partially dominance area under the site S_i and we will make S_i as parent of S_k. because if we have the k-dominant and full skylines of S_k then that can prune some of the points in the S_i. Continuing this way we will get our tree.
In step 3 of algorithm 1,we are pushing the tree to each site in forest. Because some of the sites may get pruned completely.
In step 4 of algorithm 1,it calls 3. In algorithm 3,we start from the parent nodes and goes till we encounter the leaf node, compute the k-dominant skylines and the full skylines in their local proximity. Now, this result will be returned to the parent. The parent node will compute its k-dominant skylines and the full skylines along with merging the results from its children. This process will continue till we reach the root or the originator of the query.
In algorithm 4 [1], we are applying the conventional one scan algorithm but with a slight modification. We here are initializing the T set to contain the probale non-k-dominant full skylines from the child sites and input will be probable k-dominants collected from descendents in tree, so as to avoid the separate merging of the results from the child sites into the parent site.

Algorithm 3 kdomskyfinder

```
Input: Query $Q$, Forest $F = T1, T2..$,filter
Output: $kdom$,$nonkdomfullsky$
1: If $F$ is empty
2: $(localkdom, localnonkdomfullsky) \leftarrow$
3: localoneScan(filter)
4: Else $Localnonkdomfullsky = localoneScan(filter)$;
5: $remoteFilter = localMBR.getUpperBound()$;
6: For each tree T in forest F
7: (T.kdom , T.nonk-domfullsky) =kdomskyfinder(Q, T.getSubtrees,remoteFilter);
8: Probablekdom= Probablekdom U T.kdom U localK-dom;
9: Probablenonkdom= Probablenonkdomfullsky U T.nonkdomfullsky U localnonkdomfullsky;
10: $Kdom, fullsky \leftarrow oneScanForNonLeaf()$
11: return $kdom, fullsky$
```

Algorithm 4 oneScanForNonLeaf

```
Input: $inputdata$,$T$
Output: Result set $k - dom$, $full - sky$
    run conventional onescan algo for k-dom
    return k-dom, full-sky.
```

Algorithm 5 localoneScan

```
Input: Database $D$, Query $Q$
Output: Result set $A$
    initialize T=$\Phi$ and R=$\Phi$ and inputdata=localdataset
    run conventional onescan algo for k-dom
    return $k - dom$,$full - sky$
```

In algorithm 5 [1], we are just executing the conventional one scan algorithm for finding the k-dominant skylines and full skylines in the leaf nodes or sites in the tree. Finally in step 5 of 1, we return the final results.

2.3 Approach 1

As we have mentioned earlier, the forest construction, filter points and the order of computation of skylines at each site is specific to the algorithm proposed.
Here, each site will compute skylines parallely and the heuristic for constructing the tree is dominance area. In order to avoid the situation where all the sites gets mapped to a single site we are defining the bound over each site, so that it can have that many child sites. We call this bound as the parallelism degree.
The filter points here are the upper bound of the parent node. Based upon the filter point obtained from the parent, the child site will prune some of its skyline (because each child have already computed skyline parallely with child).This will reduce the network traffic.

2.4 Approach 2

Approach 2 differs from Approach 1 by two points:

- Order of the computation of the skylines at each site.
- Filter point.

Here the filter point is the skyline point of the parent node which have the maximum dominance area and the order of the skyline computation is fixed by the parent child relation in the forest.Child will wait for the parent to compute its skyline and after receiving the filter point from the parent node ,the child node will prune points from its dataset and then will compute its skyline set and it will forward a filter point[may be updated ,as we again will compute the maximum dominance area of each tuple on this site] to its child.This process will continue till we hit the leaf.

2.5 Approach 3

This approach differs from Approach 2 by:

- Criteria over which the tree is constructed.
- No bound for parallelism is used.

Here the criteria for building the tree will be the maximum enclosed dominance area. Rest of the algorithm is same as for approach 2.

3 Evaluation and Results

3.1 Implementation

We are proposing the solution to the k-dominant skylines in the distributed environment. We have given our implementation in java. We are using multi-threaded TCP connection to set up communication in the distributed environment. For parallel execution, we are using the multi-threading service of java. We evaluated the results by creating multiple processes on a different machine where each of these processes behaves as individual server. We distribute the data among these processes and then perform the steps as mentioned in the algorithmic section. We are testing our proposed algorithms over database of size one million. This database is splitted among 25 different site environment simulated using multi-threaded aspect of java.

3.2 Comparison among approaches

We implemented the following approaches for evaluation purpose.

1. naive system
2. modified naive system
3. approach 1
4. approach 2
5. approach 3

Based on the result produced from the above mentioned approaches for similar distributed environment setup and different input data set type, we mapped the results as follows

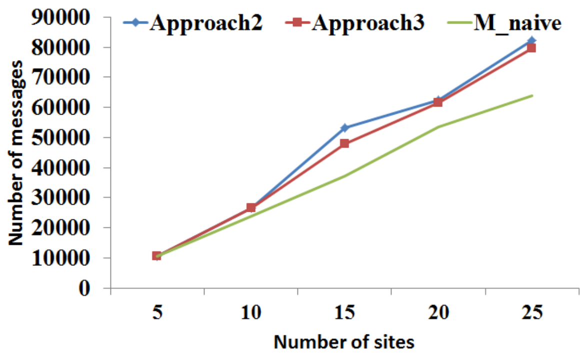

Figure 3: Result for anti-correlated data with varying number of sites[data also varying according to number of sites]

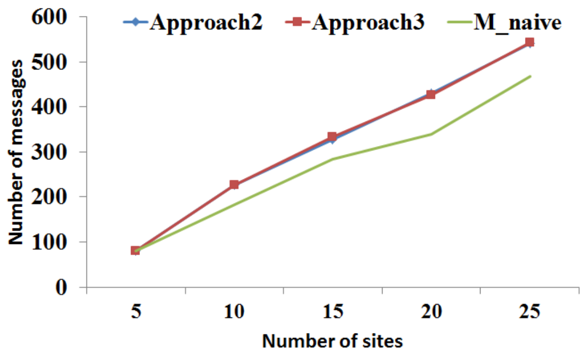

Figure 4: Result for correlated data with varying number of sites[data also varying according to number of sites]

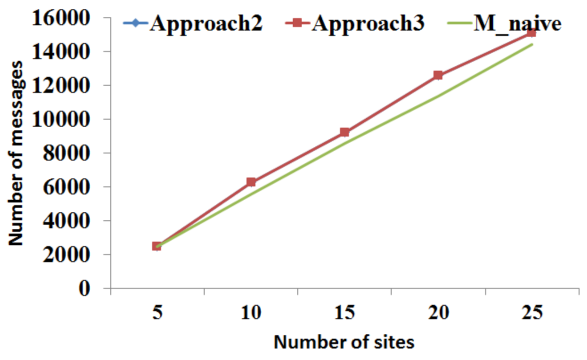

Figure 5: Result for independent data with varying number of sites[data also varying according to number of sites]

3.3 Analysis

In case of the naive system (i.e. naive way of solving the problem), the message count is observed to be constant and the message count will be given by:
Message count=total database size - originator database size.
The time for the computation of the k-dominant skyline in this approach is less than all the other approaches as it just sends all the data to the originator server and the time taken will depend upon the bandwidth of the network used. But for large database it is infeasible to occupy entire data set in memory and may need additional overhead of temporary storage at local database of query originator.But in case of modified naive approach, this number will be much less as we are sending only the full skyline set from each site. Hence comparison with modified naive will be more appropriate.

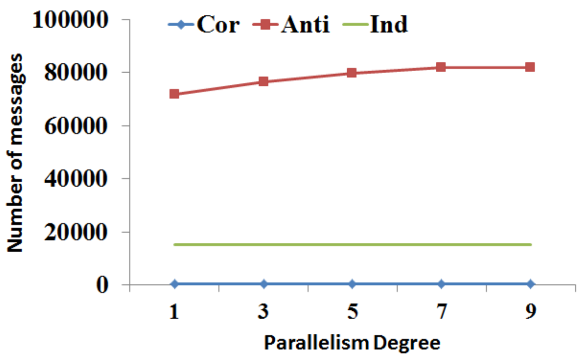

Figure 6: Result for approach 2 with varying parallel degree only

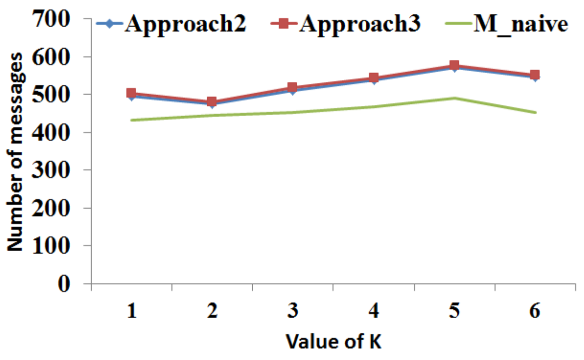

Figure 7: Result for correlated with varying k only [analysis of rest of values are not shown as the plot for them was same as for the given dataset]

Approach 1 and 2 differs response time and the filtering of the dataset.The time for approach 1 is less than the computation time for the approach 2 as expected and the message count is comparable.Theoretically, we can say that for highly distributed database the traversal time of query will dominate the computation time.So from now onwards we will not talk about approach 1.
For testing rest of our proposed approaches, we are varying a factor and keeping remaining constant. The following are the possible combinations against which we are testing our system:

1. keep k,dataset size,number of site fixed and vary parallelism degree. [see Figure 6].
2. keep dataset size, number of sites and parallelism degree constant and vary the value of k. [see Figure 7].
3. keep k,parallelism degree constant and vary the number of sites.[see Figure 3,4,5].

The time for approach 2 is better than the approach 3 in all the cases mentioned but the message count is comparable. Theoretically, it was expected that exploiting the dependencies between different sites should perform better than the modified naive algorithm but practically it is observed that the modified naive perform better in all the cases in both the aspects(computation time as well as message count).This might be happening because as our proposed algorithms have a tree for the computation. The data points which are getting pruned are equivalent to the number of skylines travelling over multiple hops.

4 Conclusion

The skyline operator is quiet useful in multi-criteria decision making applications till the number of dimensions are less. As the number of dimensions increases, we loose the insight of interesting points.Hence k-dominant skylines are more important if the number of dimensions are more. By this project, we propose an algorithm to find k-dominant skyline in distributed setting.From our analysis, we can conclude that the modified naive outperforms all other approaches.

References

[1] Chee-Yong Chan, H. V. Jagadish, Kian-Lee Tan, Anthony K. H. Tung, and Zhenjie Zhang. Finding k-dominant skylines in high dimensional space. In *Proceedings of the 2006 ACM SIGMOD International*

Conference on Management of Data, SIGMOD ’06, pages 503–514, New York, NY, USA, 2006. ACM.

[2] João B. Rocha-Junior, Akrivi Vlachou, Christos Doulkeridis, and Kjetil Nørvåg. Efficient execution plans for distributed skyline query processing. In *Proceedings of the 14th International Conference on Extending Database Technology*, EDBT/ICDT ’11, pages 271–282, New York, NY, USA, 2011. ACM.