*algorithms*

MDPI

*Article*

# MapReduce Algorithm for Variants of Skyline Queries: Skyband and Dominating Queries

**Md. Anisuzzaman Siddique [1], Hao Tian [2] and Mahboob Qaosar [1,2] and Yasuhiko Morimoto [2,*]**

[1] Department of Computer Science and Engineering, University of Rajshahi, Rajshahi 6205, Bangladesh
[2] Graduate School of Engineering, Hiroshima University, Higashi-Hiroshima 739-8521, Japan
* Correspondence: morimo@hiroshima-u.ac.jp

Received: 28 June 2019; Accepted: 7 August 2019; Published: 13 August 2019


**Abstract:** The skyline query and its variant queries are useful functions in the early stages of a knowledge-discovery processes. The skyline query and its variant queries select a set of important objects, which are better than other common objects in the dataset. In order to handle big data, such knowledge-discovery queries must be computed in parallel distributed environments. In this paper, we consider an efficient parallel algorithm for the "$K$-skyband query" and the "top-$k$ dominating query", which are popular variants of skyline query. We propose a method for computing both queries simultaneously in a parallel distributed framework called MapReduce, which is a popular framework for processing "big data" problems. Our extensive evaluation results validate the effectiveness and efficiency of the proposed algorithm on both real and synthetic datasets.

**Keywords:** skyline query; $K$-skyband query; top-$k$ dominating queries; MapReduce

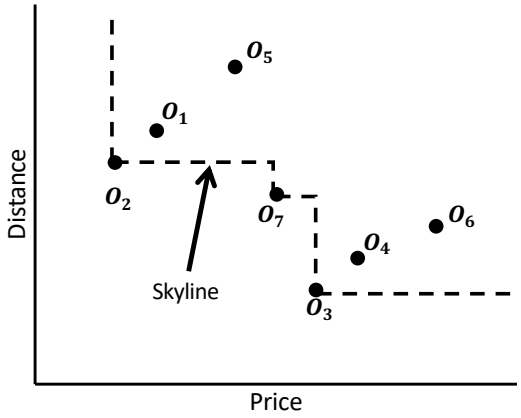
## 1. Introduction

The utilization of large data repositories is a crucial factor in improving various types of businesses. Such a kind of massive data repositories is defined as "big data". Extracting the skyline objects is a vital task for understanding the dataset in the early stage of the knowledge discovery process from large data repositories. Skyline objects in a database are objects that are not dominated by any other object in the database and the skyline query [1] is a function to find a set of skyline objects.

Given an $m$-dimensional dataset $DS$, an object $O_i$ is said to be in the skyline of $DS$ if there is no other object $O_j$ $(i \neq j)$ in $DS$ such that $O_j$ is better than $O_i$. If there exists such an $O_j$, then we say that $O_i$ is dominated by $O_j$, or $O_j$ dominates $O_i$. Figure 1 presents an example skyline. The table in the figure contains a list of hotels, each of which has two numerical attributes: distance and price. If we assume that a smaller value is better, then the skyline query retrieves objects $\{O_2, O_3, O_7\}$ as in Figure 1b. Objects $O_1$ and $O_5$ are dominated by object $O_2$. Objects $O_4$ and $O_6$ are dominated by object $O_3$.

| | $a_1$ | $a_2$ |
|---|---|---|
| ID | Price(Rank) | Distance(Rank) |
| $O_1$ | 3 (2) | 8 (6) |
| $O_2$ | 2 (1) | 7 (5) |
| $O_3$ | 7 (5) | 3 (1) |
| $O_4$ | 8 (6) | 4 (2) |
| $O_5$ | 5 (3) | 10 (7) |
| $O_6$ | 10 (7) | 5 (3) |
| $O_7$ | 6 (4) | 6 (4) |

a) Symbolic Dataset




b) Skyline

**Figure 1.** Skyline example.

Top-$k$ dominating query [2] is a variant of the skyline query. In the query, a scoring function $\mu(O)$ is used for evaluating strongness of an object $O \in DS$: $\mu(O) = |\{O' \in DS \mid O \prec O'\}|$.

The scoring function $\mu(O)$ returns how many objects the object $O$ dominate in the dataset. In the above example, $\mu(O_2) = 2$ because $O_2$ dominates two objects: $O_1$ and $O_5$. Similarly, $\mu(O_3) = 2$. The top-$k$ dominating query selects $k$ objects based on $\mu(O)$. For example, a top-2 dominating query for the example in Figure 1 retrieves $O_2$ and $O_3$.

The skyband query, also known as $K$-skyband query [3], is another well known variant of the skyline query. A $K$-skyband query returns a set of objects, each object of which is not dominated by $K$ other objects. For the dataset in Figure 1a, the skyband query for $K = 2$ retrieves objects $\{O_1, O_2, O_3, O_4, O_7\}$. Object $O_5$ is not in the skyband because it is dominated by two objects: $O_2$ and $O_1$. Similarly, object $O_6$ is not in the skyband result because it is also dominated by two objects: $O_3$ and $O_4$. We illustrate this procedure in Figure 2.

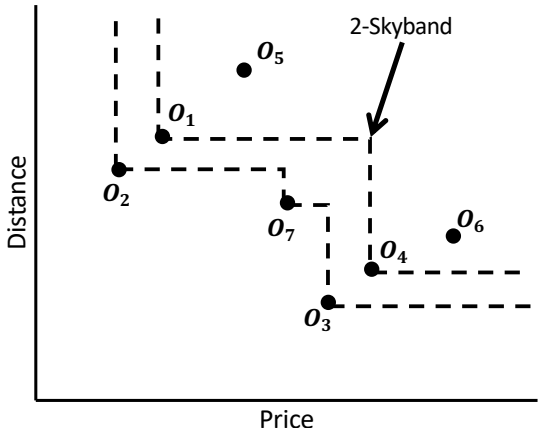



**Figure 2.** Skyband example.

As mentioned above, recent "big data" is too large to analyze intensively. Instead of analyzing raw big data, we propose using a relatively small subset, i.e., the results of skyband and dominating queries that contain the important features of the raw data. However, conventional algorithms for computing such skyline variants are not designed for parallel distributed environments. In recent years, the MapReduce framework has been applied for parallel processing of huge amounts of data on large-size clusters of commodity computers in a reliable manner. MapReduce and Hadoop, which is a popular open source variant of MapReduce, has attracted significant research attention. Our parallel algorithm utilizes the MapReduce framework. In this paper, we propose a MapReduce algorithm, i.e., a parallel algorithm that simultaneously computes skyband and dominating query.

The contributions of this paper can be summarized as follows:

- We examine the skyband and dominating queries for processing "big data".
- We develop a scalable parallel algorithm to compute the queries. The proposed algorithm simultaneously computes skyband and dominating queries. Exploiting the MapReduce framework for the skyband as well as dominating queries is an innovative approach that utilizes the advantage of parallel distributed computing environment.
- The main focus of the proposed algorithm is to distribute the computation process evenly among multiple computing nodes so that "big data" can be effectively processed. We empirically prove the efficiency of the proposed method through intensive experiments using a real dataset and synthetic datasets.

The organization of this paper is as follows. We survey the literature and review related works in Section 2. We present the concepts and properties of skyband and dominating query in Section 3. We describe analysis of proposed algorithm with detailed examples in Section 4. Next, we evaluate our algorithm through intensive experiments in Section 5. Finally, in Section 6, we conclude this paper.

## 2. Related Works

Skyline query and its variants have been widely used in several multi-criteria decision support applications. Borzsonyi et al., who first introduced the skyline query, proposed three basic algorithms for the skyline computation [1]. Those algorithms are known as block-nested-loops (BNL), divide-and-conquer (D&C), and B-tree-based schemes. Chomicki et al. proposed a sort–filter–skyline (SFS) algorithm, which improved the efficiency of the skyline computation by presorting the database attributes [4]. To optimize the average-case running time, Godfrey et al. proposed linear elimination sort for skyline (LESS) algorithm [5]. To filter the dominated objects efficiently through recursively partitioning the dataset based on the nearest objects, Kossmann proposed the nearest neighbor (NN) algorithm for computing the skyline [6].

On the other hand, instead of computing skyline from the original objects' attributes, several algorithms have proposed to use the index of the objects' attributes for computing skyline. Tan et al. has proposed two progressive algorithms to compute skyline based on attributes' bitmap and index [7]. The recent state-of-the-art algorithm is the *Branch-and-Bound Skyline*$(BBS)$, proposed by Papadias et al., which is shown to be I/O optimal for computing skylines on datasets indexed by R-trees [3]. Meanwhile, various approaches have been proposed for effective skyline querying from the high dimensional dataset. Yuan et al. proposed a *skycube* structure to reduce the cost of skyline computation over all possible subspaces [8]. Later, Xia et al. revised the *skycube* structure and proposed CSC structure as a more promising alternative for removing identical skyline objects in the *skycube* by storing each skyline object only to its minimum-subspace [9].

As a variant of the skyline query, Chan et al. introduced the concept of top-$k$ frequent skyline queries [10]. They suggested that a metric, called *skyline frequency*, can be used to rank and select skyline objects by their interesting-ness. Li et al. proposed a data-cube structure to speed up the query evaluation by analyzing the dominance relationship [11]. On the other hand, Lin et al. have considered extracting k most representative skyline objects [12]. They have introduced the concept of a representative object by the population it dominates. According to their definition, a skyline object is more representative than other skyline objects, when it dominates more objects than others. Chan et al. illustrated $k$-dominant skyline based on the measure of the $k$-dominance relationship. The $k$-dominant skyline query can control the number of retrieved objects by changing $k$. If we set a larger $k$ value, an object more likely to be dominated by another object. They developed specialized algorithms to compute the $k$-dominant skyline [13]. $K$-skyband query, which is another variant of skyline query, selects those objects which are dominated by at most ($K$-1) other objects. It has been noticed that, for any increasingly monotone aggregate function, the top-$k$ objects belong to the k-skyband, where $k \leq K$ [3,14].

There exist more spontaneous techniques for skyline query formalization. Lin et al. proposed $n$-of-$N$ skyline query to support online query on data streams, i.e., to find the skyline of the set composed from the most recent $n$ elements. The proposed method considers a very widely distributed dataset, which is impossible to process in a centralized fashion [15]. Balke et al. has also investigated skyline computation over a vertically distributed database [16]. Tao et al. examined skyline query in arbitrary subspaces [17]. Papadias et al. studied on dynamic skyline query [18]. Dellis et al. proposed the reverse skyline query, which selects the number of users who like the given object most based on the dominance relationship among the objects [19] .

Nowadays, the parallel computing paradigm becomes very popular for processing and analyzing "big data". Therefore the computation of skyline and its variants are becoming challenging today. Noted that [10,11,13] cannot be directly applied to evaluate top-$k$ dominating queries. Moreover, the computation of skyband query requires a separate algorithm. This paper proposed an efficient algorithm for computing both types of queries(top-$k$ dominating and skyband) over a large volume of data, such as "big data". For such data intensive applications, the most notable platform, which has attracted a lot of attention, is MapReduce For this kind of data-intensive application, the MapReduce framework has attracted much attention as the most prominent platform [20–22]. It facilitates the deployment of scalable parallel applications on the share-noting machines cluster for processing large dataset. Google's MapReduce or its open source equivalent Hadoop is a powerful tool for building such applications [23]. The MapReduce framework has also been utilized for some of the recent research works on the computation of skyline and $k$-dominant skyline [24–26].

Recently, Ezatpoor et al. [27] exploits the MapReduce framework for computing top-$k$ dominance on incomplete big data. Besides, Chen et al. [28] utilized the Spark streaming framework to process top-$k$ dominating query over the distributed data stream. Both [27,28] divided the data by using the method of the hash map to process the data through distributed computing nodes.

This paper complements the existing efforts to address the $K$-skyband and top-$k$ dominating query problems by the rank of objects obtained using two intuitive scoring functions. Specifically, our algorithm can provide solutions for these two types of queries within the same framework. To the best of our knowledge, there is no such MapReduce algorithm had been proposed for the $k$-dominating query and the $K$-skyband query so far.

## 3. Preliminaries

Consider an $m$-dimensional dataset $DS$ $\{a_1, a_2, \cdots, a_m\}$. We assume that the dataset is distributed into $n$ subsets $\{DS_1, DS_2, \cdots, DS_n\}$ in different locations. Without loss of generality, we assume that the dataset contains non-negative numerical values. We also assume that smaller values are preferable in each dimension/attribute. $O_{i,j}.a_p$ denotes that the $p$-th dimension's/attribute's value for object $O_{i,j}$, where $i, j$ is an object $ID$ which means object $j$ in dataset $i$ ($DS_i$). Assume that the dataset $DS$ shown in Figure 1 is distributed into three subsets, $DS_1$, $DS_2$, and $DS_3$, each of which has two attributes, $a_1$ and $a_2$, as shown in Table 1.

**Table 1.** Dataset $DS$.

| $DS_1$ | | | $DS_2$ | | | $DS_3$ | | |
|---|---|---|---|---|---|---|---|---|
| ID | $a_1$ | $a_2$ | ID | $a_1$ | $a_2$ | ID | $a_1$ | $a_2$ |
| $O_{1,1}$ | 3 | 8 | $O_{2,1}$ | 7 | 3 | $O_{3,1}$ | 5 | 10 |
| $O_{1,2}$ | 2 | 7 | $O_{2,2}$ | 8 | 4 | $O_{3,2}$ | 10 | 5 |
| | | | | | | $O_{3,3}$ | 6 | 6 |

**Definition 1.** *(Dominance) For two objects $O$ and $O'$, object $O$ is said to dominate object $O'$, denoted $O \prec O'$, if $O.a_s \leq O'.a_s$ for all attributes ($s = 1, \cdots, m$) and $O.a_x < O'.a_x$ for at least one attribute ($1 \leq x \leq m$).*

*We refer to such an O as a dominant object and such a O′ as A dominated object. If O dominates O′, then O is preferable to O′.*

In Table 1, object $O_{1,2}$ dominates object $O_{1,1}$ ($O_{1,2} \prec O_{1,1}$). This is because object $O_{1,2}$ has a smaller value for both attributes than object $O_{1,1}$.

**Definition 2.** *(Skyline) An object $O \in DS$ is in a skyline of DS (i.e., a skyline object in DS) if O is not dominated by any other object in DS. The skyline of DS, denoted $Sky(DS)$, is the set of skyline objects in DS. For the dataset DS, objects $O_{1,2}, O_{2,1}$, and $O_{3,3}$ can dominate all other objects and are not dominated by any other object. Therefore, a skyline query on dataset DS will retrieve $Sky(DS) = \{O_{1,2}, O_{2,1}, O_{3,3}\}$.*

**Definition 3.** *(The μ score) The μ score of an object shows how many objects the object dominate in the dataset. We use $\mu(O)$ to denote the μ score of an object O. In Table 1, object $O_{1,2}$ dominates objects $O_{1,1}$ and $O_{3,1}$. Therefore, the μ score of $O_{1,2}$ is 2 (i.e., $\mu(O_{1,2}) = 2$).*

**Definition 4.** *(The SB score). The SB score of an object is the number of objects dominating that object. We use $SB(O)$ to denote the SB score of an object O. In Table 1, object $O_{3,2}$ is dominated by objects $O_{2,1}$ and $O_{2,2}$. Therefore, the SB score of $O_{3,2}$ is 2 (i.e., $SB(O_{3,2}) = 2$).*

**Definition 5.** *(Dominating query) Given a positive integer k and a dataset DS, the top-k dominating query returns the k objects that have the top-k μ scores in DS. For the dataset in Table 1, a top-two dominating query retrieves $O_{1,2}$ and $O_{2,1}$.*

**Definition 6.** *(Skyband query) Given a positive integer K, the K-skyband is the set of objects that are not dominated by K other objects. For the dataset DS in Table 1, the skyband query for $K = 2$ retrieves objects $\{O_{1,1}, O_{1,2}, O_{2,1}, O_{2,2}, and\ O_{3,3}\}$. Intuitively, K represents the thickness of the skyline. A 1-skyband query is the same as a conventional skyline query.*

**Definition 7.** *(Worst rank) For an object O, assume $r_1(O)$ and $r_2(O)$ are the rank values of attributes $a_1$ and $a_2$, respectively. For example, in Figure 1, $r_1(O_4) = 6$ and $r_2(O_4) = 2$. We refer to the largest $r_s(O)$ ($s = 1, ..., m$) as "the worst rank of O" and $a_s$ as "the worst rank attribute of O." In this example, the worst rank of $O_4$ was six and the worst rank attribute was $a_1$.*

**Definition 8.** *(Domination check set) The domination check set (DC) for an object O is the set of objects that have equal or greater rank than the worst rank of O for the worst rank attribute. For example, $O_6$ has a greater rank than the worst rank of $O_4$ (7th > 6th) in $a_1$ (price). Therefore, the DC set of object $O_4$ is $\{O_6\}$. Similarly, the worst rank of $O_1$ is 6 in $a_1$ (distance). So, the DC set of object $O_1$ is $\{O_5\}$.*

From the above definitions, we have observed an important property [28] and a lemma.

**Property 1.** *Top-k dominating queries result always comes from skyband queries result. For example, a top-two dominating query for the example in Figure 1 retrieves $O_2$ and $O_3$. Those are also belongs to two-skyband result (Figure 2).*

**Lemma 1.** *Dominance relation among the objects within a dataset also remains in the transformed ranked dataset. For example, in Figure 1 object $O_2$ dominates object $O_1$, since object $O_2$ has a smaller value for both attributes than object $O_1$. This dominance relation is also true according to the rank dataset. This is because object $O_1$ has a greater rank for both attributes than object $O_2$.*

## 4. Skyband and Dominating Query Processing

Our MapReduce-based algorithm for skyband and dominating queries consists of the following five phases:

P1: Data map and ranking. Each distributed dataset was partitioned vertically. Then, each partition was dispatched to the map workers (mappers). Each map worker, next, generates ($Val$, $OID$) pairs, where $Val$ is the numeric value of the corresponding object in the attribute domain and $OID$ represents the object $ID$. After receiving ($Val$, $OID$) pairs as input, each "reduce" worker (reducer) produced the ($OID$, $Rank$) pairs for each object, where $Rank$ is the rank value of each object in the attribute domain.
P2: Shuffling. In this phase, each map worker outputs ($OID$, $AttrRank$) pairs, where $AttrRank$ is the attribute name with the corresponding attribute rank for each object in the attribute domain. Next, each "reduce" worker also produced ($OID$, $Ranks$) pairs for each object, where $Ranks$ is a list of attribute names and the respective rank value for each attribute.
P3: Worst rank computation. The coordinator collected all $(OID, Ranks)$ pairs to reduce data transmissions from map workers to reduce workers. After rearranging the $(OID, Ranks)$ pairs, the coordinator found the worst attribute rank for each object.
P4: DC sets computation. The coordinator sends the attributes with the worst ranks to the workers, which are responsible for $DC$ set computation for each object. Each worker takes an attribute rank, and the corresponding attribute's worst rank as input and outputs $DC$ sets for each object.
P5: Skyband and dominating objects computation. At this stage, the map workers take $DC$ sets as inputs and perform domination checks between the $DC$ sets and corresponding objects. Finally, the reducer produced the $SB$ score and the $\mu$ score required to compute the skyband query and top-$k$ dominating query, respectively.

### 4.1. Data Map and Ranking

We first vertically split the dataset into $m$ partitions, if the number of attributes in a dataset is $m$. Therefore, if the number of data subsets was $n$, then the total number of partitions was equal to $n \times m$ (e.g., $\{s_{1,1}, \cdots, s_{1,m}, \cdots, s_{n,1}, \cdots, s_{n,m}\}$). For simplicity, we denote $\{s_{1,1}, s_{2,1}, \cdots, s_{n,1}\}$, $\{s_{1,2}, s_{2,2}, \cdots, s_{n,2}\}$, and $\{s_{1,m}, s_{2,m}, \cdots, s_{n,m}\}$ as $\{S_1\}$, $\{S_2\}$, $\cdots$, and $\{S_m\}$, respectively. In the example, $DS_1$ had two attributes $a_1$ and $a_2$. We split $DS_1$ into two partitions called $s_{1,1}$ and $s_{1,2}$. Since we had two partitions, we needed at least two map workers to complete the computation.

Figure 3 illustrates the "data map and ranking" procedure. Figure shows that objects $O_{1,2}$ and $O_{2,1}$ have rank "1" for attribute $a_1$ and $a_2$, respectively. Therefore, $O_{1,2}$ had the smallest $a_1$ value and $O_{2,1}$ has the smallest $a_2$ value.

Recall that in the Hadoop framework in which we have implemented our system, each map worker operates on a non-overlapping partition of the input file independently and the worker emits key-value pair lists in parallel according to a user-defined "map function". In proposed algorithm, each map worker produces ($Val$, $OID$) pairs, where $Val$ is the numeric value of each object in the attribute domain and $OID$ is the corresponding object $ID$. Next, the reduce workers begin their processing job. They receive ($Val$, $OID$) pairs as inputs and produce ($OID$, $Rank$) pairs for each object, where $Rank$ is the ascending order sorted rank value of each object in the attribute domain. To calculate the rank value for each key-value pair of $S_l (l = 1, \cdots, m)$, the corresponding reduce worker sorts its attribute in ascending order. The reduce worker, then, replaces the values with their corresponding ascending rank value.

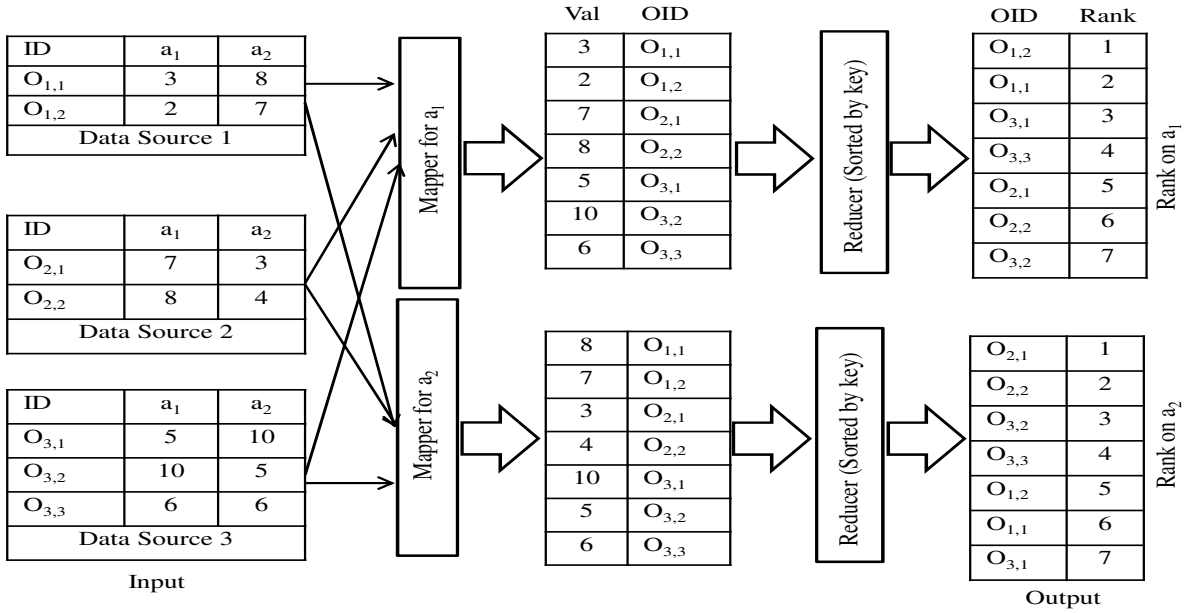



**Figure 3.** Data map and ranking process.

### *4.2. Shuffling*

The second MapReduce phase is invoked for skyband and top-$k$ dominating query computation. After generating $(OID, Rank)$ pairs in the data map and ranking phase, map workers take those pairs as inputs and produce ($OID$, $AttrRank$) pairs, where $AttrRank$ is the attribute name with the corresponding attribute rank for each object in the attribute domain. Then, each map worker dispatches the ($OID$, $AttrRank$) pairs to the reducers. After shuffling, reduce workers produce ($OID$, $Ranks$) pairs for each object, where $Ranks$ is a list of attribute names and respective rank values for each attribute. Figure 4 illustrates the "shuffling" procedure. In the example, $O_{1,1}$ has rank values of two and six for attributes $a_1$ and $a_2$, respectively. Therefore, map workers produce two key-value pairs, $(O_{1,1}, < a_1, 2 >)$ and $(O_{1,1}, < a_2, 6 >)$, for $O_{1,1}$. After shuffling those two pairs, the reduce worker generates an $(OID, Ranks)$ pair for object $O_{1,1}$ as a key-value pair, which is $(O_{1,1}, << a_1, 2 >, < a_2, 6 >>)$. Each reduce worker dispatches $(OID, Ranks)$ pairs to the coordinator.

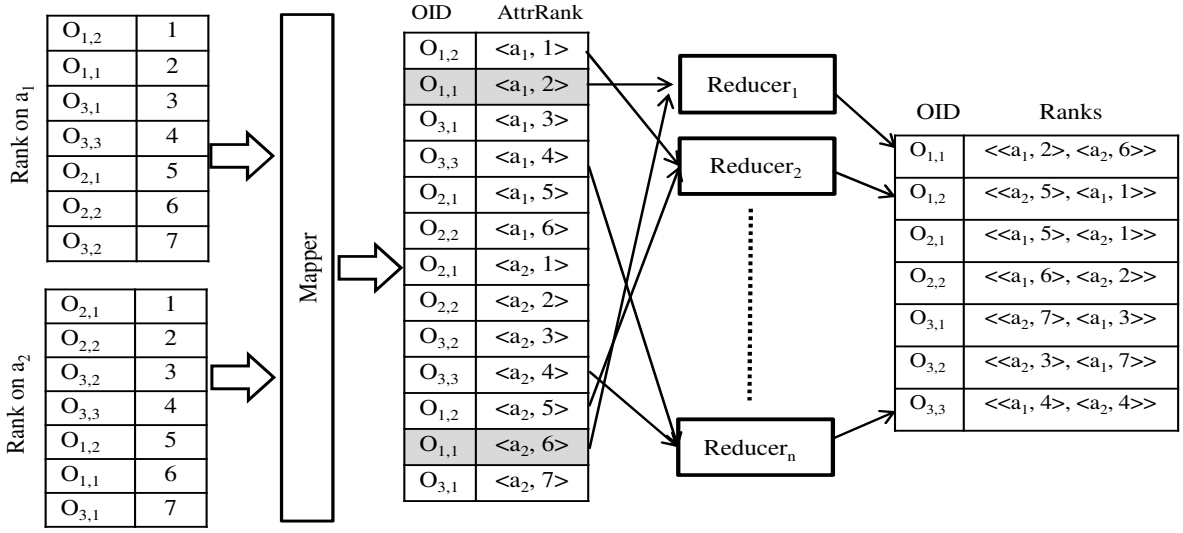



**Figure 4.** Shuffling.

### *4.3. Worst Rank Computation*

The coordinator computes the worst rank and corresponding worst rank attribute for each object. Figure 5 illustrates the "worst rank computation" by the coordinator. In the example, $O_{1,1}$ has rank values of two and six for attributes $a_1$ and $a_2$, respectively. In the object, $a_2$'s rank is the worst among all attribute ranks. Therefore, the worst rank of $O_{1,1}$ is six and the corresponding worst rank attribute of $O_{1,1}$ is $a_2$. Therefore, the coordinator generates $(O_{1,1}, < a_2, 6 >)$ as a key-value pair for $O_{1,1}$.

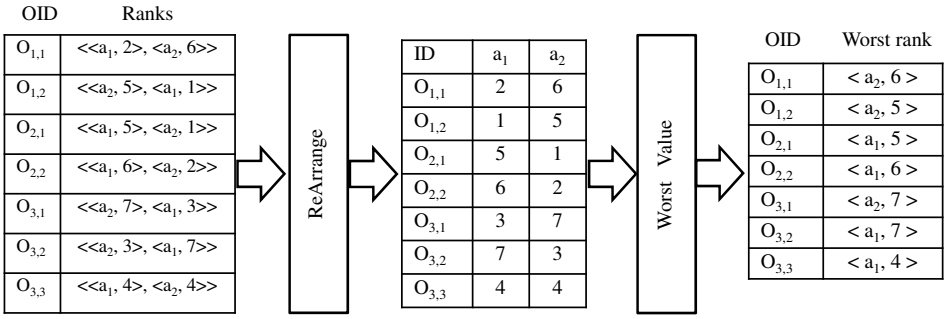



**Figure 5.** Worst rank computation process.

### *4.4. DC Sets Computation*

The coordinator distributes the output pairs to the workers according to the worst rank attribute. Figure 6 presents the "*DC* sets computation" procedure. As shown in the figure, pairs of $O_{2,1}$, $O_{2,2}$, $O_{3,2}$, and $O_{3,3}$ are distributed to worker *DC* $Comp_1$ for $a_1$. Similarly, pairs of $O_{1,1}$, $O_{1,2}$, and $O_{3,1}$ are distributed to worker *DC* $Comp_2$ for $a_2$.

Each worker outputs *DC* sets for each object. In Figure 6, because object $O_{2,1}$ has the worst rank of 5 in $a_1$, *DC* $Comp_1$ outputs $\{O_{2,2}, O_{3,2}\}$ as the *DC* set for $O_{2,1}$. It should be noted that $O_{2,2}$ and $O_{3,2}$ have a greater rank than that of $O_{2,1}$ in $a_1$. Object $O_{1,1}$ has the worst rank of 6 in $a_2$ and *DC* $Comp_2$ outputs the *DC* set member of $O_{1,1}$ as $\{O_{3,1}\}$. We calculate the *DC* sets for other objects similarly.

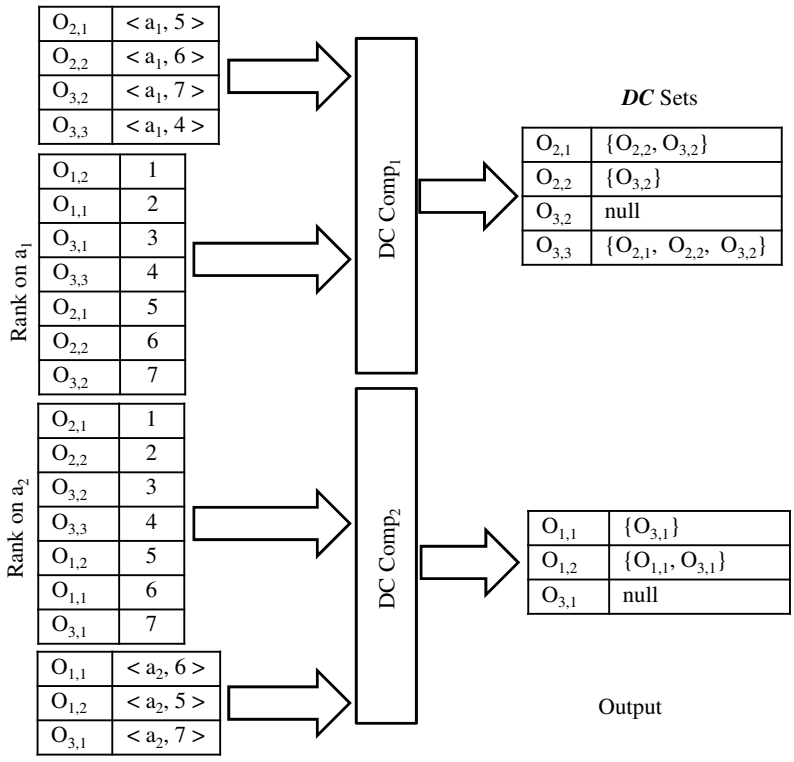



**Figure 6.** Domination check (*DC*) sets computation process.

### *4.5. Skyband and Dominating Objects Computation*

Each map worker takes the previous $DC$ sets as inputs and performs domination checks between corresponding objects and $DC$ sets to produce $(\mu/SB, Score)$ pairs, where the $\mu$ score is the number of objects dominated by an object. In contrast, the $SB$ score of an object is the number of dominant objects. However, to compute either a $\mu$ score or $SB$ score, our method does not need to perform a domination check with any objects outside the $DC$ sets. This advantage stems from the following theorem.

**Theorem 1.** *For two objects $\{O, O' \in DS\}$, if $O'$ is not in the DC set of object $O$, then $O$ cannot dominate object $O'$ (i.e., $O \not\prec O'$).*

**Proof.** Let $a_s$ be the worst rank attribute of $O$. If $O$ dominates $O'$, $O'$ must be in the $DC$ set of $O$ because $O.a_s \leq O'.a_s$. If $O'$ is not in the $DC$ set of $O$, it means $O.a_s > O'.a_s$. Therefore, $O$ cannot dominate $O'$. □

Theorem 1 demonstrates that it is sufficient to perform a domination check between an object $O$ and the corresponding $DC$ set to determine whether or not $O$ is in the query results. To analyze this result, recall object $O_{2,1}$ and its corresponding $DC$ set $O_{2,2}, O_{3,2}$. Because $O_{2,1}$ has the worst rank of 5 for attribute $a_1$, this means that this object has no possibility to dominate a higher rank object, such as $O_{2,1}, O_{1,1}, O_{3,1}$, or $O_{3,3}$. This means we need to perform a domination check only between $O_{2,1}$ and its $DC$ set $\{O_{2,2}, O_{3,2}\}$.

Figure 7 presents the skyband and dominating objects computation process. To compute $K$-skyband, if we set $K = 2$, the query returns $\{O_{1,1}, O_{1,2}, O_{2,1}, O_{2,2}, O_{3,3}\}$ as the two-skyband objects set, since the $SB$ score values for these objects is less than 2. Then, our method outputs objects $O_{1,2}$ and $O_{2,1}$ as the top-2 dominating result because both objects have the highest $\mu$ score value.

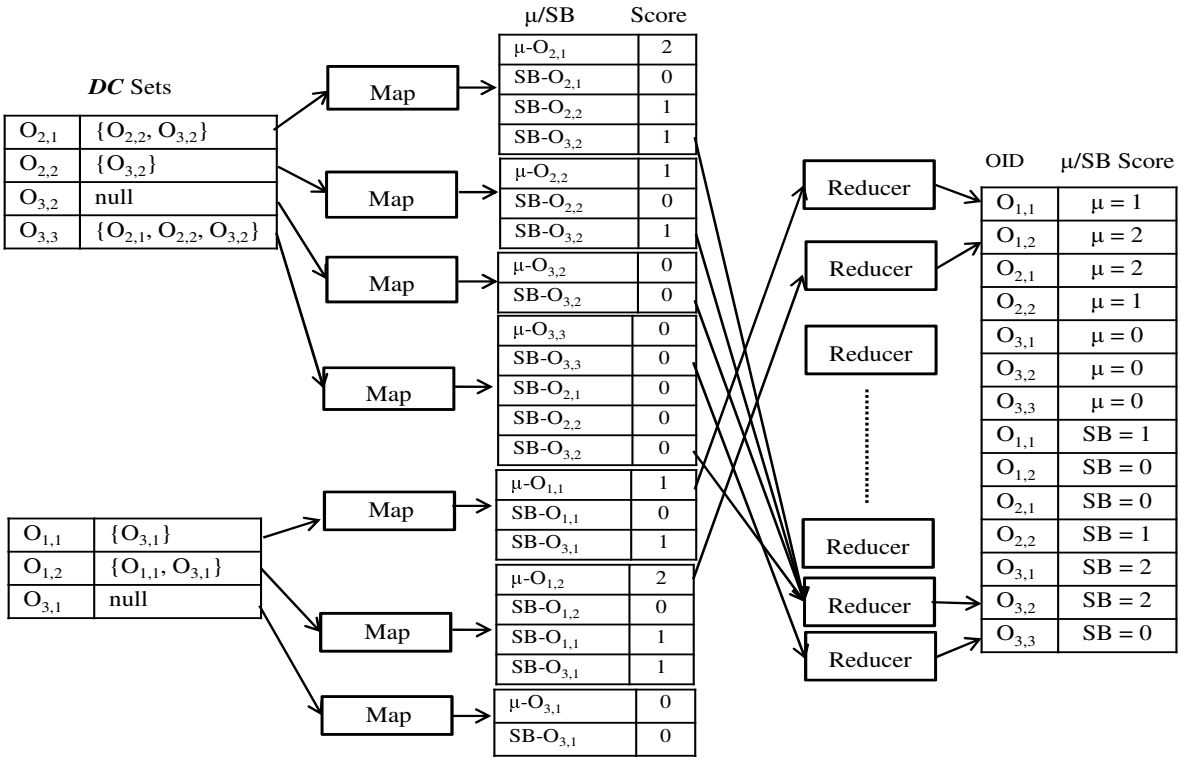



**Figure 7.** Skyband and top-$k$ dominating computation.

After performing a domination check, each map worker produces two types of keys: the $\mu$ score for the corresponding object and the $SB$ score for all objects. If an object dominates another object in the $DC$ set, the $\mu$ score value of the corresponding object and $SB$ score value of the dominated objects are

incremented by 1. Next, all of the $\mu$ scores and $SB$ scores are sent to the reduce workers. After applying a "group-by" operation, the reduce workers can provide $K$-skyband and top-$k$ dominating query results.

## 5. Performance Evaluation

This section presents the results from experimental runs to validate the effectiveness and efficiency of the proposed method. First, to validate its efficiency with a variety of problem settings, Section 5.1 reports evaluation results using synthetic data with various experimental settings. Second, to evaluate the performance of the proposed model on real datasets, Section 5.2 reports evaluation results for the real-world FUEL dataset, which was extracted from "www.fueleconomy.gov". A cluster of four commodity machines was used for our experiments. Each machine had an Intel Core i7 3.4 GHz CPU and 4 GB of memory. These machines were connected through a Gbps LAN connection. The operating system used was CentOS 7.0 with Hadoop version 2.5.2. The algorithm was implemented in Java.

### *5.1. Synthetic Datasets*

This section reports our evaluation results from applying the proposed method to synthetic datasets. First, Section 5.1.1 discusses the effect of dimensionality. Second, Section 5.1.2 discusses the effect of cardinality. Finally, Section 5.1.3 discusses the effect of skyband size. Following the common methodology in the literature, we employ correlated, anti-correlated, and independent (uniform) datasets [1].

#### 5.1.1. Effect of Dimensionality

To study the effect of dataset dimensionality, we used datasets with the cardinality of 100K, skyband size of $K = 5$, and dimensionality $m$ ranging from 2 to 6. The runtime results for this experiment are presented in Figure 8a–c. The results reveal that as dimensionality increases, the performance of the proposed method gradually decreases. This degradation in performance was caused by growth in the number of non-dominant objects in high dimensions. It should be noted that the data distribution factor also influences algorithm performance. As the number of non-dominant objects in the correlated dataset is fewer than that in the anti-correlated and independent datasets, the algorithm with the correlated data generally requires less execution time than those with the other data distributions. Execution on the correlated dataset is 10 times faster than on the independent dataset and 12 times faster than on the anti-correlated dataset.

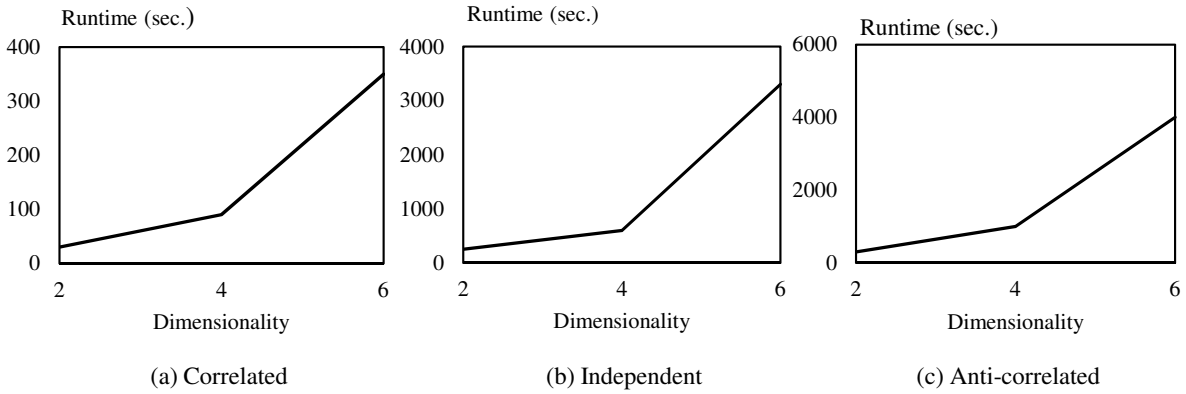
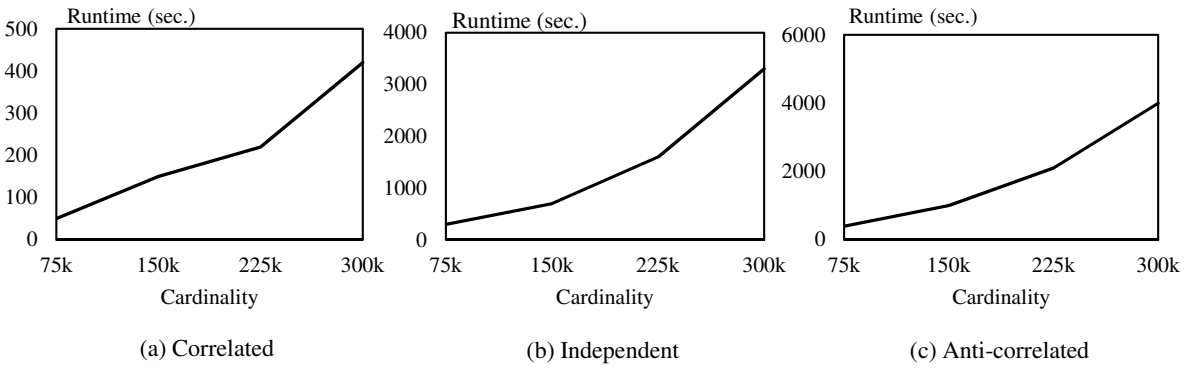



**Figure 8.** Performances with different data dimensions.

#### 5.1.2. Effect of Cardinality

We similarly examined the efficiency of the proposed method with different cardinalities. The results for varying cardinality are presented in Figure 9. For this particular test, we used a dimensionality $m$ of 6, skyband size of $K = 5$, and datasets cardinality ranging from 75,000 to 300,000. Figure 9a–c present the results for the correlated, independent, and anti-correlated datasets, respectively.

The proposed technique had the fastest runtime for all three data distributions when the cardinality is low. However, as the data cardinality increases, the performance of the proposed method consistently decreases. As expected for anti-correlated data, the proposed method fails to dominate a reasonable number of objects, and the performance is worse than that for correlated and independent data.

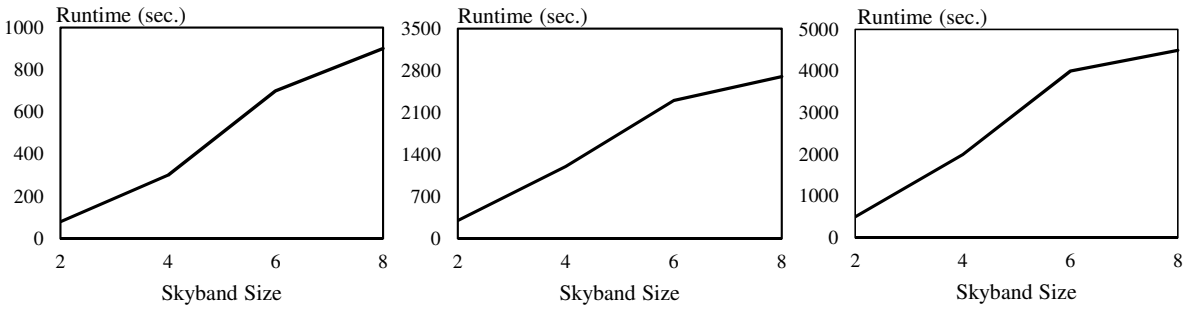



(a) Correlated (b) Independent (c) Anti-correlated

**Figure 9.** Performances for different cardinalities.

### 5.1.3. Effect of Skyband Size

We also examined the efficiency of the proposed technique with various skyband sizes. For this evaluation, we used a data dimensionality $m$ of 6, data cardinality of 100,000, and skyband size $K$ ranging from 2 to 8. Figure 10a–c present the runtime for the different distributions. As indicated by the results, our method was affected by the skyband size. Our experimental results show that the runtime increases dramatically as the skyband size increases. The large cardinality of non-dominating objects causes this.




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

**Figure 10.** Performances for different skyband sizes.

## *5.2. Performance on a Real Dataset*

In this section, we analyze the effectiveness of the proposed algorithm based on the quality of skyband objects retrieved from real-world data. The FUEL dataset contains 24,000 records with six attributes and is related to a marketing campaign for vehicles. For this dataset, we conducted similar experiments to those conducted on the synthetic datasets. For the dimensionality test, we used a data cardinality of 24,000, skyband size of $K = 4$, and dataset dimensionality $m$ varying between 3 and 6. Figure 11a presents the results. Figure 11b presents the performance of the proposed algorithm when varying cardinality from 12,000 to 24,000. The dimensionality $m$ was 5, and the skyband size was $K = 4$. In the final experiment, analyzed the performance with various skyband sizes. We used a data cardinality of 24,000 and dimensionality $m$ of 5. The time consumption is presented in Figure 11c. For all experiments on the FUEL dataset, we observed similar characteristics to those resulting from the synthetic datasets, which confirms the scalability of our algorithm on real datasets.

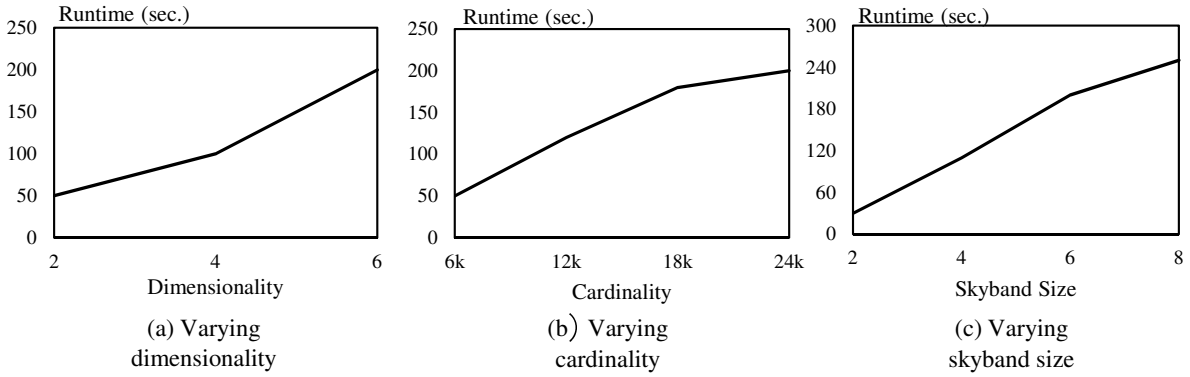



**Figure 11.** Experiments on the FUEL dataset.

### *5.3. Comparison with Existing Works*

In this paper, we have proposed an efficient algorithm for computing the skyband and dominating queries simultaneously by exploiting parallel distributed (MapReduce) computing environment. Therefore, the main focus of the proposed algorithm is to distribute the computation process evenly among distributed computing nodes, so that large data can be processed effectively. On the other hand, both [27,28] considered distributing the computation based on hash mapping of the domain space. Therefore, the data ratio within each grid cell might not be uniform after hash mapping. That means the distribution of computation process among the computing nodes is not balanced. In contrast, our proposed algorithm dynamically distributes the computation process among the computing nodes based on the worst rank of the objects. Therefore, algorithms proposed in [27,28] are incomparable with our algorithm.

## 6. Conclusions

We proposed a parallel distributed algorithm based on the MapReduce framework to solve $K$-skyband queries and top-$k$ dominating queries in this paper. In addition to the scalability provided by the MapReduce, it can perform two different types of queries simultaneously. Extensive experiments demonstrated the efficiency of our algorithm for both real-world and synthetic datasets.

The proposed algorithm can be extended to different directions. Future research should focus on the computation of both types of queries on streaming datasets. In the future, we will research on the potential of an efficient index-based (R-tree/B-tree) parallel distributed method. We also intent to implement our algorithm using spark framework.

**Author Contributions:** M.A.S. and Y.M. conceived the original idea for the study, designed the system model, and analyzed the experiment results. M.A.S. and H.T. performed the experiments and wrote the initial manuscript. M.Q. and Y.M. revised the manuscript. All authors have confirmed and approved the submitted manuscript.

**Funding:** This work is supported by KAKENHI (16K00155, 23500180) Japan. Mahboob Qaosar is supported by the Japanese Government MEXT Scholarship. Md. Anisuzzaman Siddique was supported by the Japanese Government MEXT Scholarship.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Borzsonyi, S.; Kossmann, D.; Stocker, K. The skyline operator. In Proceedings of the IEEE International Conference on Data Engineering (ICDE), Heidelberg, Germany, 2–6 April 2001; pp. 421–430.
2. Yiu, M.L.; Mamoulis, N. Efficient Processing of Top-k Dominating Queries on Multi-Dimensional Data. In Proceedings of the VLDB Endowment, Vienna, Austria, 23–27 September 2007; pp. 483–494.
3. Papadias, D.; Tao, Y.; Fu, G.; Seeger, B. Progressive skyline computation in database systems. *ACM Trans. Database Syst.* **2005**, 41–82. [CrossRef]

4. Chomicki, J.; Godfrey, P.; Gryz, J.; Liang, D. Skyline with Presorting. In Proceedings of the IEEE International Conference on Data Engineering (ICDE), Bangalore, India, 5–8 March 2003; pp. 717–719.
5. Godfrey, P.; Shipley, R.; Gryz, J. Maximal vector computation in large datasets. In Proceedings of the VLDB Endowment, Trondheim, Norway, 30 August–2 September 2005; pp. 229–240.
6. Kossmann, D.; Ramsak, F.; Rost, S. Shooting stars in the sky: An online algorithm for skyline queries. In Proceedings of the VLDB Endowment, Hong Kong, China, 20–23 August 2002; pp. 275–286.
7. Tan, K.L.; Eng, P.K.; Ooi, B.C. Efficient Progressive Skyline Computation. In Proceedings of the VLDB Endowment, Roma, Italy, 11–14 September 2001; pp. 301–310.
8. Yuan, Y.; Lin, X.; Liu, Q.; Wang, W.; Yu, J.X.; Zhang, Q. Efficient computation of the skyline cube. In Proceedings of the VLDB Endowment, Trondheim, Norway, 30 August–2 September 2005; pp. 241–252.
9. Xia, T.; Zhang, D. Refreshing the sky: The compressing skycube with efficient support for frequent updates. In Proceedings of the ACM SIGMOD International conference on Management of Data, Chicago, IL, USA, 27–29 June 2006; pp. 491–502.
10. Chan, C.Y.; Jagadish, H.V.; Tan, K.L.; Tung, A.K.H.; Zhang, Z. On High Dimensional Skylines. In Proceedings of the Extending Database Technology (EDBT) Conference, Munich, Germany, 26–31 March 2006; pp. 478–495.
11. Li, C.; Ooi, B.C.; Tung, A.K.H.; Wang, S. DADA: A Data Cube for Dominant Relationship Analysis. In Proceedings of the ACM SIGMOD International conference on Management of Data, Chicago, IL, USA, 27–29 June 2006; pp. 659–670.
12. Lin, X.; Yuan, Y.; Zhang, Q.; Zhang, Y. Selecting stars: The k most representative skyline operator. In Proceedings of the IEEE International Conference on Data Engineering (ICDE), Istanbul, Turkey, 15–20 April 2007; pp. 86–95.
13. Chan, C.Y.; Jagadish, H.V.; Tan, K.L.; Tung, A.K.H.; Zhang, Z. Finding k-Dominant Skyline in High Dimensional Space. In Proceedings of the ACM SIGMOD International conference on Management of Data, Chicago, IL, USA, 27–29 June 2006; pp. 503–514.
14. Gong, Z.; Sun, G.Z.; Yuan, J.; Zhong, Y. Efficient Top-k Query Algorithms Using K-skyband Partition. In Proceedings of the of Scalable Information Systems, Hong Kong, China, 10–11 June 2009; pp. 288–305.
15. Lin, X.; Yuan, Y.; Wang, W.; Lu, H. Stabbing the sky: Efficient Skyline computation over sliding windows. In Proceedings of the IEEE International Conference on Data Engineering (ICDE), Tokoyo, Japan, 5–8 April 2005; pp. 502–513.
16. Balke, W.T.; Güntzer, U.; Zheng, J.X. Efficient distributed skylining for web information systems. In Proceedings of the Extending Database Technology (EDBT) Conference, Crete, Greece, 14–18 March 2004; pp. 256–273.
17. Tao, Y.; Xiao, X.; Pei, J. Subsky: Efficient Computation of Skylines in Subspaces. In Proceedings of the IEEE International Conference on Data Engineering (ICDE), Atlanta, GA, USA, 3–7 April 2006; p. 65.
18. Papadias, D.; Tao, Y.; Fu, G.; Seeger, B. An optimal and progressive algorithm for skyline queries. In Proceedings of the ACM SIGMOD International conference on Management of Data, San Diego, CA, USA, 9–12 June 2003; pp. 467–478.
19. Dellis, E.; Seeger, B. Efficient Computation of Reverse Skyline Queries. In Proceedings of the VLDB Endowment, Vienna, Austria, 23–27 September 2007; pp. 291–302.
20. Blanas, S.; Patel, J.M.; Ercegovac, V.; Rao, J.; Shekita, E.J.; Tian, Y. A comparison of join algorithms for log processing in MapReduce. In Proceedings of the ACM SIGMOD International Conference on Management of Data, Indianapolis, IN, USA, 6–10 June 2010; pp. 975–986.
21. Jiang, D.; Tung, A.K.H.; Chen, G. MAP-JOIN-REDUCE: Toward Scalable and Efficient Data Analysis on Large Clusters. *IEEE Trans. Knowl. Data Eng.* **2011**, 1299–1311. [CrossRef]
22. Vernica, R.; Carey, M.J.; Li, C. Efficient parallel set-similarity joins using MapReduce. In Proceedings of the ACM SIGMOD International conference on Management of Data, Indianapolis, IN, USA, 6–10 June 2010; pp. 495–506.
23. Apache. Apache Hadoop. Available online: https://hadoop.apache.org (accessed on 3 June 2019).
24. Park, Y.; Min, J.; Shim, K. Parallel Computation of Skyline and Reverse Skyline Queries Using MapReduce. *Proc. VLDB Endow.* **2013**, *6*, 2002–2013. [CrossRef]
25. Tao, Y.; Lin, W.; Xiao, X. Minimal MapReduce Algorithm. In Proceedings of the ACM SIGMOD International Conference on Management of Data, New York, NY, USA, 22–27 June 2013; pp. 529–540.

26. Siddique, M.A.; Tian, H.; Morimoto, Y. k-dominant Skyline Query Computation in MapReduce Environment. *IEICE Trans. Inf. Syst.* **2015**, 1027–1034. [CrossRef]
27. Ezatpoor, P.; Zhan, J.; Wu, J.M.; Chiu, C. Finding Top-*k* Dominance on Incomplete Big Data Using MapReduce Framework. *IEEE Access* **2018**, 7872–7887. [CrossRef]
28. Guidan Chen, Y.W. Top-K Dominating Query Processing over Distributed Data Streams. *Glob. J. Eng. Sci. Res. Manag.* **2018**, *5*, 13–23. [CrossRef]