Accurate Aggregation Query-Result Estimation and Its Efficient Processing on Distributed Key-Value Store

Kosuke Yuki[1], Atsushi Keyaki[1], Jun Miyazaki[1(✉)], and Masahide Nakamura[2]

[1] School of Computing, Tokyo Institute of Technology, Tokyo, Japan
{zhang,keyaki}@lsc.cs.titech.ac.jp, miyazaki@cs.titech.ac.jp
[2] Kobe University, Hyogo, Japan
masa-n@cs.kobe-u.ac.jp

Abstract. We propose four methods for improving the accuracy of aggregation query-result estimation using histograms and/or kernel density estimation and the efficiency of query processing on a distributed key-value store (D-KVS). Recently, aggregation queries have played a key role in analyzing a large amount of multidimensional data generated from sensors, Internet-of-Things devices, etc. A D-KVS is a platform to manage and process such large-scale multidimensional data. However, querying large-scale multidimensional data on a D-KVS sometimes requires a costly data scan owing to its insufficient support for indexes. Since aggregation-query results do not always need to be accurate, our four methods are not only for estimating accurate query results rather than obtaining accurate results by scanning all data, but also improving query-processing performance. We first propose two kernel density estimation-based methods. To further improve query-result estimation accuracy, we combined each of these two methods with a histogram-based scheme so that we can dynamically select an optimal estimation method based on the relationship between a query and the data distribution. We evaluated the efficiency and accuracy of the proposed methods by comparing them with a current method and showed that the proposed methods perform better.

Keywords: Query-result estimation · Data summarization · Histogram · kernel density estimation · Distributed key-value store

1 Introduction

Due to the spread of the Internet and smartphone, large-scale multidimensional data, called big data have recently been collected and it is becoming increasingly important to analyze and use big data. One of the most useful operations that enable such analysis is an aggregation query; however, there are various challenges in computing aggregation queries for large-scale multidimensional data.

C. Ordonez et al. (Eds.): DaWaK 2019, LNCS 11708, pp. 310–320, 2019.
https://doi.org/10.1007/978-3-030-27520-4_22

A key-value store (KVS) [10] is suitable for managing large-scale data. A KVS is a simplified table-type database in which a tuple consists of two attributes: key and value. Because of its simple structure, it is easy to decentralize data over several servers by horizontal partitioning, which is also called a distributed KVS (D-KVS). However, in many cases, D-KVSs support an index only on a key. Therefore, it is difficult to execute flexible and complex queries because of the costs incurred in carrying out a data scan over a large amount of data. There have been several studies on efficiently processing aggregate operation for large-scale multidimensional data on D-KVSs [3,12,18–20]. Watari et al. [19] proposed a method for mutually using a relational database (RDB) [4] and D-KVS. With their method, the data space is split into several hyper-rectangles, which are called *grids*. A partial aggregation value for each grid is computed and stored in the D-KVS. Given a query, scans of the data in grids that are completely included in the query range are omitted because the aggregation values of such grids have already been computed. This optimization reduces the amount of data to be scanned. However, for grids that partially overlap the query range, it will still execute data scans for these grids, which may result in a large number of data scans in some cases.

To omit these costly data scans, we propose four methods for estimating query results, rather than obtaining accurate results, using a histogram and kernel density estimation (KDE). Histograms [7–9,14,15] are known as a lightweight and less accurate estimation scheme, whereas KDE [17] is more accurate and costly. With these estimation schemes, we construct a histogram based on the data in each grid and/or carry out KDE for each bucket of the histogram. At query processing, the value of the query result is estimated using only a pre-computed histogram and KDE results. We also aim at further improvement in estimation accuracy and query throughput with the advantages of this histogram and KDE.

Besides histograms and KDE, there have been many approaches for approximate query processing, such as wavelets [2], cardinality estimation algorithms [6], deterministic algorithms [5]. However, only a few studies refer to the distributed multidimensional approximate query processing.

We evaluated the efficiency of the proposed methods by comparing their query throughput performance with Watari et al.'s method [19] on a cluster machine and showed that the proposed methods perform better. We also compared the query-result estimation accuracy of each of the proposed methods.

2 Related Work

2.1 Partial Pre-Aggregation

Partial pre-aggregation [13,19] is a method of improving aggregation-processing efficiency by reducing the number of data scans. This method splits the database into several blocks and pre-computes aggregation results for each block. When processing aggregate operations, the pre-computed results are reused as much

as possible. Some aggregation queries can be efficiently evaluated with partial pre-aggregation values.

For example, consider an aggregate operation for a relation B to obtain the sum of the *height*s of records satisfying $age < 20$, where B consists of *age* and *height*. Assume that B is split into two blocks B_1 and B_2, each of which has its own partial sum of the *height*s. B_1 and B_2 have only the records that satisfy $age < 15$ and $age \geq 15$, respectively. According to this information, it is not necessary to scan the data in B_1, and only B_2 needs to be scanned.

Watari et al. [19] proposed a method of efficiently processing aggregation queries for large-scale multidimensional data using partial pre-aggregation with a cluster combining an RDB and a D-KVS. With their method, the data space is split into several hyper-rectangles (*grids*) by designating the number of data entries (*grid size*) in each grid. After the split, partial pre-aggregation values for each grid are pre-computed, and the results are stored. At query processing, high efficiency is achieved using the pre-computed partial pre-aggregation values.

Given a query range QR, the aggregation query of the data within QR with their method is processed as follows.

Step 1. Find all grids that intersect a QR by using the metadata of grids in the RDB. Let Gs be a set of the obtained grids. Check if each grid range is completely included in QR.

Step 2. Combine the partial pre-aggregation results of the grids in the Gs that are completely included in the QR. These partial pre-aggregation values can be obtained quickly because they are pre-computed and stored in the D-KVS.

Step 3. Scan all data in the grids in the Gs that partially overlap the QR (hereafter, referred to as *surrounding grids*) and aggregate the values within the QR.

Step 4. Combine the results obtained in Steps 2 and 3.

However, for grids that partially overlap the query range, it will still perform data scans with their method (hereafter, we call this method the *All-Scan* method), which could reduce query throughput.

2.2 Query Processing with Histograms

Poosala et al. proposed a technique called MHIST of dividing the multidimensional data distribution into a histogram [16]. The essential idea of MHIST is to choose the top $p-1$ attributes $A_1 \ldots A_{p-1}$, which belong to the distribution D, whose marginal distributions in D are the most in need of partitioning, and split D into p buckets along $A_1 \ldots A_{p-1}$. This procedure is repeated until the number of buckets reaches the predetermined upper limit.

The meaning of "the most in need of partitioning" varies depending on which histogram is adopted. From the experiment by Poosala et al., the best estimation accuracy was obtained by MHIST adopting the max-diff histogram [15] with 2 splits (hereafter, this method is abbreviated as *MHIST*). The aim of a max-diff histogram is to avoid putting attribute values with vastly different frequencies into the same bucket.

Muralikrishna et al. [11] proposed a scheme for using histograms to estimate the query results called *Uniform Scheme*. In this scheme, the data distribution in each bucket of a histogram is assumed a uniform distribution. Therefore, the estimated result for a given query Q is calculated as the following expression

$$\sum_{B_i \in Q} F_{B_i} \cdot \frac{Volume(B_i \cap Q)}{Volume(B_i)}$$

where F_{B_i} is the frequency of bucket B_i. The validity of this assumption will enhance as the volume of each bucket becomes smaller, regardless of the actual data distribution.

3 Proposed Methods

3.1 Naïve Methods

In addition to pre-aggregation methods [13,19], we introduce our two naïve data summarization methods using histograms and KDE. We store the summarized result (hereafter, *statistical data*) and estimate the query result by using these statistical data to avoid full data scans at query processing. The processes of both data summarization as preprocessing and query-result estimation using statistical data are described as follows.

Step1. Divide the data space into several grids using the All-Scan method and store the aggregated values for each grid. This step in Fig. 1 shows that the entire data space is divided into six grids.

Step2. Construct a histogram for each grid using MHIST and save these results. This step in Fig. 1 means a histogram was constructed based on the data that grid 000 contains.

Step3. In case of use KDE, which is used based on the data in each bucket of the histogram constructed at Step 2 and save the results. This step in Fig. 1 is for the result of KDE based on the data in one of the buckets of the histogram from grid 000.

Given a query Q, we use pre-aggregated values for the grids that are completely included in the query range. This part follows the All-Scan method. On the other hand, for the surrounding grids, we use statistical data obtained from MHIST and KDE to estimate aggregation results (Fig. 2).

We now discuss our two naïve methods for estimating aggregation results using statistical data and compare them with a previous method.

1. Previous method: MHIST
   First, we consider a previous study that only used MHIST and Uniform Scheme in Sect. 2.2 for estimation. We call this method *MHIST*.

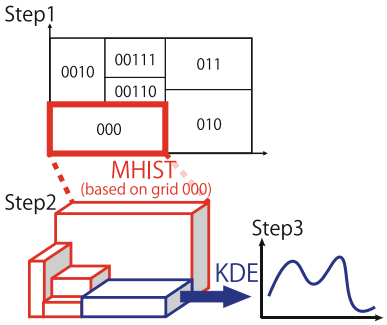

Fig. 1. Process of data summarization

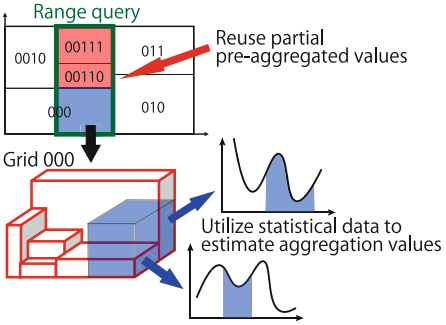

Fig. 2. Process of query-result estimation

2. Proposed naïve method 1: All-KDE
   We considered obtaining more detail summarized data by using KDE to achieve more accurate estimation than MHIST since KDE can be used to understand the data distribution in more detail. We call this method *All-KDE*.
   After histograms are constructed using MHIST, KDE is carried out for data distribution in each bucket and store several points from estimated distribution. Therefore, $Sum(B_i, Q)$ is calculated as the following expression, where $K_{ij} \in Q$ means that the j-th estimated point in B_i is included in Q.

$$Sum(B_i, Q) = \sum_{K_{ij} \in Q} K_{ij}$$

3. Proposed naïve method 2: Part-KDE
   Though All-KDE can be used to understand the distribution in more detail, the amount of the data obtained with KDE becomes large and this could cause a decrease in query throughput. As a result of the histogram constructed with MHIST, in some cases, there are several buckets that do not have a domain range for a certain dimension (the minimum and maximum values of the attribute's domain are the same). We refer to such a bucket as a *dimension diminished bucket*.
   We assume that the data distribution in the bucket can be regarded as a uniform distribution for a dimension diminished bucket; thus, KDE is not carried out. For normal buckets, KDE is carried out in the same manner as with All-KDE. Given a Q, if the dimension diminished bucket satisfies Q, the estimated value is calculated using Uniform Scheme in the same manner as with MHIST, and for normal buckets that satisfy Q, precomputed estimated values obtained with KDE are used.
   Therefore, $Sum(B_i, Q)$ is expressed as follows.

$$Sum(B_i, Q) = \begin{cases} F_{B_i} \cdot \frac{Volume(B_i \cap Q)}{Volume(B_i)} & \text{(if } B_i \text{ is a dimension diminished bucket)} \\ \sum_{K_{ij} \in Q} K_{ij} & \text{(otherwise)} \end{cases}$$

By modifying All-KDE to carry out KDE only for normal buckets, the data size of the statistical data reduces. Therefore, it is possible to achieve higher query throughput than with All-KDE and prevent deterioration in estimation accuracy. We call this method *Part-KDE*.

By combining histogram and KDE, it is expected that All-KDE and Part-KDE can improve the estimation accuracy.

3.2 Hybrid Methods

In this section, we focus on the relationship between the overlap rate OR_{ij} of query Q_i to a surrounding grid G_{ij} and the estimation error in each surrounding grid. We aim to further improve estimation accuracy by dynamically selecting the estimation method based on OR_{ij}.

Before explaining our further methods, we introduce the overlap rate and the error index. The overlap rate OR_{ij} between Q_i and one of the surrounding grids of Q_i, G_{ij}, is defined as follows.

$$OR_{ij} = \frac{Volume(G_{ij} \cap Q_i)}{Volume(G_{ij})}$$

On the other hand, the error index EI_{ij} for grid G_{ij} is defined as

$$EI_{ij} = \frac{Error\_of\_G_{ij}}{TrueValue\_of\_Q_i}$$

This error index means how much the error of the individual grid affects that of the entire query result compared with its true value. The error index of a grid is higher as its error to the entire query result becomes greater.

We introduce our two hybrid methods which dynamically select an estimation method applied to each grid to improve estimation accuracy and query throughput. This idea is based on the assumption that the best estimation method changes according to the overlap rate.

By combining MHIST and All-KDE (hereafter, *MA*), it is expected that both estimation accuracy and query throughput would improve because MHIST is the lightest among the naïve estimation methods. We also consider combining MHIST and Part-KDE (hereafter, *MP*) to enhance throughput, because Part-KDE has less overhead than All-KDE.

4 Experimental Evaluations

The experimental evaluations were conducted on a cluster with 13 PCs running HBase 1.2.0 (D-KVS) and PostgreSQL 9.6.1 (RDB) with the following two datasets.

Extended Indoor Sensor Data: Based on 2 million real data entries collected from indoor environmental sensors, each of which data entry has 16 attributes, we generated pseudo dataset by replicating the data, resulting in 100 million data entries. We call such data *extended indoor sensor data (EIS Data)*. Four-dimensional range queries are carried out for EIS data.

San Francisco Bay Area Data: We generated 22 million points of moving objects in the San Francisco Bay Area using a network-based generator [1]. Each data entry has two attributes, latitude and longitude. In addition, we assumed each data entry as a running car and gave it car speed as the third attribute. We call such data *San Francisco bay area data (SFB Data)*. Two-dimensional range queries are carried out for SFB data.

Some variables were configured for the experiments as follows.

- Maximum number of data entries in a grid (grid size): 1000
- Upper limit of the number of buckets with MHIST: 25
- Kernel function for KDE: Gaussian

4.1 Preliminary Experiment

We measured OR_{ij} and EI_{ij} for the datasets and randomly generated aggregation range queries, and plotted it. For these plots, we drew trendlines to clarify the relationship between the overlap rate and the error index for each estimation method.

The results are shown in Figs. 3 and 4. From these results, we observed that there are transition points at which the error index was inverted. For range-sum query, the transition point for MHIST and All-KDE was 1.8% of the overlap rate, and 3.7% for MHIST and Part-KDE. For range-count, that for MHIST and All-KDE was 49.4%, and 85.6% for MHIST and Part-KDE. When the overlap rate is less than these values, the error index of MHIST is the best. In other words, it is possible to optimize the estimation accuracy and query throughput by switching estimation methods based on the overlap rate. Although not shown here, SFB Data case also has the similar transition points.

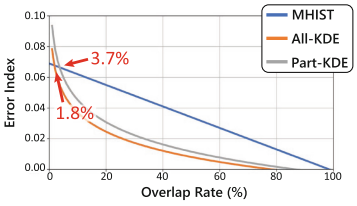

Fig. 3. EIS Data (Sum)

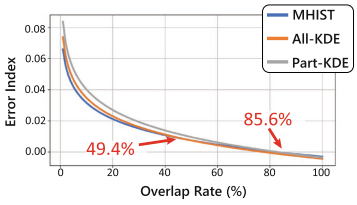

Fig. 4. EIS Data (Count)

4.2 Evaluation of Estimation Accuracy

We compared the estimation accuracy of the hybrid methods using the transition points obtained by the preliminary experiment to that of the naïve methods. We conducted randomly generated multidimensional range-sum and range-count queries, so that their selectivity can be set as 0.001% and 10%. We measured the error rate of each query for both the entire range and only the surrounding grids.

Figures 5 and 6 show the average error rates of each method when changing each selectivity for EIS data. Figure 5 indicates that MA achieved higher estimation accuracy than the naïve methods at all selectivities. The accuracy of the surrounding grids at 0.001% selectivity markedly improved because there tend to be many grids with low OR_{ij}. However, Fig. 6 shows that MA did not always lead to the best estimation. We concluded that this result was caused by the high/low relationship of the trendlines of the count queries (Fig. 4). The high/low relationship of the trendlines of the count queries was not as clear as the trendlines of the sum queries (Fig. 3). Therefore, MHIST could be inferior to the other two naïve methods even at OR_{ij} of less than 49.4%. Though MA is not always the best, its accuracy degradation is insignificant from the viewpoint of the error rate of the entire query result. On the other hand, MP was inferior to MA in many cases, but its accuracy did not deteriorate much compared with that of MA.

Also, we observed the same tendency for SFB data (Figs. 7 and 8). For both of the sum and count queries at all selectivities, MA achieved the best estimation accuracy. As for MP, we confirmed the improvement of estimation accuracy compared with Part-KDE.

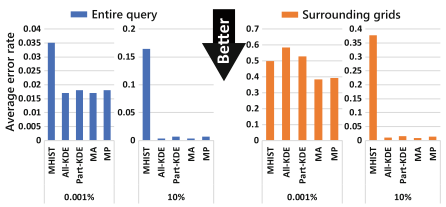

Fig. 5. EIS Data (Sum)

Fig. 6. EIS Data (Count)

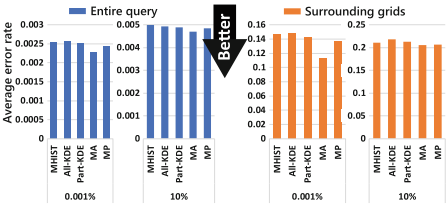

Fig. 7. SFB Data (Sum)

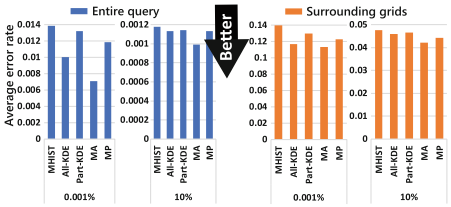

Fig. 8. SFB Data (Count)

4.3 Evaluation of Query Throughput

We also evaluated the query throughput performance for the four proposed methods and others (All-Scan and MHIST). We ran the same queries under the same conditions as mentioned in Sect. 4.2. The queries were issued from 1, 16, 32, 64, and 128 clients simultaneously while varying selectivity.

Figures 9, 10, 11, 12 depict the results of query throughput performance for each dataset and query. As an overall view, these results indicate that the hybrid methods achieved higher throughput than All-KDE and Part-KDE. In addition, Figs. 11 and 12 show that MA can achieve higher throughputs than Part-KDE when the number of clients increased, and MP can achieve high throughputs,

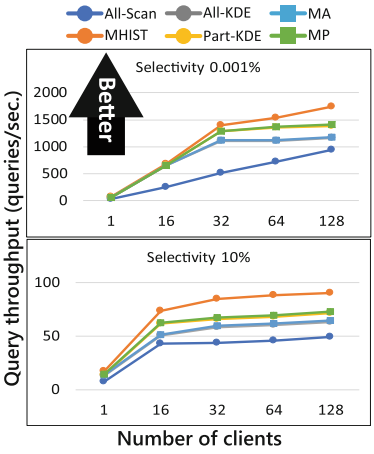

Fig. 9. EIS Data (Sum)

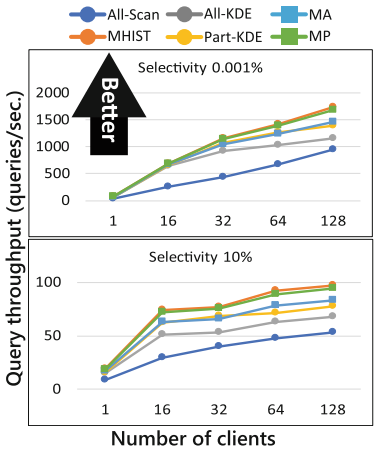

Fig. 10. EIS Data (Count)

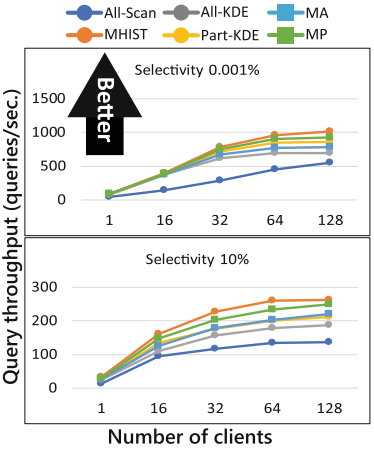

Fig. 11. SFB Data (Sum)

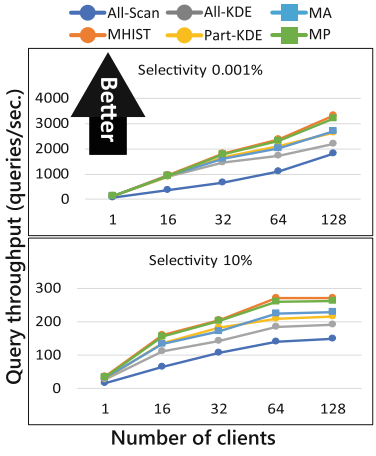

Fig. 12. SFB Data (Count)

which is comparable to MHIST. This is because MHIST is adapted at almost half or higher OR_{ij}.

In summary, MA can achieve highly accurate and efficient estimation. In contrast, MP can provide much higher query throughput than MA with small accuracy deterioration.

5 Conclusion

We proposed four methods for highly efficient and accurate estimation of aggregation queries on large-scale multidimensional data on a cluster of an RDB and a D-KVS. In particular, MA and MP dynamically select estimation methods among MHIST, All-KDE, and Part-KDE to adapt the most accurate estimation method based on the overlap rate. MA and MP can also achieve higher query throughput through the partial use of MHIST.

For future work, we will consider a method for automatically finding transition points to switch estimation methods for arbitrary data.

Acknowledgments. This work was partly supported by JSPS KAKENHI Grant Numbers 18H03242, 18H03342, and 19H01138.

References

1. Brinkhoff, T.: A framework for generating network-based moving objects. GeoInform. **6**(2), 153–180 (2002)
2. Chakrabarti, K., Garofalakis, M.N., Rastogi, R., Shim, K.: Approximate query processing using wavelets. Proc. VLDB **2000**, 111–122 (2000)
3. Eldawy, A., Mokbel, M.F.: Spatialhadoop: a mapreduce framework for spatial data. Proc. of IEEE ICDE **2015**, 1352–1363 (2015)
4. Garcia-Molina, H., Ullman, J.D., Widom, J.: Database Systems: The Complete Book. Prentice Hall, New Jersey (2002)
5. Han, X., Wang, B., Li, J., Gao, H.: Efficiently processing deterministic approximate aggregation query on massive data. Knowl. Inf. Syst. **57**(2), 437–473 (2018)
6. Heule, S., Nunkesser, M., Hall, A.: Hyperloglog in practice: algorithmic engineering of a state of the art cardinality estimation algorithm. In: Proceedings of EDBT 2013. pp. 683–692. ACM (2013)
7. Ioannidis, Y.: The history of histograms (abridged). Proc. VLDB **2003**, 19–30 (2003)
8. Jagadish, H.V., Koudas, N., Muthukrishnan, S., Poosala, V., Sevcik, K.C., Suel, T.: Optimal histograms with quality guarantees. In: Proceedings of VLDB 1998, pp. 275–286 (1998)
9. Kooi, R.P.: The Optimization of Queries in Relational Databases. Ph.D. thesis (1980)
10. Lakshman, A., Malik, P.: Cassandra: a decentralized structured storage system. ACM SIGOPS Operating Syst. Rev. **44**(2), 35–40 (2010)
11. Muralikrishna, M., DeWitt, D.J.: Equi-depth multidimensional histograms. In: Proceedings of ACM SIGMOD 1988. pp. 28–36 (1988)

12. Nishimura, S., Agrawal, S.D.D., Abbadi, A.E.: MD-HBase: design and implementation of an elastic data infrastructure for cloud-scale location services. Distrib. Parallel Databases **31**(2), 289–319 (2013)
13. Papadias, D., Kalnis, P., Zhang, J., Tao, Y.: Efficient OLAP operations in spatial data warehouses. In: Jensen, C.S., Schneider, M., Seeger, B., Tsotras, V.J. (eds.) SSTD 2001. LNCS, vol. 2121, pp. 443–459. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-47724-1_23
14. Piatetsky-Shapiro, G., Connell, C.: Accurate estimation of the number of tuples satisfying a condition. In: Proceedings of ACM SIGMOD 1984, pp. 256–276 (1984)
15. Poosala, V., Haas, P.J., Ioannidis, Y.E., Shekita, E.J.: Improved histograms for selectivity estimation of range predicates. In: Proceedings of ACM SIGMOD 1996, pp. 294–305 (1996)
16. Poosala, V., Ioannidis, Y.E.: Selectivity estimation without the attribute value independence assumption. In: Proceedings of VLDB 1997, pp. 486–495 (1997)
17. Silverman, B.W.: Density Estimation for Statistics and Data Analysis. No. 26 in Monographs on Statistics and Applied Probability. CRC Press (1986)
18. Wang, J., Wu, S., Gao, H., Li, J., Ooi, B.C.: Indexing multi-dimensional data in a cloud system. Proc. ACM SIGMOD **2010**, 591–602 (2010)
19. Watari, Y., Keyaki, A., Miyazaki, J., Nakamura, M.: Efficient Aggregation Query Processing for Large-Scale Multidimensional Data by Combining RDB and KVS. In: Hartmann, S., Ma, H., Hameurlain, A., Pernul, G., Wagner, R.R. (eds.) DEXA 2018. LNCS, vol. 11029, pp. 134–149. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98809-2_9
20. Zhang, X., Ai, J., Wang, Z., Lu, J., Meng, X.: An efficient multi-dimensional index for cloud data management. In: Proceedings of CloudDB 2009, pp. 17–24. ACM (2009)