I J C T A, 10(8), 2017, pp. 535-541 ISSN: 0974-5572
© International Science Press

A Survey of Different Data Partitioning Techniques

Aakanksha Jumle* and Swati Ahirrao**

ABSTRACT

Cloud computing offers services like online shopping, videos, images, gaming, and storage of files and so on, which can be used anytime and anywhere. It is referred as the platform to deploy scalable and highly available internet application at low cost which serves resources on demand and pay per use form. An effective data partitioning is one of the main requirements for consistent and scalable data as well as to yield good performance. The aim of the data partitioning techniques is to enhance the response time of the web application taken by the transaction. Online transaction system needs to be fast to fulfill customer requirements. The intent of this paper is to perceive different techniques of partitioning proposed to manage big data.

Keywords: Data partitioning, Scalability, Performance, TPC-C benchmark, Distributed transaction

1. INTRODUCTION

In this era, there is tremendous widening of data due to *transfer, storage, sharing of structured and unstructured data* which inundates to business. E-commerce sites and application produce huge and complex data which is referred as Big Data. It is evolving term that describes large amount of *unstructured, semi-structured and structured data*. The cloud computing gives the stable platform for precise, economical and efficient management of data in order to process and store it. The performance of the application increases by combining the advantages of cloud computing and big data.

Due to the wide use of the internet, there is a quick growth of data. In order to handle and store these huge data, a large database is needed. To cope up with large scale data management system (DBMS) would not support the system. Hence the NoSQL [1] data stores get introduced which are characterized by their high scalability, high availability and its consistency. NoSQL database is frequently used in Big Data and web application. It supports different attributes in a collection.

Data management is the main aspect nowadays, if not defined well can lead to data loss or misuse. The data management plays a major role to retrieve the quality of the data. Partitioning of big tables into smaller parts will improve manageability of a database. It is designed in such a way that data needed for a transaction is collected and kept on a partition. Partitioning will improve the response time. Data partitioning has two types:

- Horizontal data partitioning [1-11]
- Vertical data partitioning [12]

In horizontal partitioning, different rows are involved into different tables wherein vertical partitioning, columns are involved and for storing remaining columns additional tables are used.

There are two basic approaches for scaling:

* Symbiosis Institute of Technology, Symbiosis International University, Lavale, Pune, India, *Email: aakanksha.jumle@sitpune.edu.in*

** Symbiosis Institute of Technology, Symbiosis International University, Lavale, Pune, India, *Email: swatia@sitpune.edu.in*

- Scale up
- Scale out

In terms of scaling up, additional resources are added to a single node itself. Another approach is scale out (horizontal scaling) in which more nodes are added to a system. Scale up is not cost effective while scale out is preferred for improving the scalability of applications.

2. RELATED WORK

Researchers have proposed different ways of methods for improving the scalability based on Scaling out strategy. For the betterment of scalability, partitioning is considered as one of the foremost method. Cloud data stores is a NoSQL database for the application which is fast, highly scalable, provides eventual consistency, flexibility for storing the data and ease of query processing.

2.1. ElasTras: An Elastic, Scalable, and Self Managing Transactional Database for the Cloud

The author Sudipto Das introduced ElasTraS [2] which describes Schema level partitioning for achieving scalability. The motivation behind schema level partitioning is to collect related data into the same partition, as the transactions only access the data which is needed from a large database.

The accepted example for Schema level partitioning is “Tree schema” where one table acts as the root table and its primary key acts as a foreign key for other tables. And the primary key of the root table partitioning is used. The experiments of ElasTraS were evaluated by clustering of machines under Elastic Compute Cloud (EC2) infrastructure in Amazon Web Services, using TPC-C benchmark to measure the performance.

TPC-C benchmark is used to demonstrate Schema level partitioning and evaluates the performance of the system for the TPC-C workload. The TPC-C schema comprises of nine different tables, with the Warehouse table as the root table having primary key as w_{id}, and all tables (except the Item table) have these w_{id} as the foreign key, which enables the database to be partitioned using a foreign key.

The design of ElasTraS gives the assistance to two classes of the database

1) Large databases is divided into a small database and are placed across a set of servers
2) A huge number of small and independent databases can be commonly used.

A major goal of ElasTraS is to have elasticity and also to reduce cost operation of the system during failure.

2.2. Schism: A Workload-Driven Approach to Database Replication and Partitioning

The author Cralo Curino has introduced, Schism: A Workload-Driven Approach to Database Replication and Partitioning [3] to improve the scalability of shared nothing distributed databases. It intends to reduce the distributed transactions while making balanced partitions. Graph partitioning technique is used in this paper for transactional workloads. Data items which are accessed in graph partitioning by the transactions are kept on a single partition. It is static level partitioning, where this partition once formed cannot be changed and are put together on one partition. An example should be considered in which there is a single account in bank database table with five tuples, and a workload of four transactions. Each tuple is represented as a node in the graph and edges which are connected to different tuples, are used within the same transaction. Schism introduces an extension of the basic graph representation that captures the opportunity for tuple level replication. A single tuple is represented by exploding the node into a star-shaped configuration of n + 1 nodes where n represents a number of transactions that access the tuple.

2.3. Megastore: Providing Scalable, Highly Available Storage For Interactive Services

Jason Barker, is the author of Megastore [4] which is the repository for storing the required information of newly interactive online services. The author has used Google BigTable supporting arbitrary read & write throughput. It defines that the data is partitioned into various entity groups, each independently and synchronously cloned over many servers with acceptable latency. Related data items are collected as entity groups and placed under a single node. Local indexing of an entity group follows ACID(Atomicity, Consistency, Isolation, and Durability) properties. Megastore aims to make the system to have:

1) Fast read where the local reads give better utilization
2) Fast writes which optimizes the pre-preparing used by master based approaches.

2.4. Automatic Data Distribution In Large-Scale OLTP Applications

The author Xiaoyan Wang has presented, Automatic Data Distribution in Large -scale OLTP Applications [5] because of new challenges of big data which have emerged and the expansion of the workload of the requested data. The distribution is done by the analysis of the requested data. It uses DaWN and comes up with an architecture of ADDS for large OLTP transactions for distribution.

DaWN is a triangle model which stands for Data Workload and Nodes which encodes data partitioning, data placement, and workload processing. ADDS stands for Automatic Data Distribution Solution which contains data partitioning and data placements. Data Partitioning has two phases:

- Initial partitioning which is done on original data
- Online partitioning which is done on incremental data

For original data, initial partitioning is done using BEA (Bond Energy Algorithm) and for incremental data, online partitioning will be invoked where partitions are formed on the base of kNN (k-Nearest Neighbor) clustering algorithm. Data placements allocate these data to the partitions by genetic algorithm.

2.5. Workload-Aware Table Splitting For NoSQL

The author Francisco Cruz presents Table Splitting Technique [1] which considers the system workload. The data is organized into tables. These tables are partitioned into groups of tuples called regions and these regions are spread across the cluster of nodes. The region splitting procedure is done manually which means human explicitly chooses the splitting points or partitioning the regions into two halves. But these approaches for splitting the regions is unfeasible and user needs to gather the information about data access pattern of each region to find out the correct splitting point, which consumes more time. Hence automated mechanism is used to find the region splitting point that will lead to optimal load balancing.

A valuable splitting point is the point that split the region into two new regions with likely loads. The split key search algorithm satisfies the above statement. The algorithm estimates the splitting point when it receives the key from the first request of each region. For each request, if the split key differs then algorithm changes the splitting point. Data is stored into tables in HBase to form regions.

Algorithm:

```
1. Assuming the lowest key is infinity and the highest key is zero.
2. Key = First request grab for each region
3. Key is compared with the lowest key and the highest key
   If key < current estimated key,
           then key is decreased
```

If key > current estimated key,
then key is increased

4. For each request lowest key and highest key is updated accordingly

2.6. Relational Cloud: A Database-As-A-Service For The Cloud

The author Carlo Curino has proposed DBaaS [6] that is Database-as-a-Service called as Relational Cloud for improvising the scalability. There were difficult tasks that invoked the design of Relational Cloud:

- Efficient multi-tenancy to reduce the hardware footprint required for an estimated workload
- Elastic scale-out for handling growing workloads
- Database privacy

In back-end for query processing and storing, relational cloud uses existing DBMS engine. Set of queries is defined on different tables which do not get mixed in this cloud. In front-end, access pattern is monitored which is generated by the workload and which uses this information for partitioning each database.

This paper has overcome the challenges by applying a novel resource estimation and nonlinear optimization-based consolidation technique for multi-tenancy. Graph-based partitioning method is used to spread large databases across many machines for scalability. The notion of adjustable privacy showing the use of different levels of encryption layered can enable SQL queries to be processed over encrypted data.

2.7. Dynamic Workload-Based Partitioning Algorithms For Continuously Growing Databases

The author Miguel Liroz-Gistau [7] has proposed a different way of dynamic partitioning technique called DynPart and DynPartGroup algorithm for efficient data partitioning for upcoming and continuous data elements. It offers constant execution time and does not concern about the database size. The problem with static partitioning is that each time a new set of data arrives and the partitioning is redone from scratch.

DynPart works over the collection of new data items. Due to this many operations can be performed over this set of n items instead of performing it on a single item for n times.

Algorithm of DynPart:

1. Finds set of well-suited queries with the data item.
2. For each query, relevant fragments of query are found
3. Initially affinity = 0,
   Compute affinity by data and fragment
4. Calculation of feasibility of fragment
5. Highest affinity fragment having more capacity of data items is selected as feasibility fragment.

DynPartGroup first creates the groups and arranges them in descending order that is the biggest group is considered and operation is carried on it because of availability of space on fragments and there are higher chances of fitting these groups.

Algorithm of DynPartGroup

1. Create groups and place in descending order
2. Affinity value is calculated for each group
   If group is feasible

```
        then group does not break the imbalance factor
    Else
        group is divided into two equal halves
 3. These two halves are inserted back in the list and are considered individually
```

For experimental results, they have used the data from the Sloan Digital Sky Survey catalog, Data Release 8 (DR8).

2.8. Skew-Aware Automatic Database Partitioning In Shared-Nothing, Parallel OLTP Systems

The author Andrew Pavlo presented [8] a new approach for partitioning a database automatically in a *shared nothing* architecture where each node is independent of sharing the memory. The key contribution of this paper are:

1) Partitioning of database takes place automatically depending on Large Neighborhood search (LNS) technique
2) For a sample workload a new analytical cost model is introduced that estimates the coordination cost and load distribution.

Horticulture, a scalable design tool depends on parallel OLTP systems helps to select the database design for stored procedure. Horticulture analyses a database schema, anatomy of the applications of stored procedures and an element of transaction workload. Then it automatically generates partitioning strategies that minimizes distribution overhead while balancing access skew. LNS technique is used to search probable model off-line for optimization. It analyzes the potential solution with cost model that measures the performance of DBMS using a peculiar design for the sample workload trace. The design of Horticulture specifies that it first analyzes the workload to generate the best design. Secondly is to relax the design which is a subset of the best design and perform the search. If the new design has low cost than best design then select that relax design as best design and repeat it again for deriving the best design.

2.9. Horizontal Cloud Database Partitioning With Data Mining Techniques

The authors Brian Sauer and Wei Hao have presented [9] a different way of data partition using the data mining techniques. It is the methodology for NoSQL database partitioning which depends on data clustering of database log files. They have compared the results of an average response time of algorithms. Firstly for clustering the data, input given to them is the log that is generated and which contains database transaction. The log which is generated in this experiment is list of IP addresses, the requested page, and the query used on that page. After generation of log IP addresses, they were changed to random IP addresses from that region. Matching records are deleted and are converted into a vector format and loaded into the k-means algorithm for forming non-overlapping clusters. Clusters are labeled and this information is maintained in a table to display the number of queries from each region in each cluster and stored in MongoDB database.

The new algorithm has been built to overcome k-means issue that is the detection of oddly shaped data, by using minimum spanning tree which is effective than k-means. The phases are:

1) To give the input as vector format data and the graph is constructed.
2) Minimum spanning tree is constructed, for easy implementation and efficiency Prim’s algorithm was selected.
3) Cutting down the edges. The edge is compared with the average standard deviation and if greater, then remove that particular edge.
4) By using breadth first search, clusters are traversed and as well as labeled.

2.10. Scalable Transactions In Cloud Data Stores

The authors Ahirrao and R. Ingle presented scalable workload-driven partitioning scheme [10] in which data access patterns are introduced. In this partitioning technique the transaction logs and the data access patterns are interpreted, where transactional logs refer to the history of actions made in the database. Data access pattern shows that if one warehouse is out of stock then the requirement is fulfilled by different warehouse in the different partition. The partitioning is initiated referring the data access pattern that which warehouse is frequently accessing other warehouse and according to that, the partitioning takes place. TPC-C benchmark is used for evaluating the performance by implementing scalable transaction. The experiments are carried on HBase and Amazon SimpleDB cloud data stores.

2.11. Automated Data Partitioning For Highly Scalable And Strongly Consistent Transactions

Author Alexandru Turcuhas presented automated data partitioning [11] which is done for both single partition and independent transaction which tends for good partition. He has implemented the Granola rules where data is stored in main memory. Granola model tries to minimize the synchronization overhead by executing all transactions in single partition.

Transactions are used for operating those data.

- Firstly the partitioning performs static analysis of data and byte-code rewriting in which collection of data dependency information, abstract information of operation which is to be performed are given a unique tagged for associations.
- Secondly, representative of tagged operations is collected.
- Thirdly graph is represented which shows the workload trace of transaction where objects as nodes and transactions as edges follow the Schism [3] prototype with edges having weights.

3. CONCLUSION

The paper describes different techniques and parameters related to data partitioning used in the large database. It specifies different ways of partitioning algorithms. The purpose of this review is to study the techniques of horizontal partitioning like range partitioning, schema level partitioning, graph level partitioning etc. giving the high performance of data. The vital aim of these techniques is to reduce distributed transaction, improving the response time and throughput. Some of the techniques mentioned above faced difficulty in scaling of the graph representation, handling of heavy workload, selection of partition to provide data for the particular transaction and so on.

REFERENCES

[1] Francisco Cruz, Francisco Maia, Rui Oliveira and Ricardo Vila¸ca. Workload-aware table splitting for NoSQL. SAC'14 March 24–28, 2014, Gyeongju, Korea Copyright 2014 ACM 978- 1-4503-2469-4/14/03.

[2] Das S, Agrawal D, El Abbadi A (2013) ElasTraS: An elastic, scalable, and self-managing transactional database for the cloud. ACM Trans DatabaseSyst (TODS) 38(Article 5):1–45.

[3] Curino C, Jones E, Zhang Y, Madden S (2010) Schism: a workload-driven approach to database replication and partitioning. Proc VLDBEndowment 3(1–2):48–57.

[4] Baker J, Bond C, Corbett J, Furman JJ, Khorlin A, Larson J, L´eon J-M, Li Y,Lloyd A, Yushprakh V(2011) Megastore: Providing scalable, highly available storage for interactive services. In: CIDR, Volume 11. pp 223–234.

[5] Xiaoyan Wang, Xu Fan, Jinchuan Chen and Xiaoyong Du. Automatic Data Distribution in Large-scale OLTP Applications. International Journal of Database Theory and Application Volume.7, No.4 (2014), pp. 37-46.

[6] Curino C, Jones EPC, Popa RA, Malviya N, Wu E, Madden S, Zeldovich N(2011) Relational cloud: A database-as-a-service for the cloud. In: Proceedings of the 5th Biennial Conference on Innovative Data Systems Research. pp 235–240.

[7] Miguel Liroz-Gistau, Reza Akbarinia, Esther Pacitti, Fabio Porto and Patrick Valduriez. Dynamic Workload-Based Partitioning Algorithms for Continuously Growing Databases. Springer-Verlag Berlin Heidelberg 2013.

[8] Andrew Pavlo, Carlo Curinoand Stan Zdonik. Skew-Aware Automatic Database Partitioning in Shared-Nothing, Parallel OLTP Systems. SIGMOD'12, May 20–24, 2012.

[9] Brian Sauer and Wei Hao. Horizontal Cloud Database Partitioning With Data Mining Techniques. 2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC).

[10] S. Ahirrao and R. Ingle, "Scalable Transactions in Cloud Data Stores," Journal of Cloud Computing: Advances, Systems and Applications (2015) 4:21 DOI 10.1186/s13677-015- 0047-3.

[11] AlexandruTurcu,RobertoPalmieri,BinoyRavindra,SachinHirve,Automated Data Partitioning for Highly scalable and strongly consistent transactions, IEEE transactions on parallel and distributed systems,Volume-27,January 2016.

[12] S. Phansalkar and Dr. A. Dani, "Transaction Aware Vertical Partitioning Of Database (TAVPD) For Responsive OLTP Applications In Cloud Data Stores," Journal of Theoretical and Applied Information Technology, Volume. 59 No.1, January 2014.