International Journal of Advances in Applied Sciences (IJAAS)
Vol. 7, No. 1, March 2018, pp. 21~28
ISSN: 2252-8814, DOI: 10.11591/ijaas.v7.i1.pp21-28

Data Partitioning in Mongo DB with Cloud

Aakanksha Jumle, Swati Ahirrao
Computer Science Symbiosis Institute of Technology, Lavale, Pune, India

Article Info

Article history:

Received May 23, 2017
Revised Dec 27, 2017
Accepted Feb 18, 2018

Keywords:

Data partitioning
Distributed transaction
Perofrmance
Scalable Workload-Driven
TPC-C benchmark

ABSTRACT

Cloud computing offers various and useful services like IAAS, PAAS SAAS for deploying the applications at low cost. Making it available anytime anywhere with the expectation to be it scalable and consistent. One of the technique to improve the scalability is Data partitioning. The alive techniques which are used are not that capable to track the data access pattern. This paper implements the scalable workload-driven technique for polishing the scalability of web applications. The experiments are carried out over cloud using NoSQL data store MongoDB to scale out. This approach offers low response time, high throughput and less number of distributed transaction. The result of partitioning technique is conducted and evaluated using TPC-C benchmark.

Corresponding Author:

Aakanksha Jumle,
Computer Science Symbiosis Institute of Technology,
Lavale, Pune, India.
Email: aakanksha.jumle@sitpune.edu.in

1. INTRODUCTION

In present world, there is huge widening of data due to storage, transfer, sharing of structured and unstructured data which inundates to business. E-commerce sites and application produce huge and complex data which is termed as Big Data. It is mature term that evoke large amount of unstructured, semi-structured and structured data. The cloud computing furnish with the stable platform for vital, economical and efficient organisation of data for operating it.

In order to handle and store these huge data, a large database is needed. To cope up with largescale data management system (DBMS) would not support the system. Relational databases were not capable with the scale and swiftness challenges that face modern applications, nowhere they built to take benefit of the commodity storage and computing the power available currently.

NoSQL is called as Not only SQL as it partially supports SQL. These data stores are rapidly used in Big Data and in many web applications. NoSQL is basically useful for the data which is unstructured to store. Unstructured data is growing rapidly than structured data and does not fit the relational schemas of RDBMS. Hence the NoSQL [1] data stores get introduced with high availability, high scalability and its consistency. NoSQL database is widely used to process heavy data and web application.

Nowadays most of the companies are shifting to NoSQL database [1-3] for their flexibility and ability to scale out, to handle bulky unstructured data in contrast with relational database. NoSQL cloud data stores are developed that are document store, Key-value, column family, graph database, etc. NoSQL data stores comprise its advantages for coping with the vast load of data with the aid of scale out applications.

The techniques which are in use are classified into static [4-5] and dynamic partitioning [6] systems. In static partitions, the related data item are put on single partition for accessing the data, and once the partitions formed do not change further. The advantage of static partition creation, no migration of data is done so as the cost of data migration is negligible.

The dynamic partition system, the partitions are formed dynamically in which the partitions changes frequently so as to reduce the distributed transaction. As the partitions changes, the chances of migrating the data is high so as the cost of migration.

Taking into consideration of the pros and cons of the static and dynamic partitioning systems, scalable workload driven data partitioning technique is derived. The main aim of this techniques is to reduce the distributed transaction, making the database scalable and also the performance the application to get improved. The scalable algorithm tracks the data access pattern that is which warehouse is supplying to which other requested warehouse and also the transaction logs are analysed. The proposed system frames, the partition which are formed uses NoSQL database that is MongoDB using this scalable workload-driven technique which fall neither under static nor dynamic system. The transaction logs and data access patter are monitored and partitions are formed periodically.

The essential contributions of this paper are structured as follows:

a. The design of scalable workload-driven partitioning [2] which are stand on data access pattern and traces the logs, are studied and implemented in MongoDB by forming 5 partitions.
b. The TPC-C 9 tables are mapped into different 9 collections in MongoDB and transaction are carried out on 5 partitions which are statically placed. This static approach increases the distributed transaction and the performance of the application is decrease.
c. The TPC-C 9 tables are then mapped into a single collection, the scalable workload-driven technique is used to partition the data across the 5 partition and transactions are carried over those partitions. These will reduce the distributed transaction. The performance in the terms of response time is low and throughput of the system is high as compared to above case.
d. The results of both above case are taken on local machine and also on EC2 instance to check the performance over the cloud.

The rest of this paper is as follows. The section 2 is the background of the paper consist of the related work done by the researches are explained in brief. The section 3 describes the central idea of the work done which includes design of the scalable workload driven algorithm is described. Also, the architecture of the proposed system. Mapping of the TPC-C tables into MongoDB collections are explained in the section 4. Following section 5, with the implementation of the work. Section 6 states the results of the work done. Finally, section Conclusion, finalize the paper.

2. BACKGROUND

Data partitioning means physically partitioning the database which will help to scale out the database to get available all the time. A lot of work is done on the metrics for data partitioning to give the high performance of the application to be scalable and restrict the transactions on a single partition. Some of them are listed below.

The prototype is built with benchmark tool TPC-C which uses OLTP transaction for web applications. These OLTP transaction requires quick response from the applications in recent times. TPC-C benchmark is a popular benchmark which is an Online Transaction processing workload for estimating the performance on different hardware and software configuration.

The originator Sudipto Das open up with the technique ElasTraS [4] which express Schema level partitioning for gaining scalability. The intent behind schema level partitioning is to collect alike data into the same partition, as the transactions only access the data which is needed from a large database. A major goal of ElasTraS is to have elasticity and also to reduce cost operation of the system during failure.

The author Cralo Curino has put forward, Schism: A Workload-Driven Approach to Database Replication and Partitioning [7] to improve the scalability of shared nothing distributed databases. It intends to deprecate the distributed transactions while making balanced partitions. For transactional loads graph partitioning technique is used to balance the data. Data items which are accessed in graph partitioning by the transactions are kept on a single partition.

J. Baker et al., presented Megastore [5] in which data is partitioned into a compilation of entity groups. An entity group is a selection of related data items and is put on a single node so that the data items required for enhancing the approach are accessed from a single node. Megastore aims to make the system to have: Megastore provides synchronous replication but delays the transaction.

The author Xiaoyan Wang has presented, Automatic Data Distribution in Large-scale OLTP Applications [8]. The data is divided into two categories original data and incremental data. For original data that is old data, BEA (Bond Energy Algorithm) is applied on it and for incremental data that is progressive data, online partitioning will be invoked where partitions are formed on the base of kNN (k-Nearest Neighbour) clustering algorithm. Data placements allocate these data to the partitions by genetic algorithm.

The author Francisco Cruz put forward Table Splitting Technique [1] which considers the system workload. A relevant 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.

The author Curino, suggested the Relational Cloud [9] in, which scalability is reached with the workload-aware approach termed as graph partitioning. In graph partitioning, the data items, which are frequently accessed by the transactions are kept on a single partition. 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.

The author Miguel Liroz-Gistau [6] has proposed a divergent way of dynamic partitioning technique called DynPart and DynPartGroup algorithm in Dynamic Workload-Based Partitioning Algorithms for Continuously Growing Databases, for efficient data partitioning for incremental data. The problem with static partitioning is that each time a new set of data arrives and the partitioning is redone from scratch.

The authors Brian Sauer and Wei Hao have presented [10] 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. 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

3. PRESENT THE CENTRAL IDEA OF THE WORK

3.1. Design of Scalable Workload Driven Partitioning in Mongodb

The proposed system considers the mapping of TPC-C schema into MongoDB collections for the improvement of the performance. In this partitioning strategy, transaction logs and data access pattern are monitored. The data access pattern are analysed such as which warehouse is more prone to supply the requested warehouse. That means when customer place an order, and that order is satisfied by warehouse present on a partition but the item is out of stock and that transaction is fulfilled by another warehouse from another partition. This behaviour of serving the requested warehouse is tracked and patterns are formed. Based on these two factors the partitions are formed.

3.2. Scalable Workload Driven Partitioning Algorithm

The architecture of scalable workload driven algorithm [2] gives overview of the project. The database which needs to be partitioned contains data items of local and remote warehouses in which local house will represent requested warehouse and the remote warehouse will represent the supplier warehouse. The algorithm is then applied on the database and shards are formed. Hence which will restrict the transaction to a single partition and the performance and the throughput of the application will increase. The algorithm is neither static nor dynamic, it lies between them and partitions are restructured as per need, by referring the transaction logs and access patterny.

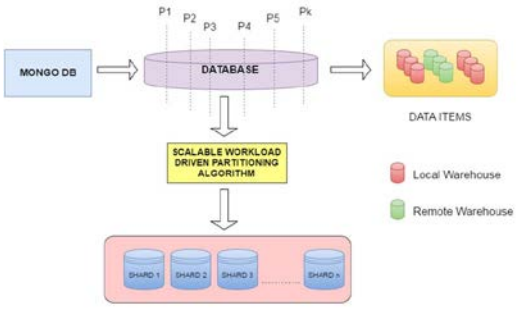

Figure 1. Architecture of the work flow

3.3. Definitions of Terms In The Algorithm

3.3.1. Load

The load of the partition [2] which is calculated in the algorithm, interprets the number of transactions executed on the each warehouse, and the total load of the partition is calculated by adding the load on each warehouse. The mean of the load is calculated to perform standard deviation on the partition which will define how much is the division of the load from the average load of the partition.

3.3.2. Association

The association of the partition [2] is calculated in algorithm, interprets the number of local transaction and distributed transaction executed on the partition. Local transaction means the transaction which are fulfilled by the requested warehouse only and the distributed transaction means the request is fulfilled by the supplier warehouse where the requested warehouse was out of stock. For example, the customer is requesting data on w1 warehouse of partition A but as there is no stock, the request is completed buy another w two warehouse of partition B.

In the scalable workload driven algorithm, the input contains number of partitions to be formed, number of the warehouse and transaction data. The output of this algorithm gives the optimised partitions. The process of the algorithm starts by distributing the warehouse statically into the partition and combinations of the partition and warehouse are formed with the help of genetic algorithm which will give the optimized combinations of it.

Later on, the calculation of the load on the each warehouse is calculated which will then sum up and give the entire load on the partition by using standard deviation. Then the load is sorted in ascending order. The association of the partition is too calculated and sorts it in descending order. A summation of both the load rank and association rank are computed and sort it in ascending order and top 5 combinations are selected as the partitions which will have optimised load balance and association. The below Figure 2 explains the scalable workload-driven algorithm [2].

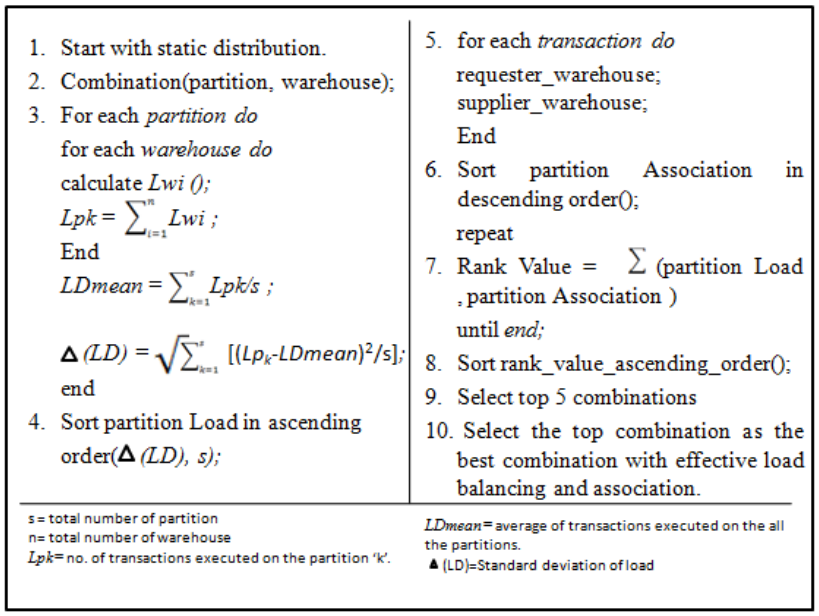

Figure 2. Scalable workload-driven algorithm

3.4. Mapping Of The Tpc-C Schema Into Mongodb

In this, mapping of TPC-C schema into the data model of the MongoDB is performed. There are total nine tables as a district, customer, stock, warehouse, orders, new-order, order-line, item, and history in the TPC-C schema. These nine tables are mapped to a single collection in MongoDB. Figure 3 shows the mapping of TPC-C schema to MongoDB. The history table has not been considered while creating the MongoDB collection. As the transaction gets triggered the hunting of for the particular data into the single collection that is single partition will perform better instead of searching the data into different 9 collections.

Using the scalable workload driven algorithm and mapping into single collection in MongoDB the partitions are formed. The reason for creating a single collection for all the 9 tables is to minimize the response time for retrieving the results.

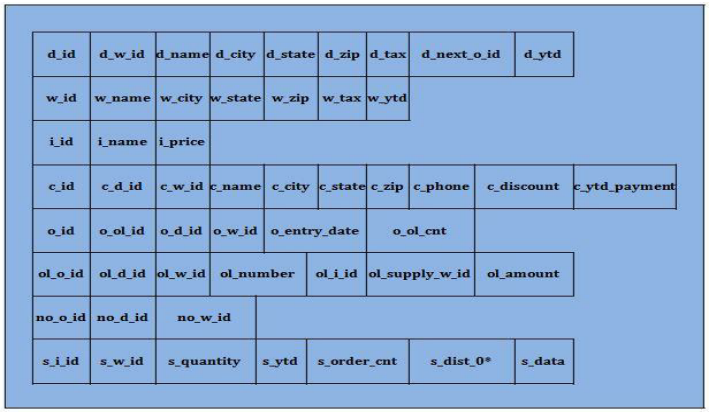

Figure 3. Mapping of TPC-C schema

4. IMPLEMENTATION

The implementation of the project considers two metric that is response time and throughput. It was performed on local machine and on cloud and the difference of their response time and throughput is measured. The below table specifies the configuration of the machines which were used for the experimental purpose.

Table1. Configuration of local machine

Configuration of Local Machine		
Edition	:	Windows 7
RAM	:	8.00 GB
Storage	:	1 GB
Processor	:	Intel Core (i3)
System Type	:	64 bit

Table2: Configuration of cloud instance

Configuration of Cloud Instance		
Edition	:	Windows 10
RAM	:	16.00 GB
Storage	:	30 GB
System Type	:	64 bit
Cost	:	$ 0.263/hr

5. RESULTS

Response time and throughput are calculated for on local machine and on cloud. Below are the graphs representing 15 warehouses, 25 warehouses and 35 warehouses. The number of users vary from 200 to 1000. The purpose of this experiment was to validate the scalability of the stated partitioning scheme with the increasing number of concurrent users. Figure 4-15 shows the response time with 15 warehouses, 25 warehouses and 35 warehouses on local and on cloud.

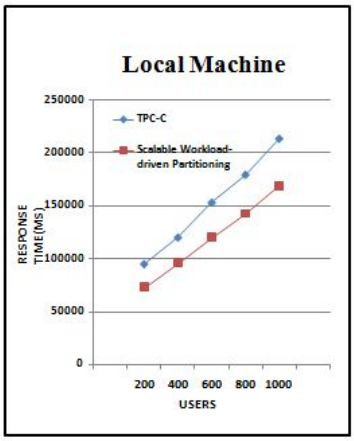

Figure 4. Response Time with 15 warehouses on local

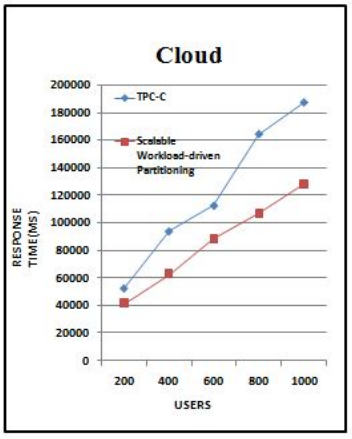

Figure 5. Response Time with 15 warehouses on cloud

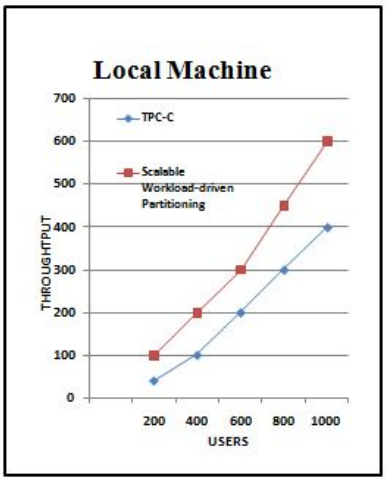

Figure 6. Throughput with 15 warehouses on local

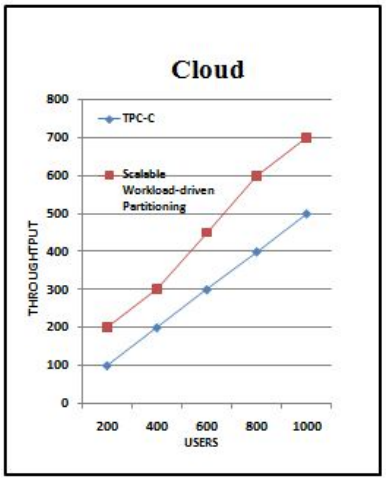

Figure 7. Throughput with 15 warehouse on cloud

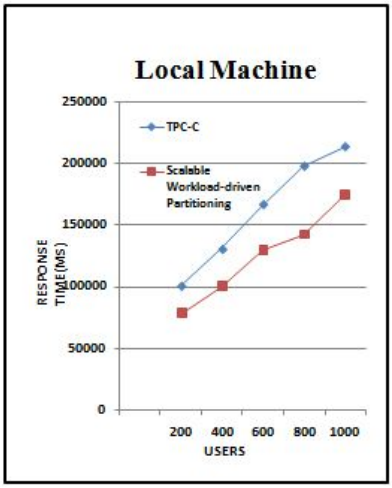

Figure 8. Response Time with 25 warehouses on local

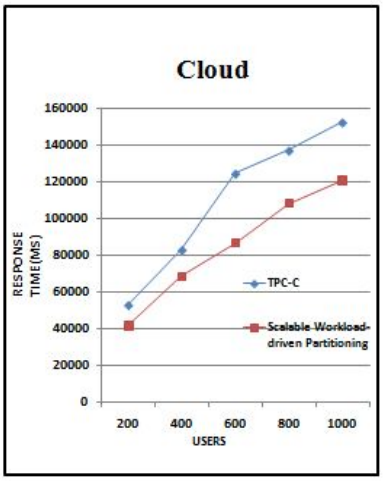

Figure 9. Response Time with 25 warehouses on cloud

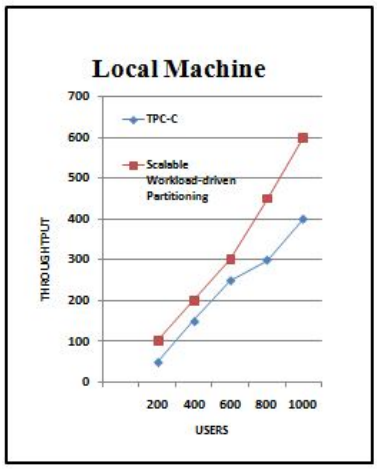

Figure 10. Throughput with 25 warehouses on local

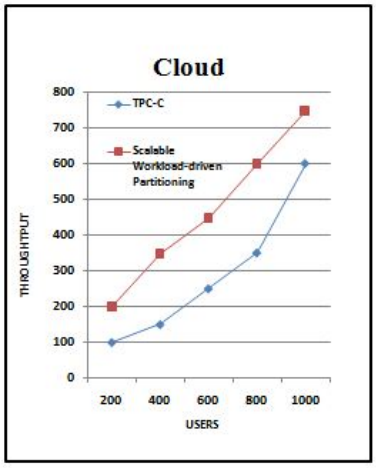

Figure 11. Throughput with 25 warehouses on cloud

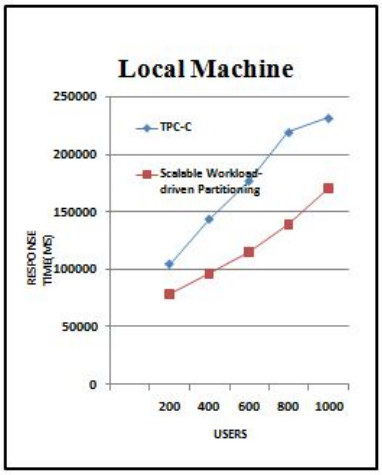

Figure 12. Response Time with 35 warehouses on local

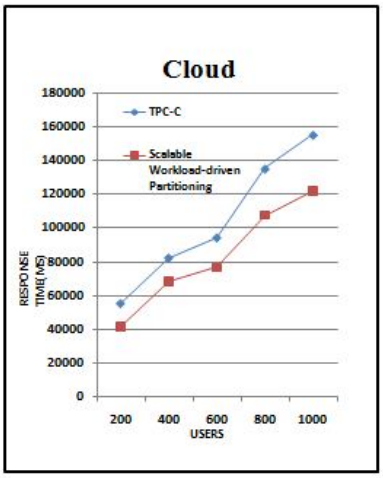

Figure 13. Response Time with 35 warehouses on local

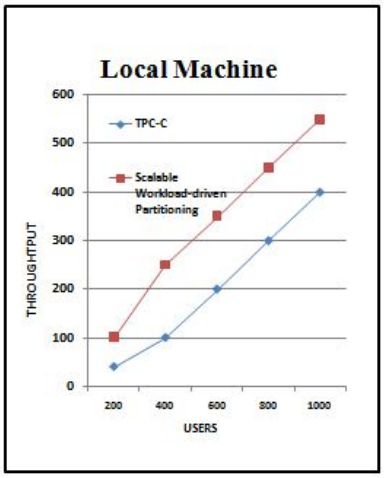

Figure 14. Throughput with 35 warehouses on local

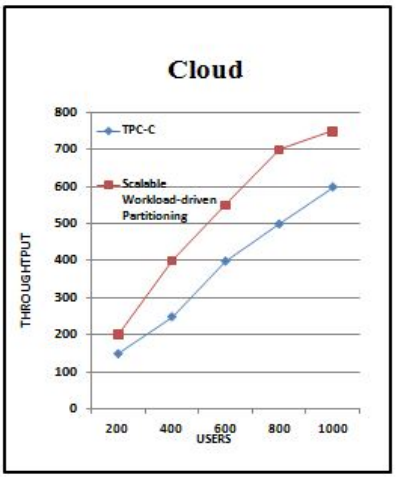

Figure 15. Throughput with 35 warehouses on cloud

6. CONCLUSION

Scalable workload-driven partitioning is implemented using MongoDB to satisfy the demands of latest cloud related applications. The experiment performed with the solution of using the workload-driven algorithm is validated over the local machine and also on the cloud. The use of EC2 cloud instance improves the metrics used for validation. By implementing the concerned scheme using the benchmark TPC-C, it has been observed that scalable workload driven partitioning reduces the number of distributed transactions and gives better response time as compared to TPC-C.

REFERENCES

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

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

[3] 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.

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

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

[6] 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.

[7] Curino C, Jones EPC, Popa RA, Malviya N, Wu E, Madden S, Zeldovich N, "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, 2011.

[8] 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, pp. 37-46, 2014.

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

[10] Brian Sauer and Wei Hao, "Horizontal Cloud Database Partitioning with Data Mining Techniques". *12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, 2015.