Volume 8, No. 5, May-June 2017
ISSN No. 0976-5697
International Journal of Advanced Research in Computer Science
REVIEW ARTICLE
Available Online at www.ijarcs.info

Partitioning Techniques in Cloud Data Storage: Review Paper

Kiranjit Kaur
Research Scholar
Guru Kashi University
Talwandi Sabo, Punjab, India

Vijay Laxmi
Dean and Professor UCCA
Guru Kashi University
Talwandi Sabo, Punjab, India

Abstract: As there is large amount of data generated from various web applications which are difficult to manage in cloud databases. Solution to this problem is to partition the data. In this paper three techniques named Horizontal, vertical and workload driven partitioning are reviewed. The main focus of the paper is to compare these techniques on the bases of complexity, scalability, consistency and number of distributed transactions. It provides result, based on that we can choose the most relevant partitioning technique to store cloud data.

Keywords: Cloud database, horizontal partitioning, vertical partitioning, workload driven partitioning, complexity, scalability, consistency, distributed transactions

I. INTRODUCTION

Cloud computing is a technology which provide us various services free of cost. One of the main services is that it provides is the facility to store data in a cloud, which can be accessed anywhere, anytime with World Wide Web (WWW). It not only reduces the cost and but also reduce the risk of data loss.

In this review paper, the major concerned is on cloud data storage and the partitioning techniques used for it. In the early days, cloud data was stored on Traditional Database Management System. With the advancement in technology, Data generated from web applications like Facebook, Twitter and etc. become bigger and bigger. It was difficult for traditional database systems to manage this data so new data store named NoSQL was developed. To further increase the performance of these data stores, partitioning was used, which improves various factors like scalability, efficiency and availability
There were different partitioning techniques available:

A. Horizontal Partitioning
B. Vertical Partitioning
C. Workload Driven Partitioning

A. Horizontal Partitioning

In this technique, data was partitioned horizontally and then these partitions stored on different machines. This technique follow the principle of static partitioning means the partitions once formed they cannot change. They remain same forever. Some of the common partitioning techniques were Range, Hashing, Schema and Graph Partitioning.

Range partitioning, partition the cloud data by using range of keys. The value of these keys should be adjacent but they should not be overlapped. Range of keys was decided on the basis of some conditions or operators.
Hash Partitioning use circle or ring to represent the dataset. The ring was divided into number of available node and each node was mapped to a point in the ring. Hash function was used for this mapping.

Schema Partitioning was basically designed to minimize the distributed transactions. In this database schema was partitioned in such a way that related rows kept in the same partition instead of separating them in different partitions. Graph partitioning was workload- based static partition in which partition were made by analysing the pattern of data. But once partitioning was done, it never changes. Means in starting this technique generates partition on the basis of workload of cloud data. But after that it never repartitions the storage and never observed the changes in workload.

B. Vertical partitioning

In this technique, data was partitioned vertically. It was also called column partitioning where set of columns was stored on different segments and distributing them accordingly. In this no two critical columns stored together which improve security of data.

C. Workload Driven Partitioning

In this technique, data generated from the web application was used. It analysis the data access pattern of web application and form partitions according to that. This improves the scalability of transactions in terms of throughput and response time.

II. LITERATURE REVIEW

Researchers have proposed a variety of systems and partition techniques to provide scalable transaction support for web applications. Some of them have been listed here:
C Curino et al. (2010) proposes workload based static partitioning algorithm which was based on graph partitioning. In this K-way min-cut graph partition algorithm was applied to improve the throughput and reduce the number of distributed transactions. The rows, which are accessed in a transaction, are kept on one partition to reduce the distributed transactions [1].

Y. Zhao et al. (2012) propose partition based data storage model. This paper describes three algorithms about partition and storage of data, about data query and connection and about fragment updation and re-partitioning. In this model more number of fragments may occurs which can decrease the performance and re-partition may results in lower efficiency of traversal query [2].

K. Grolinger et al. (2013) describe different partitioning techniques used by NoSQL data stores to achieve scalability. Their names were range partitioning and consistent hashing. Range partitioning stores the partitioned data on different servers based on ranges of a partition key. Data storage and read/write was handled by servers with specific range of keys. In this technique adjacent keys reside on same node so range queries were processed effectively. The problem with this technique is that it can lead load balancing issues. BerkeleyDB, Cassandra, HBase and MongoDB cloud data stores that implement range partitioning. In consistent hashing, nodes were placed in a ring. The ring was divided into ranges which were equal to the available nodes. Voldemort, Riak, DynamoDB, CouchDB, VoltDB and Clustrix cloud data stores that implement consistent hashing [3].

S. Das et al. (2013) propose schema level partitioning. It was a static partitioning scheme which was designed to improve the scalability of ElasTras. It's named Schema level partitioning because it was derived from the TPC-C schema. In the schema level, data partitioning was based on the partitioning key. The related rows of table were located on a single partition which minimizes the distributed transactions [4].

S. Phansalkar et al. (2014) presents transaction-aware partitioning; it used vertical partitioning for improving scalability. The main aim of the author was to decrease response time and optimized cost of processing [5].

L. Chaple et al. (2016) implements workload driven approach based on MongoDB. It was also designed for improving scalability. The approach was implementation by experimenting on MongoDB cloud data store. This approach generates less number of distributed transactions and improves scalability too [6].

J. Kohler et al. (2016) propose vertical partitioning for cloud data stores. It stores sets of columns into different segments and distributing them accordingly. For example, vertical partitioning segments contain predefined groups of columns. So data stores from the column-family category can provide vertical partitioning in addition to horizontal partitioning. One thing that was important in this is that no two critical attributes of a relation was stored in a single partition. It helps to improve privacy and security of data [7].

III. COMPARISION FACTORS

In this part analysis is done on different partitioning techniques used by different data stores based on important parameters like complexity, scalability, consistency and number of distributed transactions.

A. Complexity

This factor determines the implementation level of a partitioning technique. Some techniques are easy to implement and some are difficult ones. It is measured in terms of low/moderate/high.

B. Scalability

In this paper scalability measurement is performed in terms of throughput and response time. A technique is highly scalable if it optimized the throughput and response time both. This factor helps to achieve high performance. It is measured in terms of low/moderate/high.

C. Consistency

When there are multiple copies of the data in data store. To maintain that data consistency is required means all copies of the data must be same and updated at the same time. Consistency is measured in terms of strongly consistent/consistent/configurable/eventually consistent.

D. Distributed Transactions

In cloud storage data is distributed on different servers. Different partitioning techniques are used to manage this data. Number of distributed transactions occurs when this data is accesed. These transactions should be less to achieve efficiency.

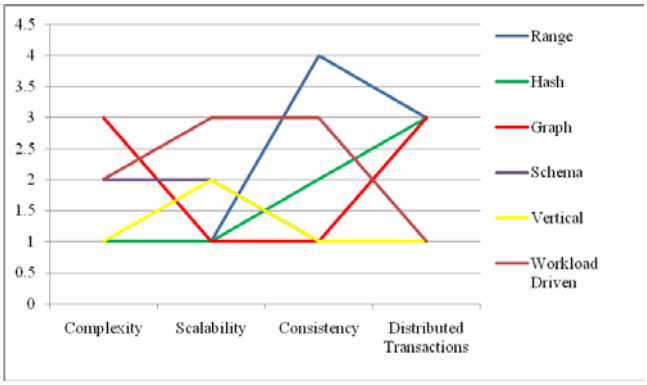
IV. GRAPHICAL ANALYSIS

In this paper comparison of different partitioning techniques is done on the bases of different important parameters.

Table 1: Show values of parameters in different partitions

Partition/ Data	Complexity	Scalability	Consistency	Distributed Transactions
Range	1	1	4	3
Hash	1	1	2	3
Graph	3	1	1	3
Schema	2	2	1	1
Vertical	1	2	1	1
Workload Driven	2	3	3	1

In complexity, scalability and distributed transactions number representation is 1-Low/Less, 2-Medium/Moderate, 3–High/More. Where consistency is measured 1-Eventually Consistent, 2-Configurable, 3-Consistent, 4-Strongly Consistent. As per values the chart is drawn to identify the relation between different partitions.



*A. **Complexity***

Range and hash partitions are simple and easy to use. These techniques were the most popular ones because of simplicity and easy implementation. Range partitioning was used in BerkeleyDB, HBase [8], MongoDB [9] and etc. Hash partitioning is used in Riak, DynamoDB [10], VoltDB [11], CouchDB [12], Clustrix [13] and many other data stores. Graph partitioning is complex and difficult to implement. Data store that implement this partition is HyperGraphDB. Schema partition is neither easy nor difficult; it's in the moderate level of complexity. It is implemented on ElasTrans data store which is an elastic, scalable and self managing transactional cloud data store.

In vertical partitioning, data is divided into columns which is simple and easy to do. It is implemented on SimpleDB, HBase and other cloud data stores. Workload Based partitioning is based on the analysis of data access patterns and then make partitions according to that. Web generated data is analysed continuously to reform the partitions after some interval. Its complexity level is moderate. It is implemented on many NoSQL cloud data stores e.g. MongoDB.

*B. **Scalability***

Here Scalability refers to measurement of partitioning performance in terms of throughput and response time. Range, Hash, Graph have low scalability whereas Schema and Vertical are moderate and workload based approach is highly scalable.

*C. **Consistency***

Range and workload based partitioning are consistent. Graph, schema and vertical partitions are eventually consistent whereas Hash partition is configurable means they can be configured to maintain consistency.

*D. **Distributed Transactions***

As the number of distributed transactions increase, the efficiency of partitioning techniques decreased. Range, Hash and Graph partitions generate more number of distributed transactions. In Schema, vertical and workload based partitions distributed transactions are less which makes them more efficient.

V. CONCLUSION

In this review paper, we have discussed various partitioning techniques used in cloud data storage and compared them. We gave critical review on complexity, scalability, consistency and distributed transactions parameters of different partitioning techniques.

We found that Range partitioning is used where ease of use and strong consistency is required. Vertical partition is used to reduce the number of distributed transactions. Workload based partitioning technique is the suitable choice where scalability, consistency and less number of distributed transactions are main considerations.

VI. REFRENCES

[1] C. Curino et al., "Schism: A workload driven approach to database Replication and Partitioning", in Proc.36th international conference on VLDB, pp. 48-57, Sept. 2010.

[2] Y. Zhao and Y. Wang, "Partition-based cloud data storage and processing model" IEEE 2nd International Conference on Cloud Computing and Intelligent Systems, vol.1, pp. 218-223, 2012.

[3] K. Grolinger et al., "Data Management in cloud environments: NoSQL and NewSQL data stores", Springer Open Journal: Journal of Cloud Computing: Advances, Systems and Applications, December 18, 2013.

[4] S. Das, D. Agrawal and A. E. Abbadi, "ElasTrans: An Elastic, Scalable and self managing transactional database for the cloud", ACM Transactions on Database Systems (TODS), vol. 38, no. 1, pp. 5:1-5:45, April 2013.

[5] S. Phansalkar and A. Dani, "Transaction aware Vertical partitioning of Databases (tavpd) for responsive oltp applications in cloud data stores", Journal of Theoretical and Applied Information Technology, vol. 59, no. 1, January 10, 2014.

[6] L. Chaple and S. Ahirrao, "Scalable transactions using MongoDB cloud data store", Journal of Theoretical and Applied Information Technology, vol. 89,no. 1, July 15, 2016.

[7] J. Kohler et al., "On the performance of Query Rewriting in Vertically Distributed Cloud Databases", Springer: Innovative Approaches and Solutions in Advanced Intelligent Systems, vol. 648, pp. 59-73, April 30, 2016.

[8] "ApacheHBase: Apache HBase Reference Guide", Available: https://hbase.apache.org/book.html

[9] "MongoDB", Available: https://docs.mongodb.com/

[10] Decandia et al. , "Dynamo: Amazon's highly available key value store", Proceedings of the 21st ACM Symposium on Operating System Principles, ACM, New York, pp 205-220,2007.

[11] "VoltDB: VoltDB technical overview", Available: http://voltdb.com/overview.

[12] "Apache CouchDB", Available: http://couchdb.apache.org/.

[13] "Clustrix", Available: http://www.clustrix.com.