Key-Value Store "MD-HBase" Enables Multi-Dimensional Range Queries

NISHIMURA Shoji

Abstract

Recent years have witnessed a significant increase in the collection and analysis of a large amount of data such as website logs, sensor information from various devices, etc. Relational databases used to be a major system for storing such data, but recently KVS (Key-Value Store) is becoming more popular as a data storage method due to its superior characteristics in scaling out according to increases in data volume. However, KVS only support simple search functions. This paper introduces "MD-HBase," which is an extended version of HBase, a KVS-type database. MD-HBase enables efficient data search performance for multi-dimensional range queries without giving up scalability characteristics.

Keywords

key value store, multi-dimensional range query, HBase, scale-out

1. Introduction

Recent technological advances allow us not only to collect a large amount of data but also to use it for optimal purposes. For example, it is now possible for location-based service (LBS) providers to provide more detailed information to their users by collecting their location information via their smartphones. LBS providers can send their users various information such as recommending nearby shops, distributing shop coupons related to shop promotion events, etc. When carrying out such services, database systems for storing location information are expected not only to collect information from thousands of devices but also to glean information such as a list of the users who are located nearest to a specific shop. A database system handling a large amount of data has to be equipped with a practical search processing ability along with an ability to scale out according to increasing data size.

Relational database systems providing general-purpose search functions used to be popular in the market. With traditional database systems, a scale-up strategy, in which hardware is replaced by more powerful hardware, was a major countermeasure to maintain scalability and meet customers' needs. This strategy makes it possible to increase system power without a drastic change in architecture, but it cannot be expected to increase system power beyond the limit of hardware performance. Moreover, this strategy runs up costs. In order to sort out such issues, a scale-out strategy at one time gathered the market's attention because it could increase system power by adding servers. However, a scale-out strategy requires modifying system architecture according to increases in data size. This means that if the database system has already been modified into a complicated architecture to achieve efficient search function performance for general purposes, the scale-out strategy cannot be employed further for such a system. The market now focuses on a database method, KVS (Key Value Store), that has a simple architecture and easy scale-out characteristics. However, KVS still has some disadvantages. One is that it can only support simple search operations due to its simple architecture. In the era of big data, database systems need to process complicated search functions efficiently while maintaining scalability. This paper introduces "MD-HBase," a KVS-type database system that enables efficient multi-dimensional range search.

2. Ordered Key Value Store (KVS)

KVS is a database model that uses "key" and "value" pairs to efficiently identify records. Ordered KVS is one KVS-type database model. It stores records in ascending order by sorting keys (**Fig. 1**). Ordered KVS is often likened to a dictionary, in that it sorts indexes (keys) in ascending order and these indexes are associated to contents (values).

When an ordered KVS needs to process too many records, it achieves scale-out by dividing tables at the appropriate key

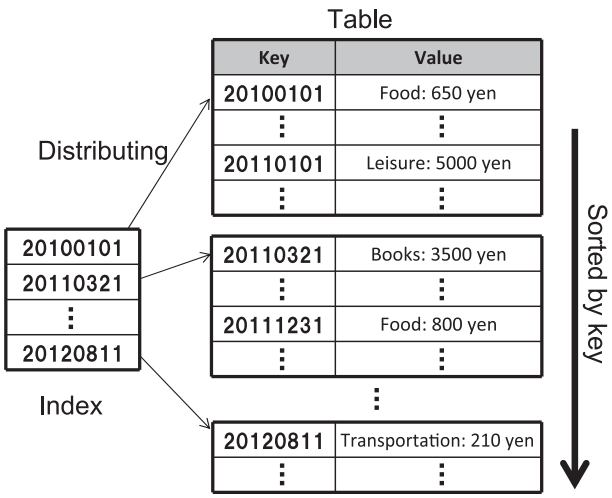

Fig. 1 Ordered KVS.

ranges and distributing these divided tables to other servers. This processing is just like dividing a dictionary by classifying it under index words and splitting the dictionary into small booklets. Distributing records may create multiple tables, so indexes are required to manage which table should be located at which position.

An ordered KVS provides two search functions, "match query" and "range query." A match query extracts the value associated with a certain key. This is the same action as searching for an item using an index word. A range query extracts all the records within a range that covers an area between a specified key and another specified key. Fig. 1 shows an example in which all the records are extracted from the range of indexes between 20110101 and 20111231. The ordered KVS has a feature that sorts records in ascending order so that this extraction operation can be performed very efficiently. Moreover, the range query searches the attributes of a record associated with a key. This search can be considered as a 1D (one-dimensional) range query.

3. Multi–dimensional Range Queries

Let us explain multi-dimensional range queries using an application employed for location-based services. Consider a shop that wants to disseminate coupons to attract customers who are currently located near the shop. A list of customers who should receive the coupon is generated by searching for customers within a certain range of latitude and longitude around the shop. This means that it needs to detect targets with a range query using two attributes, latitude and longitude. Such processing is called a multi-dimensional range query when it specifies a range that includes two or more attributes.

Achieving Multi–dimensional Range Queries with Traditional KVS

In this chapter, we introduce an example of a multi-dimensional range query achieved with an ordered KVS. An ordered KVS sorts its keys in ascending order, making it possible to efficiently execute a range query to extract the attributes associated with a key. However, in our example, there are multiple keys (latitude and longitude) to be selected, making it therefore necessary to somehow group these attributes into a single attribute. This process can be considered as transforming points (latitude and longitude) located in multi-dimensional space into a point (key) located in one-dimensional space. A space-filling curve is one of the most popular approaches for achieving this transformation.

A space-filling curve visits all points in a multi-dimensional space with a single continuous line (one-dimensional curve). A Z-order curve is one such space-filling curve and an example using a Z-order curve applied to 2D space is shown in **Fig. 2** . A line drawn in Z-order visits all the points in a 2D space without overlapping, like drawing the letter "Z." The number in each block shows the order in which the Z-order curve visits. This means that all the points in a multi-dimensional space can be numbered according to the order in which the space-filling curve visits.

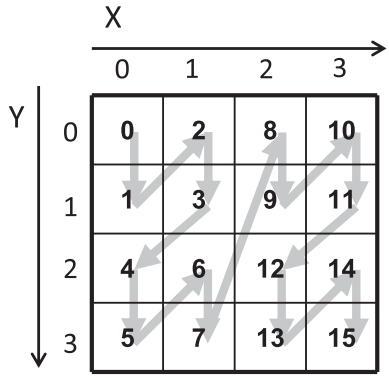

Fig. 2 Space-filling curve.

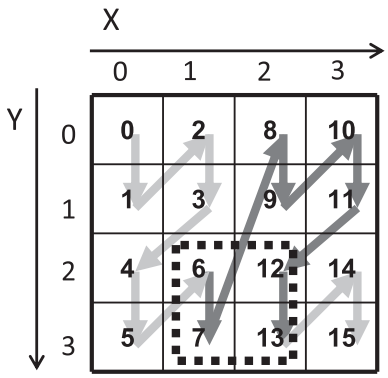

Fig. 3 Range query in the region X=[1, 2], Y=[2, 3] (Z-order curve in 2D space).

When storing multi-dimensional values in a KVS, each attribute pair is calculated to find its order on the space-filling curve and the corresponding number is employed as a key. For example, if the value of X is 1 and the value of Y is 2, the key will be 6. Now let's generate a key using the above-mentioned method and try a multi-dimensional range query for the record stored in the ordered KVS.

Fig. 3 is an example of carrying out a range query in the region of the dotted line. (The range of X is [1, 2] and the range of Y is [2, 3].) First, find the minimum and maximum values for the keys in the queried region. In our example, the minimum key value is 6 and the maximum key value is 13. Next, a range query between minimum value 6 and maximum value 13 is executed in the KVS. At this time, all the blocks visited by a dark-colored arrow (see Fig. 3) are scanned so that the records between key 8 and key 11, which are located outside the queried range, are also included in the search result. In order to avoid such false positive scans, the dimensional attributes of each key are checked to see whether they are inside the boundary or not, so that only records matching the requested condition are extracted to be calculated as the query result.

The method described above easily achieves multi-dimensional range queries in an ordered KVS; however, keys 8 to 11 are also scanned even though they are out of the queried range. This results in degraded search performance.

4. MD–HBase

MD-HBase [1] achieves efficient multi-dimensional range queries by modifying Hbase [2] , an ordered key-value store model. MD-HBase employs a method to efficiently access data in multi-dimensional space to reduce the number of false positive scans and improve the performance of multi-dimensional range queries.

4.1 Multi–dimensional Index Layer and Space–filling Curve

A multi-dimensional index is a method to improve data access performance by structuring data in a systematic manner. MD-HBase leverages the Kd-tree space-partitioning method over the ordered KVS to achieve efficient multi-dimensional range queries. A Kd-tree is a space-partitioning data structure and is one of the possible space-splitting methods for multi-dimensional indexes.

Fig. 4 describes a space-partitioning method using a Kd-tree structure. A Kd-tree splits a space in half once the number of data points in the space reach a certain number. In this example, the threshold value of data points to execute space partitioning is set to two. In Fig. 4(a), three data points are located in the 2D space, which means that the number of data points is beyond the threshold value set for space partitioning.

The Kd-tree then splits the 2D space in half at the middle of the X dimension. This results in the data point locations shown in Fig. 4(b). Let's insert more data points. The result is shown in Fig. 4(c). You can see that the number of data points on the right-hand side is beyond the threshold of space partitioning. So the right-hand side is split in half (upper and lower sections) at the middle of the Y dimension. The result is shown in Fig. 4(d) and the number of data points in the space is maintained at two or less. This means that spaces that contain more data points will be split into smaller spaces to maintain the number of data points at no more than the threshold value set for each space.

Let's combine the results of the space-partitioning method using a Kd-tree and the space-filling curve (described in chapter 3). The result is shown in **Fig. 5** . The continuous space-filling curve traces all split spaces without skipping any space

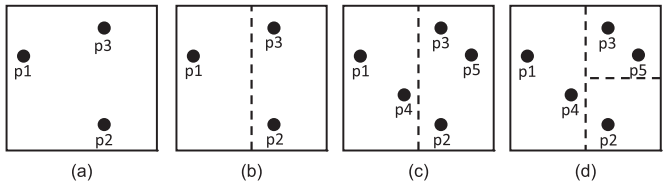

Fig. 4 Space partitioning using the Kd-tree method.

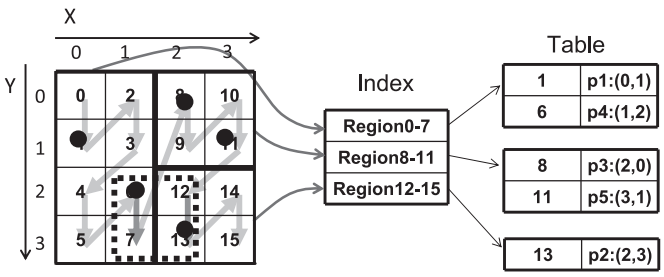

Fig. 5 MD-HBase space partitioning and index layer.

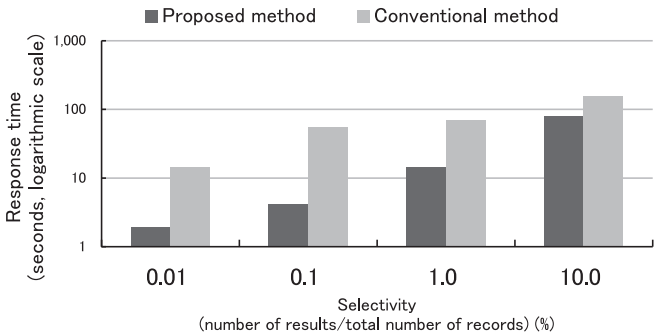

Fig. 6 Multi-dimensional range query results.

blocks. This proves that the Kd-tree space-partitioning method has the property of being able to split data points without changing the order given by the space-filling curve.

MD-HBase divides the table and creates indexes for the divided tables by using this property of the Kd-tree. Data points located in multi-dimensional space are mapped onto one-dimensional space using a space-filling curve, the Z-order curve. MD-HBase uses the numbers where the data points are located as keys to store records in ordered key-value stores. Then, if the number of records exceeds the threshold value set for each table, the table will be split according to the space-partitioning method described above. As shown in Fig. 5, MD-HBase lays out the split spaces in the order of the Z-order curve and maintains a function for referring tables to use them as indexes.

4.2 MD–HBase Multi–dimensional Range Queries

Multi-dimensional range queries using MD-HBase are described in this chapter using the example of scanning the same range shown in Fig. 3. In this example, as shown in Fig. 5, let's assume that space partitioning has been done and indexes have been structured. As with what we did in the previous example, find the minimum and maximum values allocated to the keys in the queried range. In the previous example, the records located between the minimum and maximum key values were simply scanned as they were; however, MD-HBase first checks the index to acquire the space partitioning list for the range between these minimum and maximum values. In this example, three split spaces are detected. MD-HBase checks the boundaries of each dimension of split spaces to determine whether the split spaces overlap with the query range. If the split spaces do not overlap with the query range, this means that none of the records located in these split spaces exist in the query range. MD-HBase can therefore skip scanning records for tables associated with these subspaces.

This example shows that Region 0-7 and Region 12-15 are search range targets, but Region 8-11 is not overlapped. This means that scanning the table corresponding to Region 8-11 can be skipped completely and safely.

With the example in this paper, we only skip scanning one table, but this function will contribute more effectively in cases of searching neighboring domains with high-density data or in cases with more dimensions. In the performance evaluation test of multi-dimensional range queries for a data set simulating vehicle travel, MD-HBase demonstrated a high-speed processing capability approximately ten times faster than when only a space-filling curve is applied (**Fig. 6**). Fig. 6 shows the evaluation results acquired from a range query over 400 million generated data points by simulating vehicles traveling on a road map. The horizontal axis indicates the size of the query range.

5. Conclusion

This paper introduces MD-HBase, a KVS database that achieves efficient multi-dimensional range queries, features scalability according to data size and meets the requirements of the upcoming era of big data processing. The results of this research are planned to be incorporated into the "WebOTX Batch Server," an application execution platform which promotes efficient batch processing. We are pleased to mention that MD-HBase is the result of cooperative studies with Profs. Divyakant Agrawal, Amr El Abbadi and Dr. Sudipto Das of the Department of Computer Science, University of California, Santa Barbara.

Key-Value Store “MD-HBase” Enables Multi-Dimensional Range Queries

References

1) Shoji Nishimura, et. al, “MD-HBase: A Scalable Multi-dimensional Data Infrastructure for Location Aware Services,” International Conference on Mobile Data Management 2011
http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6068416
2) Apache HBase Project
http://hbase.apache.org/

Author's Profile

NISHIMURA Shoji
Assistant Manager
Cloud System Research Laboratories
Central Research Laboratories

Information about the NEC Technical Journal

Thank you for reading the paper.
If you are interested in the NEC Technical Journal, you can also read other papers on our website.

Link to NEC Technical Journal website

Japanese

English

Vol.7 No.2 Big Data

Vol.7 No.2
September, 2012

Special Issue TOP