Journal of Theoretical and Applied Information Technology
10th February 2014. Vol. 60 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DYNAMIC WORKLOAD-AWARE PARTITIONING IN OLTP CLOUD DATA STORES

[1]SWATI AHIRRAO, [2]RAJESH INGLE
[1]Symbiosis International University, Computer Science & Information Technology, Pune, India

[2] Pune Institute of Computer Technology, Computer Engineering, Pune, India

E-mail: [1]swatia@sitpune.edu.in, [2]Ingle@ieee.org

ABSTRACT

Cloud Computing is one of the emerging field for deploying scalable applications at low cost. This paper presents a scenario of the challenges faced by application developers for building scalable web applications and provides two ways in which scalability can be achieved. First is using data partitioning which plays important role in optimizing the performance and improving scalability of data stores. Second approach works without explicit data partitioning. We have surveyed the design choices of various cloud data stores and analyse the requirements of applications and data access patterns and how these requirements are fulfilled by scalable database management systems. It also presents design model for dynamic workload-aware partitioning. In dynamic partitioning, workload is analyzed from transaction logs and frequent item sets are found out. These frequent item sets are grouped together and collocated on one partition to improve scalability. We also provide the implementation of our partitioning scheme in SimpleDB running in Amazon Cloud and Hbase.We are using industry standard TPC-C benchmark for evaluation of our partitioning scheme. We present the experimental results of our partitioning scheme by executing TPC-C benchmark.

Keywords: *Cloud, Scalability, Oltp, Data Partitioning*

1. INTRODUCTION

Data management in the cloud is basically done in two ways. i) Online Transaction Processing (OLTP) is used for recording business transactions and ii) Online Analytical Processing (OLAP) is used for analysis of large databases and finding the useful patterns. Transactions in OLTP are small. Each operation is simple select or update query that manipulate small amount of data. Update operation is significant part of OLTP workloads.These operations are frequent or repetitive and are known in advanced. This will not work for the applications which needs to access and manipulate large amounts of data. On the other hand is OLAP which access huge amount of data. The operations are not in advance. OLAP queries are read intensive and updates very few amount of data. This paper presents work on OLTP data stores.

Building scalable and consistent database system is a challenge for database researchers. Traditional enterprise infrastructure settings provides full featured ACID properties but difficult to scale out to thousands of low cost commodity servers. This introduces the need of NO-SQL Data stores. Researchers have been investigated different partitioning algorithms such as schema level, graph partitioning. All of these data partitioning algorithms are useful when the data access pattern of the web application is static. Therefore we propose to design a dynamic partitioning scheme based on the data access pattern of the web applicatons.In this paper, design model for achieving scalability of OLTP cloud data stores without compromising consistency is presented.

The Contributions of our work are as follows

- We propose the design of dynamic workload-aware partitioning. We demonstrate how this workload-aware partitioning can be used to restrict transaction to one partition.
- We provide an implementation of dynamic workload-aware partitioning in SimpleDB running in Amazon Cloud and Hbase and evaluating this partitioning scheme using TPC-C benchmark.

The rest of this paper is organized as follows. We perform analysis of various key-value stores in Section2, discuss scalable transactions in OLTP

Cloud Data Stores in Section 3, presents the design of dynamic workload-aware data partitioning scheme in Section 4 and implementation in Section 5, we show experimental evaluation in Section 6, results in Section 7 and conclusion in Section 8.

2. KEY VALUE STORES

In this section, we discuss the design model of the following cloud data stores such as Google Bigtable[1],Yahoo PNUT [2] and Amazon Dynamo[3]. In Bigtable, Tablet contains different row ranges, which are unit of distribution and load balancing. Each tablet is managed by a tablet server. To improve scalability, Bigtable uses range partitioning.

PNUT tables are horizontally partitioned into tablets. Each Tablet contains number of rows. PNUT stores tablets in servers called storage units. The tablet controller maintains the mapping of tablets to servers. This mapping is copied to multiple routers, which periodically contact the tablet controller to check for mapping updates. Scalability is achieved by, hash or range partitioning.

Dynamo [3] was designed to be a highly scalable key-value store that is highly available to reads but particularly for writes. Amazon Dynamo's partitioning scheme relies on a variant of consistent hashing mechanism to distribute the load across multiple storage hosts. High Scalability is achieved by using hash partitioning.

Key-value store provides high scalability, availability but limited consistency guarantees and lack efficient partitioning scheme for transaction processing. To overcome this, various approaches have been proposed to strengthen these key/value store with more powerful design models of data partitioning.

3. SCALABLE TRANSACTIONS IN OLTP CLOUD DATA STORES

There are two ways in which scalability can be achieved. The primary approach to achieve scalability is using data partitioning and another approach works without data partitioning.
We now discuss the work carried out in the data partitioning.Many real life applications such as banking or ecommerce applications have static data access patterns. These applications works better, when transactions are limited to single partition. Researchers are facing the challenges in building these systems as scalable, consistent, highly available .Many researchers have designed the systems with this goal and driven by notion that the data items which are accessed frequently are kept on the same partition. This minimizes the number of distributed transactions.

Sudipto Das et. al presented ElasTras[4], where scalability is accomplished by schema pattern called as schema level partitioning. ElasTras partitioning scheme is derived from TPC-C schema, so it is called as schema level partitioning. In schema level partitioning, related rows of tables are put together on single partition and distributed transactions are reduced.

P. A. Bernstein[12] et. al presented, Cloud SQL Server is a Relational database where scalability is accomplished by scaling out to low cost servers. It also uses static partitioning where transactions are enforced to execute on one partition. In Cloud SQL Server, partition is normally a table group, keyless or keyed. For Keyed table group, all the tables in the table group has common partitioning key. Row group is collection of related rows that has common partitioning key.

Curino et. al proposes design of Relational Cloud[8] with the common objective as ElasTras[4].The key feature of Relational Cloud is it uses workload-aware approach with graph partitioning [5]. In graph partitioning[5], the related rows which are accessed by transactions are kept on single partition.

J.Baker et. al presented, Megastore[9] in which also uses static data partitioning scheme where abstractions are called as entity groups. Entity groups are put on a single node so that transaction can access only single node.

As discussed before, these systems are designed with the similar objective where related rows are kept on single partition. All the four systems discussed above uses static partitioning scheme suitable for web applications whose data access pattern is static. But there are some applications such as online games where frequently accessed data items changed dynamically with time and therefore Sudipto Das et. al proposed G-Store [7], where the keys from group on different node are put together and formed a new group on single partition. Another approach works without data partitioning. These particular class of systems do not use any partitioning scheme.

Aguilera et al. [6] presented Sinfonia, in which transactions are divided into sub transactions called as mini transactions. It presents minitransaction and guarantees transactional semantics on a tiny set of operations.

Wei et. al Proposes system, Cloud TPS [11] which disperse transaction management into Local Transaction Managers(LTMs).Cloud TPS is suitable when the transactions are short, and predefined in an advanced.

D.Lomet et. al suggested design, Deuteronomy [10] in which scalability is accomplished by decomposing functions of a database storage engine kernel in two different component, transaction and data management. Transaction component executes transactions, concurrency control and recovery but not aware of physical location of data and data component. The key aspect of Deuteronomy is data can be any where in the Cloud. But In Deuteronomy, the entire load is on single transaction component for handling all the requests, so there is bottleneck for larger cloud deployments.

Researchers have discussed two different approaches for improving the scalability of databases. In this paper, we have surveyed four different systems which uses static data partitioning scheme suitable only for the web applications whose data access pattern is static. Therefore dynamic partitioning is proposed but creates overhead in creation of groups. An efficient partitioning scheme will minimize access to different database partitions and reduces distributed transactions. Therefore there is need to design efficient dynamic workload aware partitioning scheme, which will partitioned the database based on their workload.

4. DESIGN OF DYNAMIC WORKLOAD-AWARE DATA PARTITIONING

In this section, we discuss the database partitioning to allow the cloud data stores to scale out. Partitioning is a method by which scalability is achieved. It is used specially for scaling update transactions. Workload-Aware Partitioning allows designing real life and useful applications. The motivation for Workload-Aware Partitioning is that in large number of applications, transaction requires few related data items which are located across different tables. Workload-Aware Partitioning is used to collocate the related data item together and kept on the same partition. In this scheme, workload is analyzed from transaction log and frequent item sets are determined. The warehouses (wid) of these frequent item sets are collocated at one partition. So that the distributed transactions are avoided and scalability is achieved. We have used TPC-C benchmark for evaluation of our partitioning scheme. It has nine tables. Warehouse, District,Customer,History,Orders,Order_line, New_order,Stock,Item.Item id is a primary key in Item table which is foreign key in Order line table.In TPC-C,all the items are populated in the warehouse. We have made some changes to the TPC-C scheme that we are distributing items on different partitions. (only frequently purchased items) are collocated on one partition. Orderlines are monitored from transaction logs and items in an orderline which are purchased together are kept together on one partition.. For ex. If Ponds cream and Lux soap is purchased together in many transactions so these items (ol_i_id) are grouped together on same partition.so that transaction can access data from same partition.

5. IMPLEMENTATION

In this section, we are implementing our partitioning scheme by executing TPC-C benchmark. We are also comparing the results of our scheme with normalized TPC-C benchmark

A. *Implementation in Amazon SimpleDB*

We have created only one domain in SimpleDB for nine tables in TPC-C.Workload aware partitioning have reduced the number of domains accessed to only one. Domain in simpleDB contains attributes of all nine tables in TPC-C.

B. *Implementation in Hadoop Hbase*

We have configured setup for a single node Hadoop cluster where we have one master node, data node and only one region server is created.Only one table is created in hbase for nine tables in TPC-C. Table in hbase contains attributes of all nine tables in TPC-C.

6. EXPERIMENTAL EVALUATION

A. *Mapping of TPC-C to the Cloud*

TPC-C is a benchmark was designed as web application using relational databases as a backend. Therefore we need to map TPC-C to data Model of SimpleDB. TPC-C consists of nine tables. Warehouse, District, Customer, History, Orders, Order_line, New_order, Stock, Item. These nine tables of TPC-C are mapped to a only one Domain in SimpleDB running in Amazon Cloud and one table in Hbase

B. Experimental Setup

The experiments are performed on Amazon SimpleDB and Hbase.We have created two domains (tables) in SimpleDB running in Amazon Cloud and two tables in Hbase

C. TPC-C Benchmark

We have evaluated workload-aware partitioning by executing the TPC-C benchmark. It is standard OLTP workload consists of read and write transactions. There are five types of transactions.i) NEW ORDER transaction which accepts and create a new order from customer .It is mixture of read as well as write transaction ii) PAYMENT transaction which updates the balance of customer by reflecting payment of order by customer.It is also read and write transaction. iii) ORDER STATUS which keeps track of status of customer's most recent orders. It is read only transaction.iv) DELIVERY transaction finds batch of most recent 10 orders which are not yet delivered to the customer. v) STOCK level transaction which finds the recently sold items which has got a stock below threshold. It is read only transaction. In real life scenario, typically 45% transactions are NEW ORDER, 43% transactions are PAYMENT and 4% transactions are ORDER STATUS, DELIVERY, and STOCK.

7. RESULT

In this section, we are evaluating the performance of our partitioning scheme experimentally on Amazon SimpleDB and Hbase using TPC-C benchmark. We are evaluating the performance using response time of individual transactions. Dynamic workload-aware partitioning can be evaluated using the following parameters. i.e Response time and Throughput. Fig below shows results of our partitioning scheme in SimpleDB. We have kept the frequent itemset on one domain in SimpleDB and measured the response time of transaction.then we kept these frequently purchased items on two different domains in simpleDB and measured the response time of transaction.we compared the response time of transaction in both the cases and observed that response time is low when these frequently purchased items are kept together. There are five transactions in TPC-C application. We have implemented all the five transactions. Figure 1 shows response time for all transactions in dynamic workload-aware vs response time in TPC-C scheme.

8. CONCLUSION

We presented the design of dynamic workload-aware partitioning.We have used TPC-C benchmark and measured the performance of our scheme using the following metrics i.e . response time and througput . We have observed that response time for transactions are low when items are collocated at one partition.

REFERENCES:

[1] F. Chang, Dean, J., Ghemwat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., Gruber, R. E. Bigtable: A distributed storage system for structured data. In 7th OSDI (Nov. 2006), pp. 205-218.

[2] Adam Silberstein,Jianjun Chen , David Lomax, Brad McMillen,Masood Mortazavi, P.P.S .Narayan ,Raghu Ramakrishnan and Russell Sears PNUTS in Flight : Web–Scale Data Serving at Yahoo.

[3] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, W. Vogels. Dynamo: Amazon's highly available key-value store. In SOSP, pages 205- 220, 2007.

[4] S. Das, S. Agarwal, D. Agrawal, and A. El Abbadi. ElasTraS: An Elastic, Scalable, and Self Managing Transactional Database for the Cloud. Technical Report 2010-04, CS, UCSB, 2010

[5] Curino, E. Jones, Y. Zhang, and S. Madden. Schism: A Workload- Driven Approach to Database Replication and Partitioning. In VLDB,2010.

[6] M. K. Aguilera, A. Merchant, M. Shah, A. Veitch, and C. amanolis,"Sinfonia: a new paradigm for building scalable distributed systems," in Proc. SOSP, 2007

[7] S. Das, D. Agrawal, and A. El Abbadi. G-store: a scalable data store for transactional multi key access in the cloud. In SoCC '10: Proceedings of the 1st ACM symposium on Cloud computing, pages 163-174, New York, NY, USA, 2011

[8] Curino, E. Jones, R. Popa, N. Malviya, E. Wu, S. Madden, H.Balakrishnan, and N. Zeldovich. Relational Cloud: A Database Service for the Cloud. In CIDR, pages 235--240, 2011

[9] J. Baker, C. Bond, J. C. Corbett, J. Furman, A. Khorlin, J. Larson, J.-M.Leon, Y. Li, A. Lloyd, V. Yushprakh. Megastore: Providing Scalable, Highly Available Storage for Interactive Services. In Proc. CIDR, 2011.

[10] J. J. Levandoski, D. Lomet, M. F. Mokbel, K. K. Zhao. Deuteronomy: Transaction Support for Cloud Data. In 5th Biennial Conf. on Innovative Data Systems Research(CIDR), pages 123--133, Asilomar, CA, USA, January 2011.

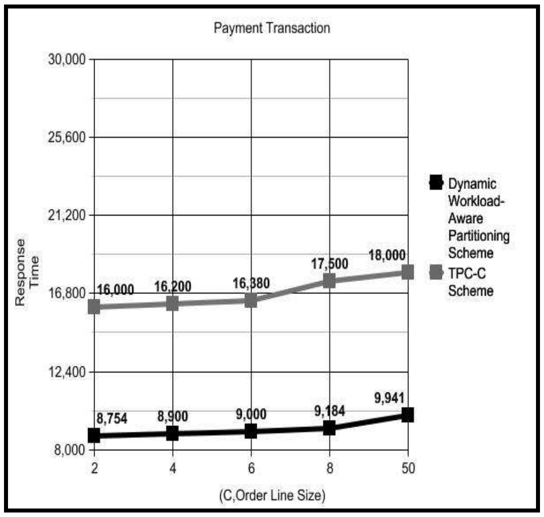
[11] Wei, G. Pierre, and C.-H. Chi. CloudTPS: Scalable Transactions for Web Applications in the Cloud Technical Report IR-CS-053, Vrije Universiteit, Amsterdam, The Netherlands, Feb. 2010.

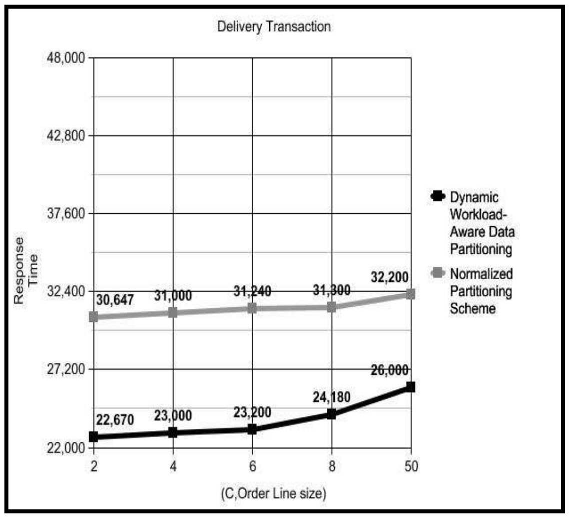
[12] P. A. Bernstein, I. Cseri, N. Dani, N. Ellis, A. Kalhan, G. Kakivaya, D.B. Lomet, R. Manner, L. Novik, T. Talius. Adapting Microsoft SQL Server for Cloud Computing. In ICDE, 2011.

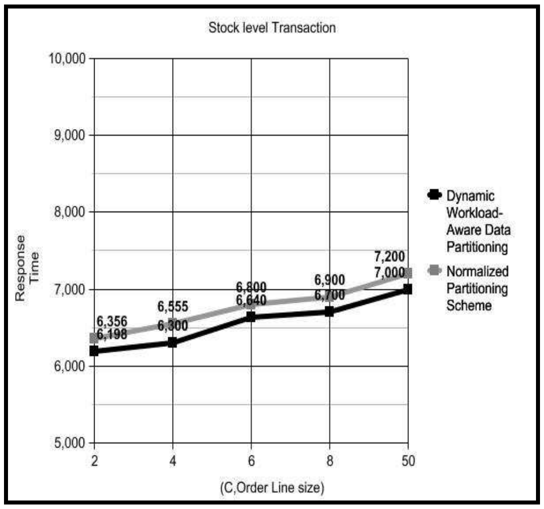
[13] D. Agrawal, A. El Abbadi, S. Antony, and S. Das. Data Management Challenges in Cloud computing Infrastructures. In DNIS, 2010.

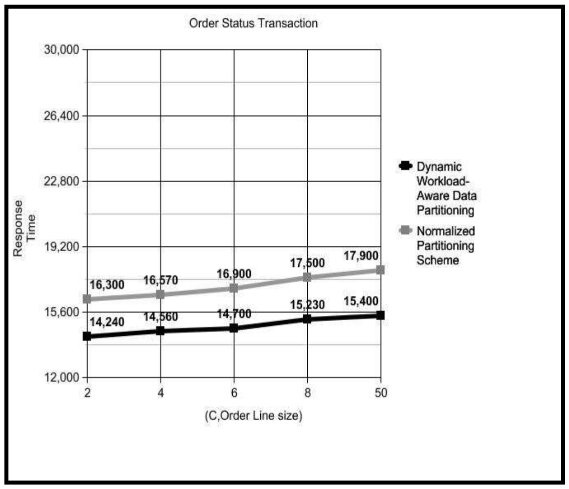
[14] S. Das, Scalable and Elastic Transactional Data Stores for Cloud Computing Platforms. PhD Thesis, 2011.

[15] S. Das, D. Agrawal, and A. E. Abbadi, "Elastras: An elastic transactional data store









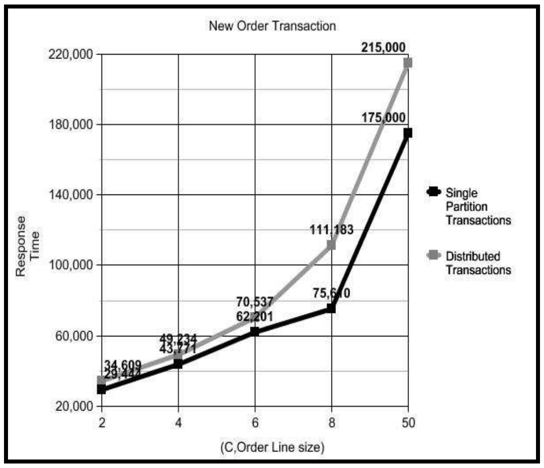


Figure 1. Response Time Of Dynamic Workload-Aware Partitioning Vs. Response Time Of Normalized TPC-C Scheme.