# Efficient Probabilistic Skyline operator over Uncertain data using Mapreduce

***Abstract*— This paper presents an efficient probabilistic skyline algorithm over uncertain data using mapreduce.**

## I. Introduction

Uncertain database management has been studied extensively, and skyline query over uncertain database provides many significant applications, such as multi-criteria decision-making over uncertain data. Given an uncertain data set including probability distribution of every object, which is represented by a weighted set of tuples called instances for one object, probabilistic skyline computation finds the probability of each object being in the skyline set, or its skyline probability. A number of efficient algorithms [12] [2] [3] [8] have been proposed for Probabilistic skyline evaluation, but they all focus on solving the problem in a single machine.

Besides, when comes to skyline query for large datasets, the straightforward approach is to parallel the skyline processing, which also has drawn a lot of interests recently [1] [11] [15] [5]. However, All previous algorithms are concentrated on paralleling the traditional skyline query, not probabilistic skyline query. In this paper, we focus on evaluating the probabilistic skyline query over uncertain database using MapReduce [6]. In addition, mapreduce has been incorporated into increasing number of database applications [9] [16] [13] [10].

Our work is most related to [7], as they also concentrated on distributed skyline queries over uncertain data. However, Ding et al. [7] assumes that every tuple in distributed uncertain database is attached with a probability, which could be regraded as a bivariate distribution, rather than multivariate distribution. Obviously the assumption is not in accord with real world. In addition, their algorithm can not be evaluated in mapreduce architecture, and is not efficient.

## II. Preliminaries

### A. Probabilistic Skyline over Uncertain Data

Given a d-dimensional data set, skyline query returns a set of points that are not dominated by any other points. A point $p$ dominating a point $q$ is denoted as $p \prec p$; conversely, $p \not\prec p$ represents that $p$ is not able to dominate $q$. The domination relationship between points can be easily extended to group relationships easily. For example, $p \prec D$ denotes that $p$ dominates all points in $D$. Next, let us introduce the components of uncertain datasets.

Let $OS = \{O_1, O_2, \ldots, O_n\}$ denotes the uncertain object set, and the number of objects is $n$. We assume that an certain object has several instances, each of which is a d-dimensional attribute vector and has a probability to happen. Formally, $O_i$ has an instance set $O_i = \{p_1, p_2, \ldots, p_m\}$, consisting of multiple $d$-dimensional instances with discrete probability density distribution of $O_i$. Any instance $p$'s occurance probability $Pr(p)$ only depends on the object's $pdf$(probability density distribution), i.e., every object is independent of each other. We assume that $\sum_{p \in O} Pr(p) = 1$ applies for every object $O_i$. This is a realistic assumption adopted in many literatures [12] [3] [8] for analyzing uncertain data.

Now we introduce the definition of Probabilistic Skyline, whose definition is quite different from traditional skyline. The goal of Probabilistic Skyline is to obtain the skyline probability $SKYProb(O_i)$ of one object $O_i$ in the skyline set, i.e., the likelihood that $O_i$ becomes a skyline point. Given two Objects $O_u$ and $O_v$, the probability that one instance $p$ in $O_v$ is dominated by another object $O_u$ is:

$$Pr(O_u \prec p) = \sum_{q \in O_u \ q \prec p} Pr(q) \tag{1}$$

Intuitively, the probability that $p$ is a skyline instance for $O_u$ is equal to that $p$ is not dominate by $O_u$, which can be represented by:

$$Pr(O_u \not\prec p) = 1 - \sum_{q \in P_u \ q \prec p} Pr(q) \tag{2}$$

Then we obtain the probability that $p$ is not dominated for all objects in $OS$ except $O_v$, which can be regarded as the likelihood of $p$ to be a global skyline instance. As objects are independent of each other, $SKYProb(p)$ $(p \in O_v)$ is computed as follows:

$$SKYProb(p) = \prod_{O_u \in OS \ O_u \neq O_v} (1 - \sum_{q \in O_u \ q \prec p} Pr(q)) \tag{3}$$

Finally, for object $O_v$, we are able to obtain the skyline probability that $O_v$ is a skyline object:

$$SKYProb(Ov) = \sum_{p \in O_v} Pr(p) SKYProb(p) \tag{4}$$

Rather than obtaining the objects whose $SKYProb(O)$ is larger than a threshold $p$ [12] [7], we compute every object's $SKYProb(O)$. Our target is the same as [2] [8].

### B. MapReduce framework

A typical MapReduce framework consists of two user-defined functions: map and reduce. For every record in the input data sets, map function will partition it into a sorted set of intermediate results. The reduce function fetches the data, from individual partition. provided by the map function, Reduce process produces the final output data. The map and reduce function could be fomally defined: $map(k_1, v_1) \rightarrow list(k_2, v_2)$ and $reduce(k_2, list(v_2)) \rightarrow list(k_3, v_3)$.

## III. A Baseline Method

It can be found that the arduous part of computing probabilistic skyline is obtaining the sum of weights of all instances that dominate a specific instance in Equation 3. Notice, in our assumption, the dataset is too large to fit into one machine's memory; in fact, all data is distributed in disks of all machines in one cluster. The most straightforward solution is partitioning objects in a pair-wise way, computing probabilistic skyline between one pair of objects in Equation 2 and one instance $p$'s skyline probability $SKYProb(p)$ in Equation 3, merging all instances' intermediate results to the final skyline probability in Equation 4. In this section, we apply the idea to a three-phase and a two-phase MapReduce frameworks.

### A. Three-phase MapReduce

In the first phase of three-phase MapReduce, we create pair-wise comparisons between any two objects. At the end of Map phase, every possible pair of two objects is assigned one key. Assume that the number of objects is $m$, and the number of the whole instances is $n$. Therefore, the number of pairs is going to be $C_m^2$. In the reduce phase, it fetches the object pair with the key (one of $C_m^2$), say the objects are $P$ and $Q$, and performs two block nested loop comparisons. We compute the sum of all instances' weights of P which dominate one instance of Q (Equation2). That is, for every q in Q, $Pr(P \not\prec q)$ is computed. Reversely, we compute the value of $Pr(Q \not\prec p)$for every p in P. The results from all reducers are written into $C_m^2$ HDFS files.

In the second phase, our target is to find the global skyline probability for each instance. So we read the output from the first phase and use the instance ID key as the partitioning key at the end of Map phase. Hence, every reducer will group all object skyline probability for each instance and do an multiplication of them (Equation3). The output from second phase reducers are written into $n$($n$ is the whole number of instances of all objects) files in HDFS.

Similarly, in the third phase, output generated by the second phase is read by map function and every object ID is assigned as a new partitioning key for reduce phase. In the Reduce function, We sum all instance's global skyline probability for one object (Equation4) as the final result, that is, the final object skyline probability.

### B. Two-phase MapReduce

For the three-phase MapReduce algorithms, in most times, we have many similar operations in the second and third phase, and it suffer from high I/O overhead. It give us the incentive that we create the two-phase MapReduce function which merge the the process of Equation3 and Equation4.

The first phase in Two-phase MapReduce is the same as in the Three-phase MapReduce. It generates $C_m^2$ HDFS files, which contain the local skyline results of every pair of objects. In the second phase, These files are read in the map function, and the object ID is assigned as the partitioning key for the reduce phase; then, in the reduce phase, we create a local hash for every instance read: instance ID is the key in hashing, and instance probability array is its value. Then we compute every instance's global skyline probability by multiplication of all values of one specific instance. After that, the object skyline probability is obtained by the sum of the product of every instance's $Skyprob$ and its own existing $Prob$.

It is easily found that efficiency in two-phase algorithms is much higher than in three-phase algorithms.

### C. Challenge

However, it has several restrictions for this approach: (1). how to partition objects evenly is hard, since the number of instances might be highly different in various object. In addition, some object might occupy billions of instances, while some other object occupies dozens; (2). besides, we don't know the distribution of instances of one object in advance. If all the instance of one object always appear in the right-corner of coordinate axis, and the object is certain to be a non-skyline object. The optimal approach should be that the object is pruned in the early stage; (3). for every object, we must put every instance into the machine, which is obviously inefficient. the communication cost of putting every instance into one machine is too expensive, especially the IO cost is high. the detailed analysis is in the next paragraph. One improvement is indexing all points before query begins.

### D. Experimental Evaluation

To compare the performance between three-phase, two-phase Hadoop implementations in one cluster and the straightforward implementation on a single machine, we implemented the three algorithms and conducted a experimental evaluation.

There're several observations found in the experiment:

1). Let me first introduce the result of single-machine algorithm. With increasing size of the dataset, the computing time is increasing exponentially. And the most arduous part in the whole computing procedure is in Equation2.

In the first experiment set, we randomly generated 100 objects, and 1 thousand instances per object. The whole computing procedure for Equation2 involves computing the local skyline probability between any pair of two objects. That is 215.6 seconds in our result. In addition, it needs 0.2s for Equation3 process and 0.02s for Equation4 process. Therefore, the whole running time is less than 216s. It can be found that 99% of the time lies on Equation2. However, in our hadoop implementation, the whole query time is about 15 mins. Therefore, if the dataset is small, the time consumed in Hadoop is less than one in single-machine algorithm.

In the second experiment set, we randomly generated 100 objects, and 10 thousand instances per object. The test shows that it needs 5 seconds to run Equation2 for every pair of objects. Since we have $C_{100}^2$ pairs, and the time in all is estimated to be more than 7 hours. However, the time consumed in the hadoop implementation is about 4 hours.

2). When compared between two Hadoop implementations, the two-phase Mapreduce framework is always faster than three-phase one. The reason is that the amount of computing is little in the third phase of the three-phase framework, and merging the second and third phase into one phase reduces one copy of reading data from the HDFS to memory, increasing I/O efficiency.

## IV. MapReduce Framework

The whole MapReduce framework has two phases. In the first map phase, instances of objects are evenly mapped to $N$ machines through angle-based partition scheme. In the reduce phase, unqualified objects are filtered out based on pruning rules proposed, as these unqualified objects are not able to be in the $p-$skyline object set. Objects which are not filtered in this phase become candidate objects and outputted in the reduce function. In the second phase of MapReduce, all candidate objects are grouped into several machines based on a novel grid partition, and every object's final skyline probability is computed. $p-$skyline object set is generated.

### A. partitioning

Angle-based space partitioning scheme [14] uses the hyperspherical coordinates of the data points to project data to parallel machines. In the map phase, Angle-based partitioning strategy maps every data point of instance from Cartesian coordinate space to a hyperspherical space, group data based on angular coordinates into N partitions evenly.

Given a d-dimensional data point $p = [p_1, p_2, \ldots, p_d]$, the hyperspherical coordinates of $p$ include a radial coordinate $r$ and $d-1$ angular coordinates $\varphi_1, \varphi_2, \ldots, \varphi_{d-1}$ [14], denoted by $\{r, \varphi_1, \varphi_2, \ldots, \varphi_{d-1}\}$, where r is the distance to the origin, and $\varphi_i$ represents an angular coordinate $0 \leq \varphi_i \leq \frac{\pi}{2}$. Similarly, an angle-based space partition is represented by $V^A = \{\Phi_1^A, \Phi_2^A, \ldots, \Phi_{d-1}^A\}$, where $\Phi_i^A$ is an angle range $[\varphi_i^j, \varphi_i^k]$ in the $i^{th}$ dimension.

In the detailed implementation of the Map Phase, the number of reducers have been defined prior to the launch of the application, and the default partitioning angles $V^A$ has been computed. the whole instance dataset is an input to Map function. In the Map phase, angular coordinates of data point of every instance are computed, and mapped to target reducer.

### B. Pruning Power

Given an object $O = \{p_1, p_2, \ldots, p_m\}$, the bottom left corner of one object $O_{min}$, is represented by a $d-$dimensional virtual point $(\min_{p_i \in O} p_i[1], \min_{p_i \in O} p_i[2], \ldots, \min_{p_i \in O} p_i[d])$, where $p_i$ is an instance in $O$. Similarly, $O_{max}$ is the top right corner point of $O$.

*Property 1:* Given two objects $O^a$ and $O^b$, $O^a_{max} \prec O^b_{min}$, $O^b$ is not able to be a skyline object in $p-$ skyline set.

*Property 2:* Given an object $O^a$ and a instance $p \notin O^a$, the instance skyline probability of $p$ must be 0 if $O_{amax} \prec p$.

*Property 3:* Given an object $O^a$ and an instance set whose cardinality $|O^a| = m$, $O^a$'s instances are mapped into several partitions (machines). Assume a partition $V^A$ has $n$ instances out of $m$ $(n <= m)$ for $O_a$, $\{p_1, p_2, \ldots, p_n\}$. We use $SKYProb^+(O)$ to represent the upper bound of likelihood of object skyline probability. Similarly, $SKYProb^+(p)$ denotes the likelihood of instance skyline probability.

$$\begin{aligned} SKYProb^+(O_a) =& Pr(p_1)SKYProb^+(p_1)+ \\ & Pr(p_2)SKYProb^+(p_2) + \cdots + \\ & Pr(p_n)SKYProb^+(p_n)+ \\ & (1 - Pr(p_1) - Pr(p_2) - \cdots - Pr(p_n)) \end{aligned} \tag{5}$$

If $SKYProb^+(O_a)$ is less than $p$, we say that $O_a$ can not in the $p-$skyline result.

*Proof:* The real Skyline Probability of an object $O_a$,

$$\begin{aligned} SKYProb(O_a) =& Pr(p_1)SKYProb(p_1)+ \\ & Pr(p_2)SKYProb(p_2) + \cdots + \\ & Pr(p_n)SKYProb(p_n) + \cdots + \\ & Pr(p_m)SKYProb(p_m) \end{aligned} \tag{6}$$

$$Pr(p_1) + Pr(p_2) + \ldots Pr(p_m) = 1 \tag{7}$$

Then,

$$\begin{aligned} SKYProb^+(O_a) <=& Pr(p_1)SKYProb(p_1)+ \\ & Pr(p_2)SKYProb(p_2) + \cdots + \\ & Pr(p_n)SKYProb(p_n)+ \\ & Pr(p_{n+1}) + \cdots + Pr(p_m) \\ <=& Pr(p_1)SKYProb^+(p_1)+ \\ & Pr(p_2)SKYProb^+(p_2) + \cdots + \\ & Pr(p_n)SKYProb^+(p_n)+ \\ & (1 - Pr(p_1) - Pr(p_2) - \cdots - Pr(p_n)) \end{aligned} \tag{8}$$

■

### C. Preprocess

Before MapReduce job is launched, every object $O$'s $O_{min}$ and $O_{max}$ is computed locally by iterating all instances of every object $O$ and computing $O_{min}$ and $O_{max}$ of every object, and stored in a file.

### D. First Phase

The first map phase maps data points (instances) to $N$ machines according to angular partition scheme. The defined map function computes every instance's hyperspherical coordinate, and maps instances to designated reducer (machine). In addition, if any instance of one object $O$ is mapped to a reducer $V^A$, $O_{min}$ and $O_{max}$ are also transmitted to the

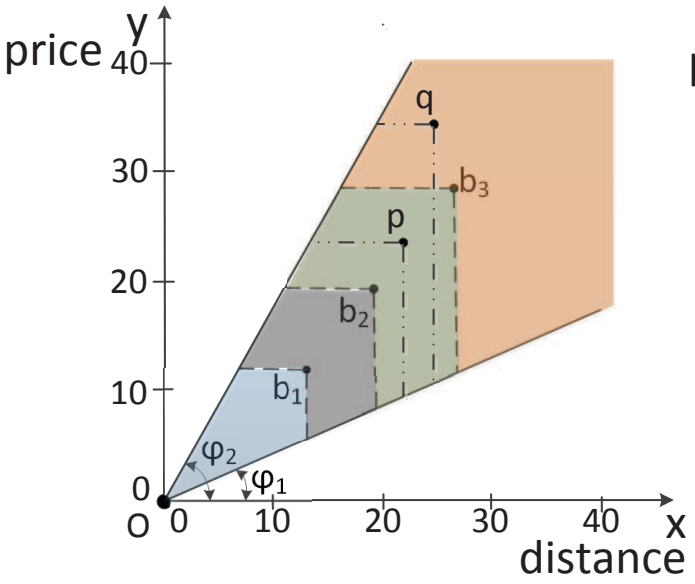



Fig. 1. Prune 3

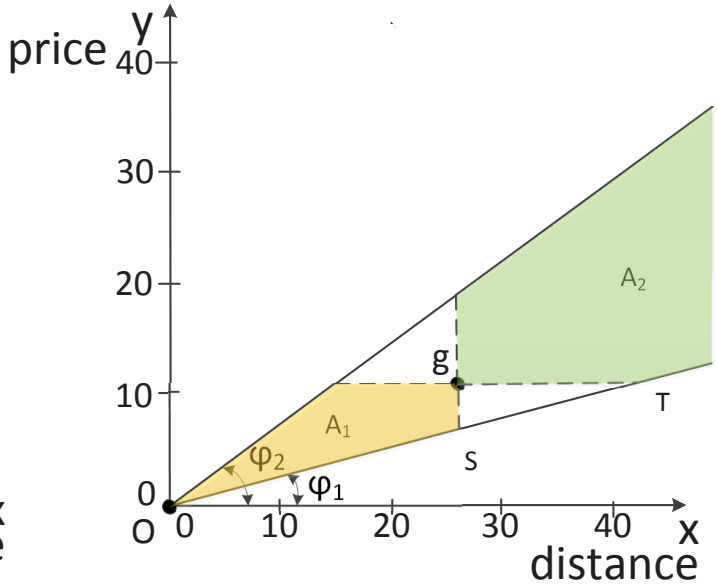



Fig. 2. Pruning Power

partition. Therefore, $O_{min}$ and $O_{max}$ might appear in more than one partitions.

After necessary data is transferred to the reduce phase, only data in this partition is visible. Objects are firstly examined if they can be pruned under the Pruning rule 1. The procedure to examine Pruning rule 1 is as follows. A list $L_{min}$ contains all $O_{min}$ of all objects in this partition; similarly, a list $L_{max}$ contains all $O_{max}$ in this partition. Sort-filter-skyline [4] method is an approach to speed up traditional skyline query by presorting instances. Data point with the sum of all the dimensions of each point is presorted in ascending. The starting element is retrieved from $L_{max}$, and compared with the elements in $L_{min}$ to check the dominance relationship. Any object whose $O_{min}$ is dominated by a $O_{max}$ in $L_{max}$ is filtered, since the object can not be in the $p-$skyline set.

After this, Pruning rule 2 works as follows: we collected remained objects which are not pruned in $L_{min}$. For every instance in these remained objects, we check if a instance in remained objects is dominated by some objects's $O_{max}$. If the dominance relationship occurs, the instance's skyline probability is marked to 0 for the instance can not become skyline instance.

### *E. Rectangle Pruning*

In this subsection, a strategy is proposed to apply Property 3 for filtering unqualified objects in the reduce phase. As we see, in order to compute $SKYProb^{+}(O_a)$ in Equation 5, it is necessary to compute every instance's $SKYProb^{+}(p)$ in this partition $V^A$. Therefore, the challenge comes from how to efficiently obtain $SKYProb^{+}(p)$.

The straightforward method of computing $SKYProb^{+}(p)$ is to iterate every instance in this partition, and compare the dominance relationship between $p$ and every other instances in this partition. The detailed computing procedure is similar to the one in section 3, except that the skyline probability obtained here is the $SKYProb^{+}(p)$ as it only sees incomplete instances. As we already know, the straightforward approach requires high computing complexity.

One interesting observation is that most data points close to right-most boundary are always dominated by left-bottom data points. Take Figure 1 as an example. Assume the partition whose angle is between $\phi_1$ and $\phi_2$ is $V^A$ Three pivot points $b_1$, $b_2$ and $b_3$ separate $V^A$ into four areas, denoted by 4 colors, blue, grey, green and red in order. $p$ is an instance located in green area. It is found that $p$ is always dominated by any instance in blue and grey areas since $b_2$ dominates $p$, and $p$ is also affected by partial instances in green area. Similarly, all instances in blue and grey areas and partial instances in green and red areas dominate $q$. According to this observation, we are able to precompute the sum of instance existing probabilities in blue and grey areas, and use the intermediate result for computing skyline probability of $p$ and $q$. Because most points in green and red area are dominated by instances in blue or (both) grey area, precomputing speeds up the skyline probability computing. Therefore, we partition $V^A$ into several areas, and instances in $V^A$ are grouped separately based on area partitioned. Then those left bottom groups of instances contribute to the skyline probability of right-most data points.

Formally, pivot points $P$ is a set of d-dimensional data points, whose cardinality is $m$, $P = \{b_1, b_2, \ldots, b_m\}$. Correspondingly, $A_i$ is denoted as the d-dimensional hyper-rectangle, left bottom corner of which is the origin, and the top right corner of which is $b_i$. If an instance $p$ is dominated by $b_i$, any instance in $A_i$ dominate $p$.

*Property 4:* Assume that a data point $g$ is dominated by a pivot point $b_i$, all instances residing in $\{A_i\}$ dominate $g$.

Now the problem turns to studying the pruning power of a pivot point $g$. We make the assumption that our dataset is defined in the hypercube $[0, 1]^d$ and data points are uniformly

distributed. Followed by this assumption, the number of data points located in a region could be represented by the region's volume. To simplify the model of pruning power, only one pivot point divides the angle area $V^A$ to two parts denoted as $A_1$ and $A_2$. $A_1$ lies in the left bottom to g, and $A_2$ lies in the right top to g. Figure 2 depicts the scenario. The pruning power could be represented by how many data points in $A_2$ is able to repeatedly apply $A_1$ without comparing one by one, since intermediate sum of instance probability in $A_1$ has been computed well. Therefore, given a pivot point $g$, the pruning power of $g$ is defined as:

$$PP(g) = A_1(x_g, y_g, \phi_1, \phi_2) * A_2(x_g, y_g, \phi_1, \phi_2) \quad (9)$$

where $A_1$ and $A_2$ is two function, which computes corresponding area shown in Figure 2.

In [14], how to compute $A_2$ is introduced. Three cases are considered in formula induction: one is $0 \leq \phi_1 < \phi_2 \leq \pi/4$, the other is $\pi/4 \leq \phi_1 < \phi_2 \leq \pi/2$, and the final one is $0 \leq \phi_1 \leq \pi/4 \leq \phi_2 \leq \pi/2$. The detailed computing formula is in Equation 10.

$$A_2 = \begin{cases} \int_{x_g}^{1} \int_{xtan\phi_1}^{xtan\phi_2} dydx - \int_{x_g}^{min(1, \frac{y_g}{tan\phi_1})} \int_{xtan\phi_1}^{y_g} dydx \\ \qquad \text{if } 0 \leq \phi_1 < \phi_2 \leq \pi/4 \\ A_2(g_y, g_x, \frac{\pi}{2} - \phi_2, \frac{\pi}{2} - \phi_1) \\ \qquad \text{if } \pi/4 \leq \phi_1 < \phi_2 \leq \pi/2 \\ A_2(x_g, y_g, \phi_1, \frac{\pi}{4}) + A_2(g_y, g_x, \frac{\pi}{2} - \phi_2, \frac{\pi}{4}) \\ \qquad \text{if } 0 \leq \phi_1 \leq \pi/4 \leq \phi_2 \leq \pi/2 \end{cases} \quad (10)$$

Similarly, we compute the area of $A_1$. As $y_g = x_g * tan\phi_g$, $A_1$ is the region defined by $0 \leq x \leq x_g$, and $xtan(\phi_1) \leq y \leq min(xtan(\phi_2), y_g)$. Therefore, $A_1$ is represented by

$$A_1 = \int_0^{x_g} \int_{xtan(\phi_1)}^{min(xtan(\phi_2), y_g)} dydx \quad (11)$$

Assume that $0 \leq \phi_1 < \phi_2 \leq \pi/4$, $PP(g)$ is represented by

$$PP(g) = (\int_{x_g}^{1} \int_{xtan\phi_1}^{xtan\phi_2} dydx - \int_{x_g}^{min(1, \frac{y_g}{tan\phi_1})} \int_{xtan\phi_1}^{y_g} dydx) * \int_0^{x_g} \int_{xtan(\phi_1)}^{min(xtan(\phi_2), y_g)} dydx \quad (12)$$

We now evaluate the pruning power of a given point via an experiment. $N = 8$ partitions are defined by $\phi_i$ for $1 < i < 8$, and the pivot point $g$ is always located in $V^A$. $\phi_i$ is tested only when $0 < \phi_i < \pi/4$. Notice that the upper partitions ($\pi/4 < \phi_i < \pi/2$) are symmetrical to these partitions ($0 < \phi_i < \pi/4$). For each $x_g$, we obtain three points where $y_g$ is computed by $y_g = x_g * tan(\phi_g)$ for different $\phi_g$ respectively. Figure 4 shows the cases when three points are located in three positions based on one x value. The points are located in lines at angle of $\phi_1$, $\phi_2$, and the middle line's angle is between $\phi_1$ and $\phi_2$.

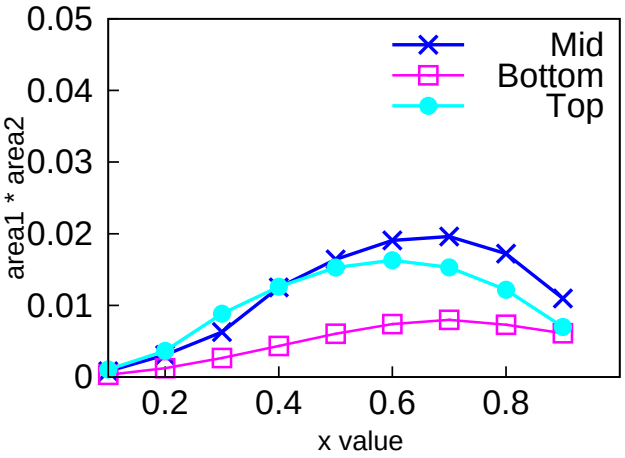



Fig. 3. line

$\phi_1$ and $\phi_2$ is randomly set in the experiment. And the pivot points are selected when $x$ value varies from 0.1 to 0.9. Figure 3 depicts the experiment result. we find that most points in mid lines generate the largest pruning power, compared to points in lines of $\phi_1$ and $\phi_2$.

Based on the experiment result, in order to maximize the pruning power of pivot point, we let pivot points lie in a line $L$ at an angle of $\phi_3$, where $\phi_3 = \frac{\phi_1 + \phi_2}{2}$. Given an input number $N_p$ of how many pivot points are put, pivot points are selected every even interval of x. Take Figure 4 as an example. Pivot points are all located in a line at an angle of $\phi_3$, and $b_{i-1}$ dominates bi, for $1 \leq i \leq n$. As a result, given any point $p$ dominated by $b_i$, instances in $A_i$ dominate $p$

The detailed computing procedure is as follows. Pivot points set $P$ is selected based on $N_p$, $\phi_1$ and $\phi_2$ provided. At the beginning of the reduce phase, the local machine knows which objects are in this partition. For each $A_i$, it holds an object vector $V(o_1, o_2, \ldots, o_m)$ that maintains the sum of existing instances' probability. Given an instance $p$, we firstly

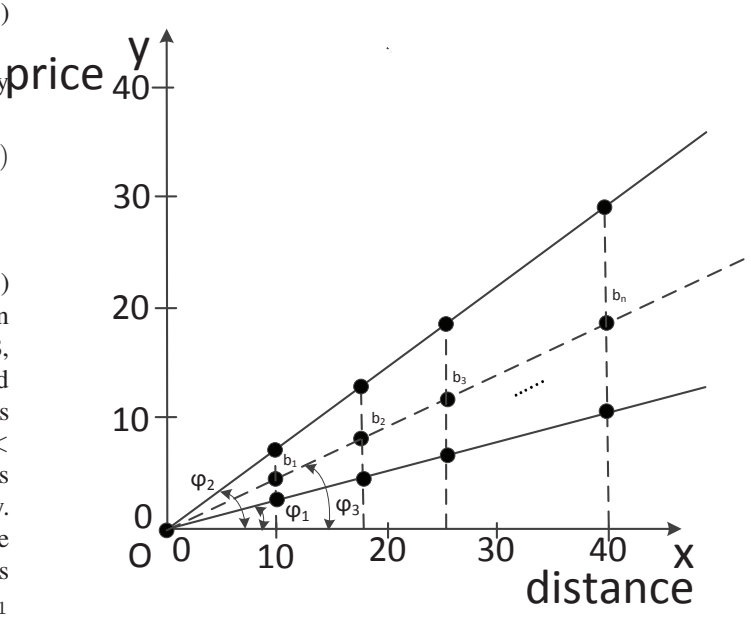



Fig. 4. line

find the nearest pivot point $b_i$ which dominates $p$, and area set $S_A$ which partially dominates $p$. Another object vector $V(o_1, o_2, \ldots, o_m)$ is maintained for comparing domination relationship in $S_A$. If any instance in areas of $S_A$, the instance's existing probability is counted to the object vector. Then, aggregated by the object vector of $A_i$ precomputed before, the product of every element in the object vector is the final $SkyProb^+(p)$.

*Definition 1:* Let $C_o$ be an object set where skyline probability of each object is at least larger than $p$.

*Definition 2:* Let $I$ be an instance set where some instance might dominate instances in $C_o$.

At the end of the first MapReduce phase, we collect candidate object set $C_o$ and influencing instance set $I$. Definition 1 gives the definition of $C_o$, and any object $o$ with $SkyProb^+(0)$ larger than $p$ is in $C_o$. Definition 2 gives the definition of $I$, and any instance $p$ with instance skyline probability $SkyProb(p)$ larger than 0 is gathered in $C_o$. In the final data gathered after the first MapReduce phase, there are two types of instances remained. In the first case, the instance is belonged to some object in $C_o$ and it is also in $I$. In the second case, the instance is only belong to $I$. As soon as the data is collected, it is able to move forward the merging phase.

## V. Second MapReduce Framework

After the first MapReduce phase is completed, the skyline candidate object set $O_c$ is collected. In the next step, the final skyline probability of every object is necessary to compute for determining $p-$skyline objects, Before we illustrate our merging algorithm for filtering $O_c$, we firstly study the data distribution of remained candidate objects after the first MapReduce phase.

### A. Shape Analysis

Since the angle-based pruning only knows the local data at a fixed region, the objects remained in a region decide the final step of merging. Therefore, we study the shape of instance distribution after the first MapReduce phase. To evaluate this, we conducted an experiment to depict the shape of instance distribution. Figure 5 shows the three figures, each of which depicts an instance distribution. The detailed experiment parameters are illustrated as follows. In the two-dimensional space, objects are distributed between [0, 1] in each domain. Three types of object distribution are independent distribution, correlated distribution, and anti-correlated distribution respectively. The cardinality of objects is 5000. For each object, instances are created under a rectangle region, the edge size of which follows a normal distribution in range [0, 0.2] with expectation 0.1 and standard deviation 0.025 The number of instances in each object follows uniform distribution in range [1, 100]. we evenly split the first quadrant into 6 angles, and skyline probability is computed in each angle region using the algorithm in the first MapReduce. After that, we collected the objects which are still remained in each region, and put all of them in the x-y axis.

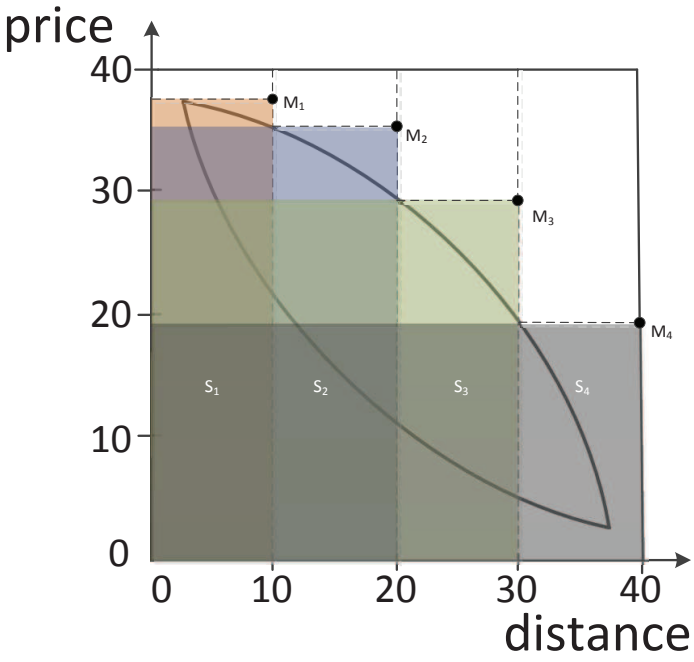


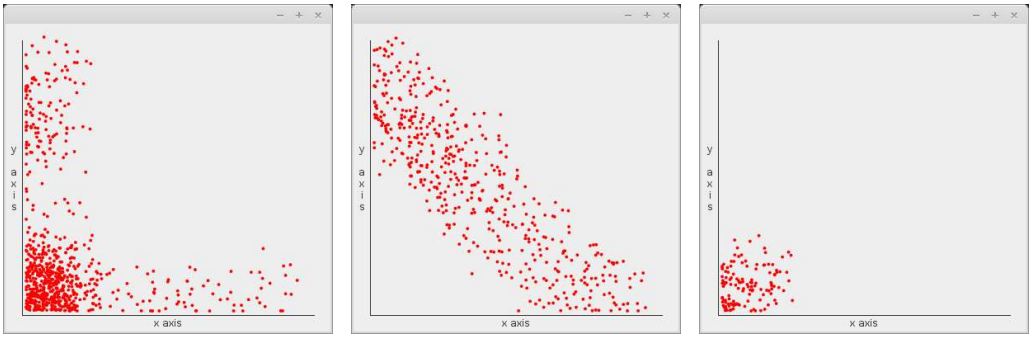

Fig. 6. Merging Phase

In Figure 5, three types of object distribution exhibit different data distributions. In Independent or Anti-correlated distribution of data, there are still many instances remained. Therefore, the Strategy proposed in the merging phase should be able to apply in every case if data distribution, especially for Independent and Anti-correlated Distribution data.

### B. Rectangle Splitting

In this section, we devise an approach to efficiently evaluate skyline probability for those remained objects. To utilize the MapReduce framework, we partition remained data to several domains which are able to work individually. The general idea is to split data into several partitions, and each partition is not affected by others. Take Figure 6 as an example. Assume the remained instances form the anti-correlated distribution. X axis is evenly split into 4 partitions $S_1, S_2, \ldots, S_4$, and 4 maximum coordinate points are computed for each partition. Each point is dominated by any instance in this partition. Then each grid region is rounded by the origin and the maximum point. In Figure 6, green area represents only instances which dominate the instances in $S_3$. Therefore, for computing instance skyline probability in $S_3$, it only needs instances in green area.

Followed by the example introduced above, the data is partitioned based on evenly along the x-axis. Given the the number of reducers $n$, the data is partitioned into n parts. Meanwhile, mapping method groups data into $N$ groups. Every group has two types of data, instances in $C_o$ and $I$ respectively. Take Figure 6 as an example. The instances of $C_o$ in $S_2$ is in $S_2$ rectangle, and instances in the blue area are instance set $I$, which dominates $C_o$. Based on the grouping scheme, the mapping phase maps the two types of data to every slave machine. It is obvious that the choice of $n$ affects the efficiency of skyline probability computing. In the experiment




(a) Independent Distribution (b) Anti-correlated Distribution (c) Correlated Distribution

Fig. 5. Data Point Distribution after phase 1.

section, we observe the performance trend with the variety of $n$.

In the reduce phase, skyline probability of every instance is computed. Assume an instance $p$ is in a partition $V^A$. Since $p$ is only dominated by corresponding region of instances, the instance skyline probability $SkyProb(p)$ is obtained using the standard approach. It finds all instances which instance $p$ by iterating all instances in this partition, and do Equations introduced in the Section II. Then instance skyline probability of every instance is exported to a file in each partition.

After every instance in $C_o$ obtains its instance skyline probability, the second phase is completed. object skyline probability is grouped by iterating all instance skyline probability by Equation 4. $p-$skyline objects area able to be obtained by retrieving objects whose object skyline probability is larger than or equal to $p$.

## VI. Experimental Validation

In this section, we first evaluate our proposed Rectangle pruning methods on a single node. Then, we implemented our MapReduce Distributed System using Akka for communication between nodes. The introduction of MapReduce framework For fair comparison, we developed a single MapReduce phase solution as a baseline solution. All solutions were implemented in Java and Scala.

The experiments were conducted on a 24-node shared-nothing cluster. Each node was equipped with one Intel Xeon 2.4 GHz processor, 2 GBytes of memory. All nodes were connected by GigaBit Ethernet network. We utilized the default configuration in Hadoop. Uniform, correlated and anti-related datasets were randomly generated in a five-dimensional space. The parallel solutions were executed with eight nodes by default. Simulation results were recorded after the system model reached a steady state.

### A. Summary

EX (Exhaustive) is an exhaustive algorithm for computing skyline probability in parallel environment. Our approach use two optimized way to speed up the skyline probability computation: one is Rectangle optimization in the first MapReduce phase, the other is one in the second phase.

- vary the object number
- vary $p$
- vary dimension
- how much percent of data is able to be pruned by the first and third pruning rules provided, given the number of partitions, and the number of pivot points. Meanwhile, the response time is measured.
- Given the number of partitions in the Second MapReduce phase, the response time of the reduce phase is measured.
- all evaluation has three types of data distribution: Correlated, anti-correlated, independent.
- communication cost

TABLE I 6 partitions

| Partition | iD2T2M0C5000 | | | cD2T2M0C5000 | | | aD2T2M0C5000 | | |
|---|---|---|---|---|---|---|---|---|---|
| | original No. | after Prune1 | after Prune3 | original No. | after Prune1 | after Prune3 | original No. | after Prune1 | after Prune3 |
| 1 | 734 | 72 | 43 | 354 | 202 | 35 | 1050 | 532 | 78 |
| 2 | 893 | 13 | 11 | 1225 | 290 | 15 | 1526 | 663 | 25 |
| 3 | 1171 | 7 | 5 | 3522 | 347 | 22 | 1651 | 692 | 24 |
| 4 | 1175 | 8 | 7 | 3587 | 319 | 12 | 1644 | 842 | 34 |
| 5 | 873 | 20 | 11 | 1321 | 257 | 11 | 1605 | 916 | 37 |
| 6 | 717 | 84 | 36 | 379 | 106 | 13 | 1154 | 427 | 69 |

TABLE II 4 partitions

| Partition | iD2T2M0C5000 | | | cD2T2M0C5000 | | | aD2T2M0C5000 | | |
|---|---|---|---|---|---|---|---|---|---|
| | original No. | after Prune1 | after Prune3 | original No. | after Prune1 | after Prune3 | original No. | after Prune1 | after Prune3 |
| 1 | 1101 | 72 | 41 | 726 | 268 | 35 | 1480 | 519 | 85 |
| 2 | 1563 | 8 | 8 | 3689 | 383 | 23 | 2027 | 818 | 39 |
| 3 | 1583 | 9 | 7 | 3766 | 357 | 12 | 2083 | 967 | 54 |
| 4 | 1089 | 69 | 38 | 753 | 137 | 15 | 1622 | 549 | 88 |

TABLE III 2 partitions

| Partition | iD2T2M0C5000 | | | cD2T2M0C5000 | | | aD2T2M0C5000 | | |
|---|---|---|---|---|---|---|---|---|---|
| | original No. | after Prune1 | after Prune3 | original No. | after Prune1 | after Prune3 | original No. | after Prune1 | after Prune3 |
| 1 | 2577 | 73 | 38 | 3762 | 398 | 44 | 2834 | 842 | 112 |
| 2 | 2511 | 72 | 38 | 3844 | 221 | 22 | 2964 | 957 | 131 |

TABLE IV 6 partitions in Three Dimension

| Partition | iD2T2M0C5000 | | | cD2T2M0C5000 | | | aD2T2M0C5000 | | |
|---|---|---|---|---|---|---|---|---|---|
| | original No. | after Prune1 | after Prune3 | original No. | after Prune1 | after Prune3 | original No. | after Prune1 | after Prune3 |
| 1 | 1106 | 17 | 10 | 1066 | 187 | 146 | 1126 | 19 | 9 |
| 2 | 1149 | 14 | 10 | 387 | 57 | 40 | 1152 | 14 | 12 |
| 3 | 750 | 13 | 8 | 1171 | 234 | 186 | 748 | 15 | 11 |
| 4 | 1132 | 12 | 2 | 1092 | 180 | 155 | 1131 | 16 | 4 |
| 5 | 1266 | 14 | 13 | 367 | 57 | 37 | 1255 | 14 | 8 |
| 6 | 789 | 13 | 14 | 1267 | 212 | 179 | 788 | 15 | 10 |

TABLE V 4 partitions in Three Dimension

| Partition | iD2T2M0C5000 | | | cD2T2M0C5000 | | | aD2T2M0C5000 | | |
|---|---|---|---|---|---|---|---|---|---|
| | original No. | after Prune1 | after Prune3 | original No. | after Prune1 | after Prune3 | original No. | after Prune1 | after Prune3 |
| 1 | 1413 | 61 | 44 | 1392 | 211 | 171 | 2081 | 6 | 5 |
| 2 | 1159 | 66 | 37 | 1171 | 223 | 185 | 748 | 5 | 5 |
| 3 | 1447 | 63 | 35 | 1399 | 194 | 160 | 2176 | 9 | 8 |
| 4 | 1228 | 92 | 46 | 1267 | 202 | 173 | 788 | 8 | 7 |

TABLE VI 2 partitions in Three Dimension

| Partition | iD2T2M0C5000 | | | cD2T2M0C5000 | | | aD2T2M0C5000 | | |
|---|---|---|---|---|---|---|---|---|---|
| | original No. | after Prune1 | after Prune3 | original No. | after Prune1 | after Prune3 | original No. | after Prune1 | after Prune3 |
| 1 | 2523 | 116 | 83 | 2505 | 428 | 348 | 2670 | 26 | 7 |
| 2 | 2603 | 149 | 84 | 2608 | 387 | 333 | 2792 | 16 | 6 |

TABLE VII 6 partitions

| Partition | iD2T2M0C5000 | | | cD2T2M0C5000 | | | aD2T2M0C5000 | | |
|---|---|---|---|---|---|---|---|---|---|
| | naive(ms) | prune1-2(ms) | prune1-2-3(ms) | naive(ms) | prune1-2(ms) | prune1-2-3(ms) | naive(ms) | prune1-2(ms) | prune1-2-3(ms) |
| 1 | 4064 | 210 | 835 | 178 | 130 | 602 | 2213 | 532 | 1520 |
| 2 | 5200 | 18 | 205 | 1001 | 179 | 423 | 2322 | 532 | 1038 |
| 3 | 9732 | 6 | 203 | 10955 | 458 | 817 | 2629 | 579 | 1069 |
| 4 | 9426 | 6 | 123 | 11306 | 441 | 655 | 2448 | 700 | 1347 |
| 5 | 5371 | 7 | 117 | 884 | 136 | 298 | 2453 | 732 | 1527 |
| 6 | 3753 | 114 | 302 | 51 | 15 | 149 | 1805 | 276 | 689 |

TABLE VIII 4 partitions

| Partition | iD2T2M0C5000 | | | cD2T2M0C5000 | | | aD2T2M0C5000 | | |
|---|---|---|---|---|---|---|---|---|---|
| | naive(ms) | prune1-2(ms) | prune1-2-3(ms) | naive(ms) | prune1-2(ms) | prune1-2-3(ms) | naive(ms) | prune1-2(ms) | prune1-2-3(ms) |
| 1 | 9077 | 240 | 947 | 482 | 177 | 795 | 4103 | 771 | 2035 |
| 2 | 17444 | 18 | 186 | 15771 | 609 | 984 | 5146 | 926 | 1667 |
| 3 | 17874 | 9 | 204 | 16606 | 543 | 785 | 5027 | 1137 | 2201 |
| 4 | 8952 | 145 | 330 | 271 | 42 | 157 | 4358 | 618 | 1243 |

TABLE IX 2 partitions

| Partition | iD2T2M0C5000 | | | cD2T2M0C5000 | | | aD2T2M0C5000 | | |
|---|---|---|---|---|---|---|---|---|---|
| | naive(ms) | prune1-2(ms) | prune1-2-3(ms) | naive(ms) | prune1-2(ms) | prune1-2-3(ms) | naive(ms) | prune1-2(ms) | prune1-2-3(ms) |
| 1 | 51631 | 353 | 1158 | 16783 | 841 | 1735 | 20143 | 1882 | 4041 |
| 2 | 52009 | 268 | 554 | 18263 | 435 | 643 | 20859 | 2085 | 4363 |

TABLE X 6 partitions in Three Dimension

| Partition | iD2T2M0C5000 | | | cD2T2M0C5000 | | | aD2T2M0C5000 | | |
|---|---|---|---|---|---|---|---|---|---|
| | naive(ms) | prune1-2(ms) | prune1-2-3(ms) | naive(ms) | prune1-2(ms) | prune1-2-3(ms) | naive(ms) | prune1-2(ms) | prune1-2-3(ms) |
| 1 | 5663 | 203 | 921 | 2243 | 72 | 665 | 6247 | 318 | 1265 |
| 2 | 10115 | 46 | 267 | 47671 | 47 | 347 | 9552 | 222 | 623 |
| 3 | 4732 | 50 | 297 | 112 | 5 | 178 | 5062 | 163 | 508 |
| 4 | 4875 | 79 | 262 | 1165 | 9 | 123 | 6395 | 187 | 488 |
| 5 | 11461 | 81 | 236 | 55666 | 22 | 235 | 10420 | 215 | 449 |
| 6 | 5804 | 110 | 296 | 158 | 2 | 94 | 5415 | 169 | 493 |

TABLE XI 4 partitions in Three Dimension

| Partition | iD2T2M0C5000 | | | cD2T2M0C5000 | | | aD2T2M0C5000 | | |
|---|---|---|---|---|---|---|---|---|---|
| | naive(ms) | prune1-2(ms) | prune1-2-3(ms) | naive(ms) | prune1-2(ms) | prune1-2-3(ms) | naive(ms) | prune1-2(ms) | prune1-2-3(ms) |
| 1 | 18532 | 257 | 998 | 32354 | 103 | 780 | 16427 | 596 | 1643 |
| 2 | 11271 | 127 | 324 | 3585 | 17 | 1647 | 10852 | 347 | 878 |
| 3 | 18049 | 123 | 352 | 34568 | 25 | 236 | 16147 | 377 | 904 |
| 4 | 13742 | 186 | 397 | 4043 | 6 | 120 | 12966 | 347 | 900 |

TABLE XII 2 partitions in Three Dimension

| Partition | iD2T2M0C5000 | | | cD2T2M0C5000 | | | aD2T2M0C5000 | | |
|---|---|---|---|---|---|---|---|---|---|
| | naive(ms) | prune1-2(ms) | prune1-2-3(ms) | naive(ms) | prune1-2(ms) | prune1-2-3(ms) | naive(ms) | prune1-2(ms) | prune1-2-3(ms) |
| 1 | 58577 | 536 | 1440 | 58079 | 163 | 862 | 55168 | 1584 | 3968 |
| 2 | 64964 | 710 | 1158 | 64775 | 69 | 301 | 60498 | 1448 | 3064 |

TABLE XIII 6 partitions with 50K |O|

| Partition | cD2T2M0C50000 | | | iD2T2M0C50000 | | | aD2T2M0C50000 | | |
|---|---|---|---|---|---|---|---|---|---|
| | original No. | after Prune1 | after Prune3 | original No. | after Prune1 | after Prune3 | original No. | after Prune1 | after Prune3 |
| 1 | 292 | 28 | 13 | 7213 | 384 | 65 | 7808 | 322 | 77 |
| 2 | 4741 | 56 | 18 | 8835 | 81 | 21 | 9257 | 39 | 17 |
| 3 | 24034 | 56 | 19 | 11722 | 41 | 10 | 9917 | 35 | 20 |
| 4 | 23902 | 72 | 17 | 11873 | 45 | 19 | 9913 | 42 | 14 |
| 5 | 4881 | 52 | 12 | 8886 | 77 | 15 | 9500 | 58 | 22 |
| 6 | 322 | 39 | 15 | 7073 | 189 | 70 | 8001 | 321 | 48 |

TABLE XIV 4 partitions with 50K |O|

| Partition | cD2T2M0C50000 | | | iD2T2M0C50000 | | | aD2T2M0C50000 | | |
|---|---|---|---|---|---|---|---|---|---|
| | original No. | after Prune1 | after Prune3 | original No. | after Prune1 | after Prune3 | original No. | after Prune1 | after Prune3 |
| 1 | 1486 | 33 | 17 | 10956 | 389 | 66 | 12029 | 331 | 83 |
| 2 | 26485 | 59 | 23 | 15784 | 49 | 13 | 14127 | 37 | 27 |
| 3 | 26480 | 68 | 18 | 15974 | 63 | 18 | 14191 | 65 | 27 |
| 4 | 1538 | 52 | 17 | 10878 | 194 | 69 | 12394 | 325 | 51 |

## REFERENCES


[1] F. N. Afrati, P. Koutris, D. Suciu, and J. D. Ullman. Parallel skyline queries. In *ICDT*, pages 274–284, 2012.

[2] M. J. Atallah and Y. Qi. Computing all skyline probabilities for uncertain data. In *Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 279–287. ACM, 2009.

[3] C. Böhm, F. Fiedler, A. Oswald, C. Plant, and B. Wackersreuther. Probabilistic skyline queries. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 651–660. ACM, 2009.

[4] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. Skyline with presorting. In *ICDE*, pages 717–719, 2003.

[5] A. Cosgaya-Lozano, A. Rau-Chaplin, and N. Zeh. Parallel Computation of Skyline Queries. In *HPCS*, page 12, 2007.

[6] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, Jan. 2008.

[7] X. Ding and H. Jin. Efficient and progressive algorithms for distributed skyline queries over uncertain data. *Knowledge and Data Engineering, IEEE Transactions on*, 24(8):1448–1462, 2012.

[8] D. Kim, H. Im, and S. Park. Computing exact skyline probabilities for uncertain databases. 2011.

[9] W. Lu, Y. Shen, S. Chen, and B. C. Ooi. Efficient Processing of k Nearest Neighbor Joins using MapReduce. *PVLDB*, 5(10):1016–1027, 2012.

[10] A. Okcan and M. Riedewald. Processing theta-joins using MapReduce. In *SIGMOD Conference*, pages 949–960, 2011.

[11] S. Park, T. Kim, J. Park, J. Kim, and H. Im. Parallel Skyline Computation on Multicore Architectures. In *ICDE*, pages 760–771, 2009.

[12] J. Pei, B. Jiang, X. Lin, and Y. Yuan. Probabilistic skylines on uncertain data. In *Proceedings of the 33rd international conference on Very large data bases*, pages 15–26. VLDB Endowment, 2007.

[13] R. Vernica, M. J. Carey, and C. Li. Efficient parallel set-similarity joins using MapReduce. In *SIGMOD Conference*, pages 495–506, 2010.

[14] A. Vlachou, C. Doulkeridis, and Y. Kotidis. Angle-based space partitioning for efficient parallel skyline computation. In *SIGMOD Conference*, pages 227–238, 2008.

[15] P. Wu, C. Zhang, Y. Feng, B. Y. Zhao, D. Agrawal, and A. El Abbadi. Parallelizing Skyline Queries for Scalable Distribution. In *EDBT*, pages 112–130, 2006.

[16] C. Zhang, F. Li, and J. Jestes. Efficient parallel kNN joins for large data in MapReduce. In *EDBT*, pages 38–49, 2012.

TABLE XV 6 partitions (Time: us)

| Partition | cD2T2M0C50000 | | | iD2T2M0C50000 | | | aD2T2M0C50000 | | |
|---|---|---|---|---|---|---|---|---|---|
| | naive(ms) | prune1-2(ms) | prune1-2-3(ms) | naive(ms) | prune1-2(ms) | prune1-2-3(ms) | naive(ms) | prune1-2(ms) | prune1-2-3(ms) |
| 1 | 666 | 81 | 1307 | 354417 | 3752 | 7206 | 445281 | 3570 | 6284 |
| 2 | 139598 | 152 | 1295 | 461749 | 506 | 1711 | 565292 | 550 | 1671 |
| 3 | 4038701 | 1231 | 2915 | 845655 | 247 | 1450 | 632030 | 649 | 1702 |
| 4 | 3936687 | 1327 | 2761 | 883083 | 358 | 1555 | 643929 | 597 | 1643 |
| 5 | 143735 | 124 | 1049 | 48208 | 428 | 1553 | 627326 | 748 | 1828 |
| 6 | 484 | 60 | 825 | 338058 | 2196 | 3549 | 450133 | 4001 | 6177 |

TABLE XVI 4 partitions (Time: us)

| Partition | cD2T2M0C5000 | | | iD2T2M0C5000 | | | aD2T2M0C5000 | | |
|---|---|---|---|---|---|---|---|---|---|
| | naive(ms) | prune1-2(ms) | prune1-2-3(ms) | naive(ms) | prune1-2(ms) | prune1-2-3(ms) | naive(ms) | prune1-2(ms) | prune1-2-3(ms) |
| 1 | 11369 | 119 | 1536 | 844773 | 5721 | 8986 | 1045580 | 5222 | 8040 |
| 2 | 4859706 | 1166 | 3207 | 1601874 | 667 | 1924 | 1367131 | 1266 | 2541 |
| 3 | 4806449 | 1230 | 3016 | 1660244 | 674 | 1895 | 1390984 | 1250 | 2475 |
| 4 | 11858 | 67 | 923 | 821776 | 3250 | 4662 | 1148428 | 6292 | 8563 |

TABLE XVII 4 partitions with 5K |O| in MapReduce (Time: ms)

| Partition | cD2T2M0C50000 | | iD2T2M0C50000 | | aD2T2M0C50000 | |
|---|---|---|---|---|---|---|
| | Map | Reduce | Map | Reduce | Map | Reduce |
| 1 | 3k | 9k | 3k | 14k | 3k | 22k |
| 2 | 3k | 26k | 3k | 16k | 3k | 36k |
| 3 | 3k | 26k | 3k | 15k | 3k | 34k |
| 4 | 3k | 9k | 3k | 14k | 3k | 24k |

TABLE XVIII 6 partitions with 5K |O| in MapReduce (Time: ms)

| Partition | cD2T2M0C5000 | | iD2T2M0C5000 | | aD2T2M0C5000 | |
|---|---|---|---|---|---|---|
| | Map | Reduce | Map | Reduce | Map | Reduce |
| 1 | 3k | 6k | 3k | 12k | 3k | 19k |
| 2 | 3k | 12k | 3k | 13k | 3k | 21k |
| 3 | 3k | 20k | 3k | 13k | 3k | 31k |
| 4 | 3k | 20k | 3k | 14k | 3k | 32k |
| 5 | 3k | 12k | 3k | 12k | 3k | 23k |
| 6 | 3k | 6k | 3k | 11k | 3k | 19k |

TABLE XIX 4 partitions with 50K |O| in MapReduce (Time: ms)

| Partition | cD2T2M0C50000 | | iD2T2M0C50000 | | aD2T2M0C50000 | |
|---|---|---|---|---|---|---|
| | Map | Reduce | Map | Reduce | Map | Reduce |
| 1 | 8k | 7k | 8k | 907k | 8k | 2284k |
| 2 | 8k | 3835k | 8k | 2107k | 8k | 2612k |
| 3 | 8k | 3889k | 8k | 2045k | 8k | 2632k |
| 4 | 8k | 7k | 8k | 968k | 8k | 2325k |

TABLE XX 6 partitions with 50K |O| in MapReduce (Time: ms)

| Partition | cD2T2M0C50000 | | iD2T2M0C50000 | | aD2T2M0C50000 | |
|---|---|---|---|---|---|---|
| | Map | Reduce | Map | Reduce | Map | Reduce |
| 1 | 8k | 6k | 8k | 801k | 8k | 2054k |
| 2 | 8k | 923k | 8k | 1749k | 8k | 2512k |
| 3 | 8k | 2634k | 8k | 1705k | 8k | 2512k |
| 4 | 8k | 2988k | 8k | 1877k | 8k | 2606k |
| 5 | 8k | 883k | 8k | 1793k | 8k | 2443k |
| 6 | 8k | 6k | 8k | 786k | 8k | 1991k |