How to cite this paper:

Gulzar, Y., Alwan, A. A., Salleh, N., & Al Shaikhli, I. F. (2019). Identifying skylines in cloud databases with incomplete data *Journal of Information and Communication Technology, 18* (1), 19-34.

IDENTIFYING SKYLINES IN CLOUD DATABASES WITH INCOMPLETE DATA

Yonis Gulzar, Ali Amer Alwan Aljuboori, Norsaremah Salleh & Imad Fakhri Al Shaikhli
Kulliyyah of Information Communication and Technology
International Islamic University Malaysia, Malaysia

yonis.gulzar@live.iium.edu.my; aliamer@iium.edu.my; norsaremah@iium.edu.my; imadf@iium.edu.my

ABSTRACT

Skyline queries is a rich area of research in the database community. Due to its great benefits, it has been integrated into many database applications including but not limited to personalized recommendation, multi-objective, decision support and decision-making systems. Many variations of skyline technique have been proposed in the literature addressing the issue of handling skyline queries in incomplete database. Nevertheless, these solutions are designed to fit with centralized incomplete database (single access). However, in many real-world database systems, this might not be the case, particularly for a database with a large amount of incomplete data distributed over various remote locations such as cloud databases. It is inadequate to directly apply skyline solutions designed for the centralized incomplete database to work on cloud due to the prohibitive cost. Thus, this paper introduces a new approach called Incomplete-data Cloud Skylines (ICS) aiming at processing skyline queries in cloud databases with incomplete data. This approach emphasizes on reducing the amount of data transfer and

Received: 5 January 2018 Accepted: 3 October 2018 Published:11 December 2018

domination tests during skyline process. It incorporates sorting technique that assists in arranging the data items in a way where dominating data items will be placed at the top of the list helping in eliminate dominated data items. Besides, ICS also employs a filtering technique to prune the dominated data items before applying skyline technique. It comprises a technique named local skyline joiner that helps in reducing the amount of data transfer between datacenters when deriving the final skylines. It limit the amount of data items to be transferred to only those local skylines of each relation. A comprehensive experiment have been performed on both synthetic and real-life datasets, which demonstrate the effectiveness and versatility of our approach in comparison to the current existing approaches. We argue that our approach is practical and can be adopted in many contemporary cloud database systems with incomplete data to process skyline queries.

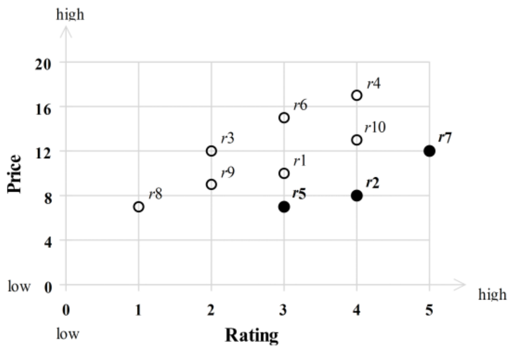
Keywords: Preference queries, query processing, skyline queries, incomplete data, cloud databases.

INTRODUCTION

Skyline queries are one of the predominant preference queries that have received significant attention in database literature. It has been marked as a rich area of research in database community for the recent decade. Skyline process attempts to identify the superior data items, which are not dominated by other data items in the database (Bharuka & Kumar, 2013; Borzsony, Kossmann, & Stocker, 2001; Gulzar, Alwan, Salleh, & Shaikhli, 2017a; Khalefa, Mokbel, & Levandoski, 2008; Swidan, Alwan, Turaev, & Gulzar, 2018). Skyline set comprises a set of non-dominated data items which is named (skylines) in a given database. Given two data items *p* and *q*, it can be said that *p* dominates *q* if and only if *p* is better than *q* in all dimensions and *p* is not worse than *q* in at least one dimension. A running database example (*restaurant*) has been utilized throughout this paper to elaborate the detail process of skyline query. Assuming a database relation named *restaurant,* which consists of two attributes (dimensions) each contain the details of 10 different restaurants. The first dimension indicates the rating of the restaurant that is given by the customers, while the second dimension represents the food price in each restaurant as demonstrated in Figure 1.

ID	Rating	Price
r_1	3	10
r_2	**4**	**8**
r_3	2	12
r_4	4	17
r_5	**3**	**7**
r_6	3	15
r_7	**5**	**12**
r_8	1	7
r_9	2	9
r_{10}	4	13

(a)



(b)

Figure 1. Example of skyline query.

It can also be observed that restaurants r_8 and r_5 their price is the cheapest among all restaurants. Nevertheless, r_8 have the lowest rate compared with other restaurants. Therefore, based on the skyline concept, restaurant r_8 is dominated by restaurant r_5. Similarly, restaurant r_7 has a higher price value compared with the price values of restaurant r_1 and r_9. However, r_7 has the highest rate value compared to all other restaurants. Therefore, r_7 could be one of the potential skyline results as illustrated in Figure 1(a). Applying skyline technique on *restaurant* database would result into only those restaurants that are the cheapest in price and highest in rate. Hence, the skyline result will include r_5, r_2, and r_7 restaurants.

Various strategies have been suggested utilizing the concept of skyline concentrating on different types of databases such as complete, incomplete and uncertain. Some of the previous techniques focused on processing skyline query in the complete database. These techniques aim at reducing the searching space and minimizing the number of pairwise comparisons among the data items through avoiding the unnecessary pairwise companions between data items to identify the skylines. On the other side, other techniques concentrate on developing new solutions taking into account the new challenges introduced by the incompleteness of the data when processing skyline queries. This includes losing the transitivity property of skyline technique due to the missing values, which further leads to the issue of *cyclic dominance.* Hence, applying skyline technique designed for complete a database on an incomplete database is prohibitive and can incur high cost due to the unnecessary exhaustive pairwise comparisons between data items (Alwan et al., 2016; Bharuka & Kumar, 2013; Khalefa et al., 2008). However, these solutions are suggested to fit with centralized database in which database relation is located in one site and only local access is needed to identify the skylines.

In this regard, in cloud database merging of data before applying skyline technique result into transferring unnecessary tremendous amount of data from different remote datacenters. This solution is extremely undesirable as it leads to a prohibitive cost due to transferring a large amount of data, which incur high processing cost. Moreover, it also leads to a large number of unnecessary pairwise comparisons between data items, which can be avoided before applying skyline process. Processing skyline queries for a database with incomplete data in cloud context might not be as easy as in centralized context.

This paper is an extension of the work in (Gulzar et al., 2017b). It presents a new approach, Incomplete-data Cloud Skylines (ICS) for processing skyline queries in cloud database with incomplete data. In this context, database relations are distributed over various locations and remote access needs to be conducted in order to retrieve the skylines of the database. In this work, we assume that the database relations are divided horizontally and spread over different sites. The proposed approach comprises three phases, namely: (i) identifying the skylines of each relation in all datacenters, (ii) joining the skylines of all relations, and (iii) identifying global skylines.

RELATED WORK

A great research effort has been devoted highlighting the problem of skyline queries in database systems. In this section, we examine and report the relevant works of skyline queries in both complete and incomplete databases. Skyline queries have been first introduced into the database community by (Borzsony et al., 2001). They have proposed two different algorithms to process skyline queries in complete database, namely, Block Nested Loop (BNL) and Divide and Conquer (D&C). Later, many algorithms have been developed which are inspired by BNL and D&C techniques by either utilizing the idea of partitioning proposed in D&C or applying the concept of sorting to improve BNL technique. These algorithms are Linear Elimination Sort Skyline (Godfrey et al., 2005), Branch and Bound Skyline (Papadias et al., 2005), and SkyTree (Lee & Hwang, 2014). The main theme of these techniques is to process skyline queries in the centralized complete database.

Many other approaches have been proposed to derive skylines in incomplete databases. The review work in Gulzar et al. (2017a) has summarized skyline query approaches proposed by many researchers in incomplete databases that include BUCKET and Iskyline (Khalefa et al., 2008), Replacement Based Sets Skyline Queries (Arefin & Morimoto, 2012), Baseline, Virtual Point based algorithm, the k-iskyband algorithm (Miao et al., 2013). The work in Alwan et al. (2016) proposed a framework that inspired by the work introduced by Khalefa et al. (2008). In addition, the work in Lee et al. (2016) proposed two algorithms for incomplete data, namely: baseline algorithm called BUCKET and sorting-based bucket skyline algorithm (SOBA). In SOBA two optimized techniques: *bucket level orders* and *point lever orders* have been used to reduce domination tests between data items, minimize the size of skyline set and overall increases the efficiency of skylines processing over incomplete data. Lastly, Wang et al. (2017) has introduced an approach to process skyline queries for massive incomplete data. The main idea of the proposed approach is based on dividing the initial database into two clusters (restrict and loose) according to the importance of the dimension. This followed by applying skyline technique on the dimensions of higher importance. Similarly, skyline technique is applied to the loose dimensions, which have lower importance. Lastly, the skylines of both clusters are compared with each other to return the final skylines.

However, it can be concluded that most of these related works mentioned above assumed that the database is centralized, and data are stored in a single database relation. Nevertheless, there have been several skyline techniques proposed for distributed complete databases such as Sort First Skyline Join (Vlachou et al., 2011), Iterative (Sun et al., 2008), and Skyline Join Algorithm (Zhang et al., 2016). These techniques assume that data is partitioned either vertically or horizontally and might exist in more than one database relation. To evaluate skyline queries, join operator needs to be performed combining the data of the relations before applying the skyline technique. However, it is impractical to directly apply these techniques on incomplete distributed databases due to the prohibitive cost and the issue of cyclic dominance and losing transitivity property of skyline technique.

To the best of our knowledge, the most recent work that raised the issue of processing skyline queries in incomplete distributed databases is contributed by Alwan et al. (2017). They suggested that the database is distributed over more than two relations and these relations are divided horizontally. The proposed technique encompasses three phases, namely: identify the skylines of each relation, joining the skylines of the relations and determining the final skylines. Several optimization techniques have been employed to eliminate those dominated data items before finding the global skyline of all relations. However, this work is limited to the case of database relations which are vertically partitioned where the dimensions (attributes) of the relations are located on different sites and remote access needs to be performed during the skyline process. Besides, the architecture of cloud is quite different from distributed environment.

DEFINITIONS

This section gives some necessary definitions and annotations that are related to skylines queries in a cloud database with partially complete data. These definitions and notations are important to explain the details of our proposed approach. Our technique has been developed in the context of relational databases with partially complete data, *D*. A relation of the database *D* is denoted by *R* (*d*1, *d*2, ..., *dm*) where *R* is the name of the relation with *m*-arity and *d* = (*d*1, *d*2, ..., *dm*) is the set of dimensions.

Definition 1 Incomplete Database: given a database *D* (*R*1, *R*2, ..., *Rn*), where *Ri* is a relation denoted by *Ri* (*d*1, *d*2, ..., *dm*), *D* is said to be *incomplete* if and only if it contains at least a data item *pj* with missing values in one or more dimensions *dk* (attributes); otherwise, it is *complete.*

Definition 2 Dominance: Given two data items p_i and $p_j \in D$ database with *d* dimensions, p_i dominates p_j (the greater is better) (denoted by $pi \succ pj$) if and only if the following condition holds: $\forall\ d_k \in d, p_i.d_k \geq p_j.d_k \wedge \exists d_l, \in d, p_i.d_l > p_j.d_l$.

Definition 3 Skyline Queries: Select a data item *pi* from the set of *D* database if and only if *pi* is as good as *pj* (where $i \neq j$) in all dimensions (attributes) and *strictly* better than p_j *in* at least one dimension (attribute). We use *Sskyline* to denote the set of skyline data items, *Sskyline* = $(pi\ \forall\ pi, pj \in D, pi \succ pj)$.

Definition 4 Comparable: Let the data items *ai* and *aj* ∈ *R*, *ai* and *aj* are comparable (denoted by $ai\ \varepsilon\ aj$) if and only if they have no missing values in at least one identical dimension; otherwise *ai* is incomparable to *aj* (denoted by $a_i \not\varepsilon\, a_j$).

Definition 5 Cloud Database: given a set of databases *D* (*DB*1, *DB*2, ..., *DBn*), where *DB*1 is a database denoted by *DB* (*R*1, *R*2, ..., *Rn*), where *R* represent a database relation belong to *DBi*, *D* is a cloud database if the databases are deployed over different datacenters located on different sites.

METHODOLOGY

The Proposed Incomplete-data Cloud Skylines Approach

In this section, the detail steps of the proposed approach, Incomplete-data Cloud Skylines Approach (ICS) for processing skyline queries in cloud incomplete database are presented and explained. The proposed approach focuses on processing skyline queries with the intention of decreasing the number of pairwise comparisons and the amount of data transferred during skyline evaluation. To achieve this aim, we attempt to ensure that the dominated data items reside in different datacenters are eliminated before applying skyline technique. This will help to avoid many unnecessary pairwise comparisons between data items while holding the transitivity property and avoids the issue of cyclic dominance. The phases of the proposed approach for processing skyline queries in incomplete database in the cloud is elaborated in Figure 2.

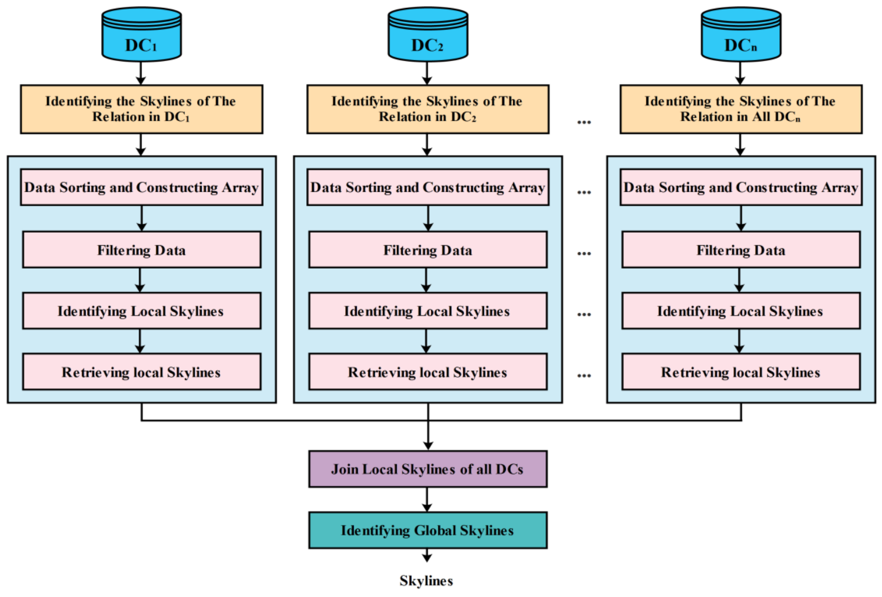

DC₁
DC₂
DCₙ
Identifying the Skylines of The Relation in DC₁
Identifying the Skylines of The Relation in DC₂
Identifying the Skylines of The Relation in All DCₙ
Data Sorting and Constructing Array
Filtering Data
Identifying Local Skylines
Retrieving local Skylines
Join Local Skylines of all DCs
Identifying Global Skylines
Skylines

Figure 2. The phases of the ICS approach

The steps of determining the skylines of each relation include sorting data and constructing array, filtering data, identifying local skylines and retrieving local skylines. Combining the skylines of each relation is performed to identify the skylines candidate. Finally, further comparisons are performed on the combined data items to derive the final global skylines. These phases are explained in detail as the following.

Identifying the Skylines of Each Relation in All Datacenters

The first phase, identifying the skylines of each relation in all datacentres, attempts to identify the skylines of each relation separately, which are located at different datacenters, aiming at discarding all dominated data items from the join operation. Thereby it results in propagating only the most candidate data items into the next phases. This process assists by avoiding joining of dominated data items via performing filtration. That leads in eliminating the unnecessary pairwise comparisons between data items and reduce the amount of data transfer significantly. The detail processes of this phase are elaborated in the following subsections.

Sorting data and constructing array

This step is responsible for analyzing the initial incomplete database relation and attempts to sort the data items based on non-missing dimensions in non-ascending order. Then a set of arrays is constructed and the *id's* of sorted data items are stored in connected arrays. The number of arrays constructed mainly depends on the number of dimensions with no-missing value. This step helps in reducing the searching space, which further leads to decrease the number of pairwise comparisons between data items in the subsequent phases.

Filtering data

This step is one of the most significant phases in introducing the local skylines of each involved table. This phase is responsible for eliminating the dominated data items before applying skyline technique. This is achieved by scanning the whole data items in each array in sequential order using round robin fashion. The scanning process ends when all data items have been read at least once. It might happen that some data items are read more than once. Therefore, a counter is needed to count the number of reading of each data item. The idea behind using the counter is to sort the data items according to their count values in decreasing order. Hence, the data items with the highest count score have a higher potential to be in the skylines set. Besides, it also helps in eliminating a large number of dominated data items. The outputs of this process are a list of data items with their corresponding count values.

Identifying local skyline

In this step, the data points that have no potential to be part of the skyline are eliminated before applying skyline technique. That also helps in reducing unnecessary pairwise comparisons to make the proposed approach more efficient. This eliminating process will be executed by removing all the data points from the list with count score less than two. The rest data points will be stored in candidate set for the further process.

Retrieving local skyline

This step is responsible for the implementation of skyline technique over the data items presented in the candidate set. The aim is to find the local skylines separately from all relations stored in different datacenters at distant locations. This process is conducted in parallel on all datacenters. That helps to reduce the maximum amount of data to be transferred from one data center to another for evaluation of final skylines. The process starts by reading the first data item in the candidate set and then compared with the remaining data items. The read data item named as *processing data item p*, while the data item to be compared with *p* is called *candidate data item q*. During the comparison process if *p* dominates *q* then *q* will be immediately eliminated from the candidate set. Else if neither *p* dominates *q* and nor *q* dominates *p*, then *q* will remain in the candidate set for further processing. However, if *q* dominates *p* then *p* will not be removed immediately; rather it will remain until the end of the iteration process. This is because *p* may have good potential to eliminate other data items and helps to sustain transitivity property and solves the issue of cyclic dominance. This process continues until all remaining data items are processed. It should be noted that no two data items are compared more than once. We argue that this process is effective in avoiding many unnecessary pairwise comparisons between data items. The output of this step is the set of the local skylines of each relation to be joined to form the final skylines.

Joining Skylines of all Relations

This phase intends to combine the identified local skylines of all relations into one relation at one datacenter. It should be noted that the output of this phase is a set of data items with the high potential to be in the skyline set. Thus, many unnecessary pairwise comparisons can be avoided, and only limited number of data items will be propagated into the next phase.

Identifying Global Skylines

This is the last phase of our proposed approach for processing skyline queries in a database with incomplete data over the cloud environment. It tries to determine the final skyline set which contains those data items that are not dominated by other data items in all involved relations. The sub-phases of the first phase (identifying the skylines of each relation) of our proposed approach will be performed on joined local skylines. If the joined data item is not dominated by the other data items in the candidate skyline set, then it is retrieved as part of the final skyline. Otherwise, it is removed from the candidate skyline set. In this process, we guarantee that the final skylines are the skylines of the relations in all cloud datacentres and no other data items might dominate the identified final skylines.

EXPERIMENT SETTINGS

Various experiments have been performed over different synthetic and real datasets to evaluate the performance of the proposed approach, ICS. The ICS approach has been compared with the most recent works: Incoskyline (Alwan et al., 2016) and Sort-based Incomplete Data Skyline (SIDS) (Bharuka & Kumar, 2013). Since skyline technique is a CPU intensive and needs exhaustive pairwise comparisons between data items, therefore, this work concentrates on measuring the efficiency of the proposed approach with respect to the number of pairwise comparisons and amount of data transfer between datacenters. These are considered as the most influenced parameters in processing skyline queries (Alwan et al., 2016; Bharuka & Kumar, 2013; Khalefa et al., 2008; Soliman et al., 2010). The number of pairwise comparisons has been computed with respect to the number of dimensions, and database size. These two metrics are measured by varying the number of dimensions, the number of dimensions with missing values, and the database size. In our experiments, we assumed that the database is fragmented vertically into three database relations situated on three different datacenters and the user has submitted the query into datacenter 1. Two different datasets have been involved in the experiment namely: synthetic (correlated) and real dataset (NBA and MovieLens).

Table 1

The Parameter Setting of Synthetic and Real Datasets

Dataset Name	Parameter Settings			
	No. of dimensions (d)	No. of dimensions with missing values (d')	Dataset Size (KB)	Data Centers
MovieLens	4	3	400, 800, 1200, 1600, 2000	3
NBA	6-18	5-17	40,80,120,160,200	3
Correlated	4-12	3-11	100, 200, 300, 400, 500	3

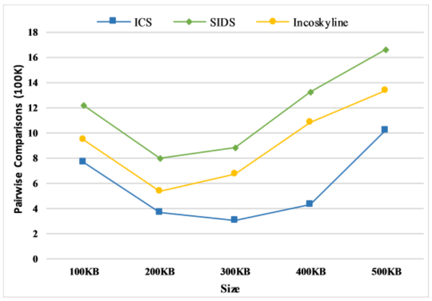
Table 1 summarizes the parameter setting for synthetic and real datasets. Form the table we notice the first real dataset, MovieLens has a total of 4 dimensions and the number of dimensions with missing values is 3. Besides, the size of the dataset ranging between 400-2000KB and the database tables are spread over three different remote datacenters. Similarly, for NBA real dataset, the total number of dimensions varies between 6 – 18 dimensions and the number of dimensions with incomplete data is between 5-17 dimensions. Furthermore, the dataset size varying between 40 – 200 KB and the number of datacenters is 3. Lastly, for the correlated synthetic dataset, the number of dimensions is varies between 4-12 dimensions and the number of dimensions with missing values between 3-11 dimensions. While the dataset size in the range of 100-500KB and the total number of datacenters is 3.

RESULTS AND DISCUSSION

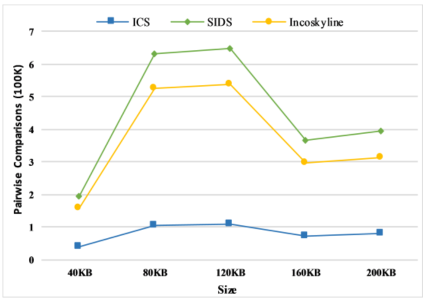
This section presents and discusses the result of the experiments that have been conducted on the synthetic and real datasets. The experiments have been designed with the aim of studying the impact of the dataset size and the number of dimensions on the number of pairwise comparisons and the processing time of each approach. Due to the limited resources and the high cost of constructing a real cloud environment with physical datacenters and other necessary tools, the proposed approach has been tested with a simulated cloud environment. We attempt to represent the cloud environment using some database tables which are distributed over several remote locations and simultaneous run on these database tables using the proposed approach is carried out to identify the local skylines of each database table. Then, the local skylines of each datacenter are further propagated to the next phase using the join operator. Lastly, the final skylines of all involved database tables are retrieved. Extensive experiments have been accomplished and the result discussed below.

Dataset Size

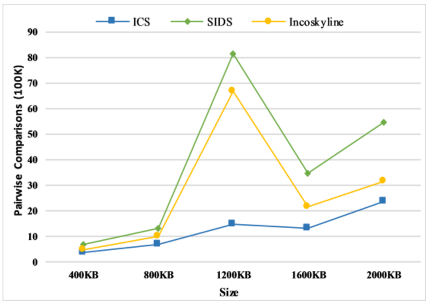
The experiment reports in this section attempt to examine the impact of the database size on processing skyline queries. For this set of experiment, the database size is variable, and the number of dimensions is fixed. Figure 3 illustrates the results obtained from real and synthetic datasets. Figure 3(a) shows the number of pairwise comparisons derived for the correlated dataset. The number of dimensions is 8 and the size of the database is varying from 100KB to 500KB. Figure 3(b) depicts the experiment results of NBA dataset. In this dataset, the number of dimensions is fixed to 18 and dataset size varies between 40KB to 200KB. Figure 3(c) presents the experiment results of the MovieLens dataset where database size varies from 400KB to 2000KB and the number of dimensions is 8. From the results, it is observed that our strategy outperforms Incoskyline and SIDS and database size have no significant impact on the performance of our proposed approach. This is due to applying the process of data filtration and local skyline identifier that helps in reducing the number of pairwise comparisons.



(a) Correlated



(b) NBA

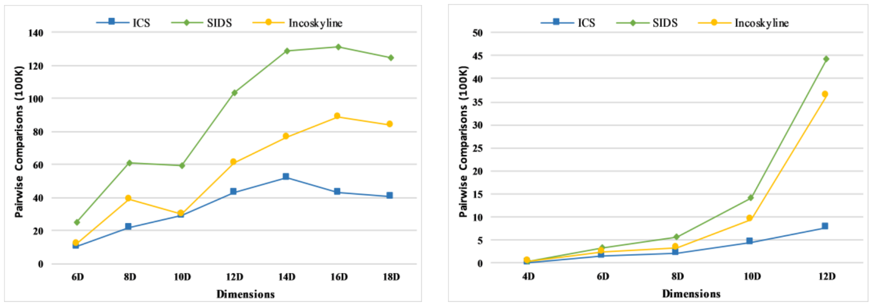


(c) MovieLens

Figure 3. Database size effect.

Number of Dimensions

In this set of experiment, we attempt to investigate the impact of the number of dimensions belongs to the database on the performance of the skyline process. In this experiment, the size of the dataset has been fixed while varying the number of dimensions. Figure 4 depicts the results obtained for both datasets, namely, real and synthetic datasets. Figure 4(a) illustrates the experiment results of the real dataset, NBA where the number of dimensions is varying between 4 to 18 while the dataset size is set to be 120KB. Figure 4(b) describes the experiment results of the synthetic dataset (correlated) in which the number of dimensions is varying between 4 to 12 and dataset size is fixed to 200KB. It can be concluded that our approach introduced a lower number of pairwise comparisons and steadily outperform SIDS and Incoskyline techniques. It is also noticed that increasing number of dimensions has a reasonable impact on the skyline process, which leads to a larger number of pairwise comparison in identifying the skylines. Nevertheless, this increment in the number of dimensions has no significant impact on our proposed approach and the number of pairwise comparisons is marginally increased.

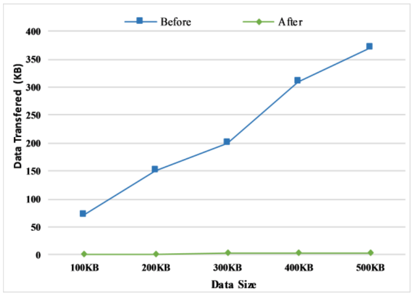


(a) NBA (b) Correlated

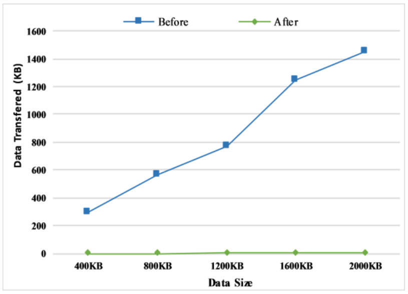
Figure 4. Number of dimensions effect.

Data Transfer

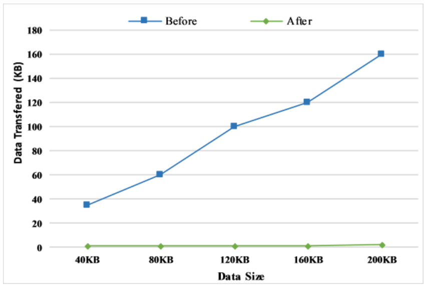
This set of experiments concentrates on examining the impact of the dataset size on the amount of data transfer among datacenters during the skyline operation. The amount of data transfer indicates the total amount of data items that need to be transferred across the cloud datacenters to evaluate the skylines since it influences the performance of the skyline query process in a cloud environment. Figure 5a, 5b, and 5c depict the results of the proposed approach on synthetic (correlated), MovieLens, and NBA datasets, respectively. From the results, we can observe that applying skyline technique to each datacenter separately before transferring the data items is beneficial and leads to great reduces to the amount of data transferred. Amount of data transfer considered as a critical factor for query processing in distributed and cloud environments (Alwan et al., 2017). This is because the lesser amount of data to be transferred the faster the skyline process would be. Hence, transferring the local skylines to query submitted datacenter is far better than transferring all the data from each datacenter. Experiment results showed that we have successfully saved up to 95% to 98% of data from being transferred. That, in turn, saves the network cost.



(a) Correlated



(b) MovieLens



(c) NBA

Figure 5. Amount of data transfer.

CONCLUSIONS

In this paper, the issue of processing skyline queries in incomplete cloud database have been discussed. A new skyline approach called ICS has been proposed to process skyline queries in incomplete cloud databases. The detail steps of each phase of ICS have been explained. We also described how the proposed approach managed to derive the final skylines of the database in a cloud environment. We also showed the significance of using sorting and filtering techniques and how these techniques boost the skyline process. Experiments over different types of datasets have been accomplished to measure the performance of the proposed approach. The results showed that our approach has significantly outperformed the previous techniques (SIDS and Incoskyline) in processing skyline queries in incomplete cloud databases by taking less number of pairwise comparisons and amount of data to be transferred from one datacenter to another. From the results it can be also noticed that dataset size and number of dimensions have insignificant impact on our proposed approach. It can also be noticed that the idea of local skyline joiner has a great impact on reducing the amount of data transfer from one datacenter to another.

ACKNOWLEDGMENT

This research is supported by the project FRGS15-205-0491, Ministry of Higher Education, Malaysia.

REFERENCES

Alwan, A. A., Ibrahim, H., Udzir, N. I., & Sidi, F. (2016). An efficient approach for processing skyline queries in incomplete multidimensional database. *Arabian Journal for Science and Engineering, 41*(8), 2927-2943. doi: 10.1007/s13369-016-2048-z

Alwan, A. A., Ibrahim, H., Udzir, N. I., & Sidi, F. (2017). Processing skyline queries in incomplete distributed databases. *Journal of Intelligent Information Systems, 48*(2), 399-420. doi: 10.1007/s10844-016-0419-2

Arefin, M. S., & Morimoto, Y. (2012). Skyline sets queries for incomplete data. *International Journal of Computer Science & Information Technology, 4*(5), 67-80.

Bharuka, R., & Kumar, P. S. (2013). Finding skylines for incomplete data. *Proceedings of the 24th Australasian Database Conference - Volume 137*. Adelaide, Australia.

Borzsony, S., Kossmann, D., & Stocker, K. (2001). The skyline operator. *Proceedings 17th International Conference on Data Engineering*. Cancun, Mexico.

Godfrey, P., Shipley, R., & Gryz, J. (2005). Maximal vector computation in large data sets. *Proceedings of the 31st International Conference on Very large Data Bases*. Trondheim, Norway.

Gulzar, Y., Alwan, A. A., Salleh, N., & Shaikhli, I. F. A. (2017a). Processing skyline queries in incomplete database: Issues, challenges and future trends. *Journal of Computer Science, 13*(11), 647-658. doi: 10.3844/jcssp.2017.647.658

Gulzar, Y., Alwan, A. A., Salleh, N., & Shaikhli, I. F. A. (2017b). Skyline query processing for incomplete data in cloud environment. Paper presented at the *6th International Conference on Computing & Informatics (ICOCI)*, 24-25 April, Kuala Lumpur, Malaysia.

Khalefa, M. E., Mokbel, M. F., & Levandoski, J. J. (2008, 7-12 April 2008). Skyline query processing for incomplete data. Paper presented at the *IEEE 24th International Conference on Data Engineering*, Cancun, (Mexico). PP. 556-565.

Lee, J., & Hwang, S.-w. (2014). Scalable skyline computation using a balanced pivot selection technique. *Information Systems, 39* (Supplement C), 1-21. doi: 10.1016/j.is.2013.05.005

Lee, J., Im, H., & You, G.-w. (2016). Optimizing skyline queries over incomplete data. *Information Sciences, 361*, 14-28. doi: 10.1016/j.ins.2016.04.048

Miao, X., Gao, Y., Chen, L., Chen, G., Li, Q., & Jiang, T. (2013). On efficient k-skyband query processing over incomplete data. Paper presented at the *18th International Conference on Database Systems for Advanced Applications*, 22-25 Apr,(2013). Wuhan, Chian.

Papadias, D., Tao, Y., Fu, G., & Seeger, B. (2005). Progressive skyline computation in database systems. *ACM Trans. Database Syst., 30*(1), 41-82. doi: 10.1145/1061318.1061320

Soliman, M. A., Ilyas, I. F., & Ben-David, S. (2010). Supporting ranking queries on uncertain and incomplete data. *The VLDB Journal, 19*(4), 477-501. doi: 10.1007/s00778-009-0176-8

Sun, D., Sai, W., Jianzhong, L., & Tung, A. K. H. (2008), Skyline-join in distributed databases. Paper presented at the *IEEE 24th International Conference on Data Engineering Workshop*. Cancun, Mexico

Swidan, M. B., Alwan, A. A., Turaev, S., & Gulzar, Y. (2018). A model for processing skyline queries in crowd-sourced databases. *Indonesian Journal of Electrical Engineering and Computer Science, 10*(2), 798-806. doi:10.11591/ijeecs.v10.i2.pp798-806

Vlachou, A., Doulkeridis, C., & Polyzotis, N. (2011). Skyline query processing over joins. *Proceedings of the ACM SIGMOD International Conference on Management of Data*. Athens, Greece.

Wang, Y., Shi, Z., Wang, J., Sun, L., & Song, B. (2017). Skyline preference query based on massive and incomplete dataset. *IEEE Access, 5*, 3183-3192. doi: 10.1109/ACCESS.2016.2639558

Zhang, J., Lin, Z., Li, B., Wang, W., & Meng, D. (2016). Efficient skyline query over multiple relations. *Procedia Computer Science, 80* (Supplement C), 2211-2215. doi: https://doi.org/10.1016/j.procs.2016.05.381