Journal of Theoretical and Applied Information Technology
10th January 2014. Vol. 59 No.1

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

TRANSACTION AWARE VERTICAL PARTITIONING OF DATABASE (TAVPD) FOR RESPONSIVE OLTP APPLICATIONS IN CLOUD DATA STORES

SHRADDHA PHANSALKAR, Dr. A.R. DANI
Research Scholar, Symbiosis International University
Research Guide, Symbiosis International University
Email: [1]shraddhap@sitpune.edu.in, [2]ardani_123@rediffmail.com

ABSTRACT

Online transaction Processing (OLTP) applications are business applications which are characterized by high-frequency short lived data transactions. In cloud domain, applications are expected to be highly responsive and low cost with optimized levels of consistency. Cloud data stores rely on an appropriate data partitioning scheme to achieve promising levels of responsiveness and scalability. This work presents a novel, transaction aware, static, vertical data partitioning scheme based on denormalization which performs well for OLTP applications in cloud domain. The scheme is implemented and tested on contemporary cloud data stores i.e Amazon SimpleDB and Hadoop HBase. Our work also proposes a mathematical specification model for TAVPD based data partitioning and suggests appropriate evaluation factors for a data partitioning scheme in cloud database.

Keywords: *Partitioning, Selective Consistency, Responsiveness, Consistency Index, Poisson Distribution*

1. INTRODUCTION

The design of OLTP databases in cloud domain is now an optimization problem requiring scalable performance with weaker levels of consistency. Traditionally OLTP database design requires only correctness of data, high concurrency requirements, isolation and durability guarantees (ACID). Serializability is the supreme form of isolation where the correctness of data is ensured. However cloud-based OLTPs have extended the benchmarks of the business databases to promising levels of performance with high throughput, low latency, high responsiveness and low processing cost. CAP theorem[12] by Brewer states that it is not possible to provide strong consistency and good scalability together in presence of network partitioning. This forces us to look at consistency guarantee in cloud databases as an *optimization problem*[6]. Thus a scheme for development of a data model for cloud based OLTP which is selective in data consistency and promises good responsive and scalable behavior with low processing cost is required to be modeled.

Several proposals have been made to provide transactional support to scalable web applications. The first approach suggests schemes where transactions (operations) are partitioned to smaller abstractions called subtransactions like Sinfonia[17] lightweight minitransactions. The minitransactions guarantees transactional semantics on only small set of operations such as *atomic* and *compare and swap*. It optimizes the 2 phase commit protocol[16] by piggybacked messages. CloudTPS[4] decentralizes the transaction management with local transaction manager(LTM) which also acts as data manager in transactional layer. It works well for smaller transactions which access few partitions and gives CloudTPS linear scalability.

The second approach is *data partitioning*, which is a common method used for improving the performance of databases. The data is divided into smaller pieces called partitions. All the schemes are driven by the concept that data items, which are accessed together, must be collocated. Several schemes suggesting data partition mechanisms and their effect on scalability and consistency issues have been proposed. However it is factored that distributed transactions cause consistency issues. Hence only *single partition transactions* should exist. Here workload aware *tuple–based partitioning*[13] is proposed. The tuples of database which are co accessed by a transaction

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

are kept together and reduces the number of distributed transactions. This is a graph based data partitioning algorithm. Das et al. suggests *schema-level partitioning*[5] called Elastras where the root table called Primary Partition Table is identified and the key of the primary partition table is made part of the key of all the secondary partition tables. All the related tables in the schema are put on one partition. This not only avoids cross partitioning queries but also allows writing join queries which span over a partition. Megastore[8] is static partitioning of data into abstractions called *entity groups*, which represent the granule for transactional access. This is a hierarchical key structure which provides strong consistency guarantees on top of high availability. Gstore[14] proposes of a partitioning scheme suited for applications like online gaming where the related entities are located in different partitions physically and related logically.

Most of the data partitioning techniques implement variants of horizontal data partitioning like range partitioning, hash partitioning where all the *attributes* of an object (entity in RDBMS) in a relation are stored together. OLTP applications are characterized by a large volume of transactions which require access to small set(subset) of attributes of an object in a transaction. This implies collocation of subsets of attributes of the objects which are accessed together in a single transaction. Thus our work proposes a data partitioning scheme for collocation of data from row based partitioning to column based partitioning. This reduces the remote attribute access cost, thereby reducing response time of transactions. Secondly cloud data stores promise row level consistency. Horizontal partitioning of the databases results access to data on different partitions. Some of the schemes partition the schema [5] to avoid the distributed transactions. But they cannot avoid multi row transactions over a single partition. Promising consistency to these transactions is an overhead on the application. This can be avoided with vertical partitioning of data.

Transaction processing cost is the total of the remote attribute access cost and the local attribute storage cost. A partitioning scheme is evaluated [3] with the transaction processing cost as its performance metric.

Vertical data partitioning causes denormalization of data which causes an overhead to ensure the consistency of database. Not all data however is dynamic or critical. Consistency can be selectively applied to the data depending on its criticality in the domain. If we characterize data by an index indicative of its dynamic nature and importance, we can optimize the requirement of consistency. Our work proposes this approach and refers it as consistency index of the data item.

The contributions of our work are hereby:

1. Introducing a novel, ***transaction aware, static, vertical data partitioning scheme (TAVPD)*** based on *denormalization* referred as *optimized second normal form* which ***performs*** well for OLTP applications in cloud domain.
2. Implementation of the scheme on contemporary cloud data stores like Amazon SimpleDB and Hadoop Hbase[19] using TPCC benchmark.
3. Comparison of the TAVPD scheme with normalized partitioning scheme with respect to response time and transaction processing cost.
4. Implementing selective consistency with classification of the data based on their consistency index.
5. Proposing a formal algorithm using set theory to model TAVPD based data sharder.

2. RELATED WORK

Vertical Partitioning (also called attribute partitioning) is a technique to improve the performance of transactions. In vertical partitioning, attributes of a relation R1 are clustered into non-overlapping groups and the relation R is projected into fragment relations according to these attribute groups. In distributed database systems, these fragments are allocated on different sites. Thus the objective of vertical partitioning is to create vertical fragments of a relation so as to minimize the cost of accessing data items during transaction processing. If the fragments closely match the requirements of the set of transactions provided, then the transaction processing cost, response time could be minimized and throughput will be maximized. Several vertical partitioning algorithms have been proposed in the literature. Severance et al. [21] measure the affinity between pairs of attributes and cluster attributes according to their pair wise affinity by using the bond energy algorithm (BEA). Kennedy [20] considers a mathematical model of attribute partitioning where each attribute a_i is of known length, and has probability p_i of being requested by a query. The joint probability that attributes a_i and a_j are requested by the same query is assumed to be p_ip_j . A cost function based on this assumption is derived, which reacts the

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

expected amount of data that must be transmitted in order to answer to query.
The input to the Vertical Partitioning algorithms is an Attribute Access Matrix (AAM).It is a 2-D matrix with rows as transactions[T1…..Tm] and attributes [A1……An] as columns. An example AAM is given below.

Table 1 : Attribute Access Matrix

Attributes/ Transactions	a1	a2	……. ..	An
T1	1	0	..	1
T2	0	1	..…	0
………	…….	…	…..	
Tm	0	0	……	1

AAM[i,j] = 1 → T_i access a_j
AAM[i,j] = 0 → T_i does not access a_j

Bond energy algorithm is used to group the attributes of a relation based on the attribute affinity values. The affinity between two attributes i,j is calculated as follows

$$Aff_{ij} = \sum_{t=1}^{T} q_{t,ij}$$

where $q_{t,ij}$ is number of time T accesses i, j together. The binary vertical partitioning algorithm uses the clustered affinity matrix to partition an object into two non-overlapping fragments [20] by giving algorithms to quantitatively clump the attributes together and by taking into account blocks of attributes with similar properties. The approach taken in this algorithm is splitting rather than grouping. The rationale behind this approach is that the optimal solution is much closer to the group composed of all attributes, assumed to be the starting point, than to groups that are single attribute partitions.
Graph-based Vertical Partitioning Algorithm: In this approach[22] the AAM is considered as a complete graph called the affinity graph in which an edge value represents the affinity between the two attributes. Then, forming a linearly connected spanning tree, the algorithm generates all meaningful fragments in one iteration by considering a cycle as a fragment.
In our approach, we propose a vertical data partitioning scheme which will have all the attributes accessed by a transaction together. One of the approaches which has proximity to our notion is tuple based partitioning[13]. However it exploits the co access to the data tuples whereas our approach finds the co access to the data attributes in the schema.
Also we find that not all data needs to be treated equally for the requirement of consistency. Kraska et al. has proposed a similar system [18] to classify data into three *categories* depending on the guarantees of consistency level and also allows to switch them dynamically at run time. The temporal characteristics of the data is monitored and gathered to take the decisions. This is called consistency rationing [18].Our approach also exploits the diversities in the data criticality and dynamicity. It discriminates the data on the basis of number of correct reads observed in a given time period and the probability that the user obtains the expected number of correct reads given total number of reads, writes and the period of agreement between the replicas. Thus the consistency index is a probability distribution function ranging between [0,1] which implies the probability of obtaining an index for a given attribute.

3. TAVPD scheme

TAVPD scheme refers to transaction aware vertical partitioning of the data. The attributes in the table which have reference in the same transaction will now be collocated on a single partition. The relation tables can be denormalized to an *optimized second normal form* discussed in the next subsection. The vertical partitions so obtained can be **overlapping** with regards to the primary key attributes as well as non primary key attributes with consistency index below a given threshold. The threshold is a value between [0,1] which can be defined by the application. Values closer to 1 assure stronger consistency guarantees. The concept of consistency index is discussed ahead. *The objective function is to obtain a partitioning scheme with minimal number of partitions with selective overlapping which leads to minimum response time, optimized transaction processing cost and subject to preservation of correctness of data.*

3.1 Optimized 2nd normal form

Database normalization is the process of organizing the fields and tables of a relational database to minimize redundancy and dependency. Normalization usually involves dividing large tables into smaller (and less redundant) tables and defining relationships between them. The objective is to isolate data so that additions, deletions, and modifications of a field can be made in just one

Journal of Theoretical and Applied Information Technology
10th January 2014. Vol. 59 No.1

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

table and then propagated through the rest of the database using the defined relationships. Normalization is essential for strong consistency guarantees as it avoids duplication of the data. Application databases demand normalization up to 3 NF.

The second normal form requires that every non primary key attribute to be fully functionally dependent on the primary key. Optimized second normal form states that the non-primary key can be partially dependent on the primary key if the consistency index of the key attribute has consistency index below a threshold. Optimized second normal form when opted selectively handles the inconsistencies due to denormalization of the data.

3.2 Denormalized TAVPD grouping scheme

The criterion for denormalization is that the attributes which are co accessed in the transaction are logically kept together in a single relation. This often brings the relations to non 2 NF.TAVPD scheme requires the relations to be in o*ptimized normal form* which agree with 1 NF. Boyce Codd's second normal form states that in a relation a non-primary key attribute should be fully functionally dependent on the composite primary key. TAVPD needs a much more flexible normal form. The optimized normal form allows a non-primary key to be partially dependent on the primary key if and only if the non-key attribute has consistency index below a threshold level. This implies that the non primary key is either less dynamic or less critical. The threshold ias a value between [0,1]. Lower the value, lesser is the consistency guarantee, higher the value, higher is the consistency guarantee.

3.3 TPCC Scheme

TPC-C benchmark is an OLTP workload. I t is a mixture of read only and update intensive transactions for online shopping. With "business throughput" as the performance metric, it performs five transactions as NewOrder transaction, Payment, Stocklevel, Delivery, OrderStatus on 9 entities as shown in the E-R diagram. The normalized TPCC schema is shown in the fig 1. TAVPD is applied on the scheme and a denormalized TAVPD TPCC schema is generated and shown in Fig 2.Please note that the TAVPD is applied only on the attributes which are accessed at least once in the transactions.The number of entities is reduced to 5 from 9. The attributes that are co accessed in the 5 transactions are kept together.

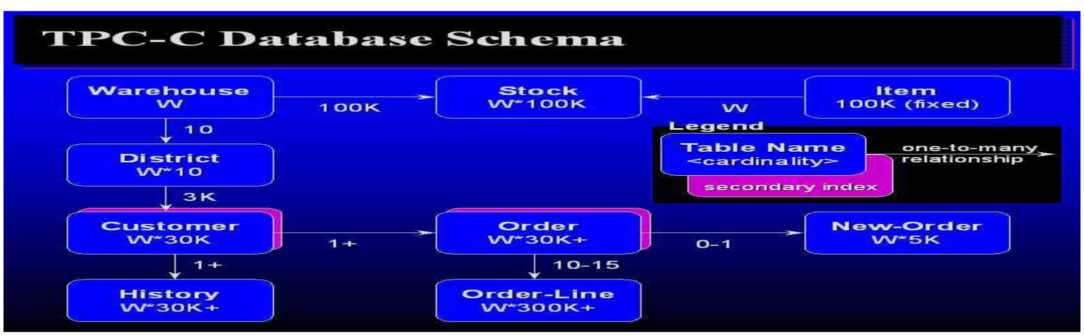


Fig 1. Normalized TPCC Schema

d_id	d_wid	d_next_o_id	d_ytd

DistrictDynamic

w_id	w_ytd

WareHouseDynamic

c_id	c_d_id	dtax	c_w_id	wtax	c_balance	c_ytd_payment	c_delivery_cnt	c_payment_cnt	cdiscount

CustomerDetails

o_id	o_d_id	o_w_id	o_c_id	ol_cnt	o_entry_d	ol_number	ol_i_id	ol_supply_w_id	ol_delivery_d	ol_quantity	ol_amount	ol_dist_info

OrderDetails

i_id	i_name	i_price	s_w_id	s_quantity	s_ytd	s_order_cnt

StockDetails

Fig 2. TAVPD TPCC Schema

Journal of Theoretical and Applied Information Technology
10th January 2014. Vol. 59 No.1

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

4. CONSISTENCY INDEX: HOW IMPORTANT IS DATA?

Data is often classified as static or dynamic data. The quantification of dynamicity as well as the criticality of the data is required. The second contribution of our work is to propose a consistency index for a data item which is logically a measure of the criticality of the data. Dynamicity refers to number of updates a data undergoes in a period. Criticality refers to importance of the correctness of this data in the read operation. It is referred as *consistency index (C_i)*. It is mathematically given *by the ratio of number of correct reads (intended/calculated) to the total number of reads on a data item in an observed time(T)*. The formula suggests that the consistency index falls in the bounds of [0,1].The occurrence of reads and updates on a data item follows Poissans distribution. In a N-replicated system, the consensus period (agreement) between the replicas play a vital role in deciding the time required to penetrate the changes in the system. Higher the period of consensus lesser is the guarantee of reading the updates. We suggest that following factors have their interaction effects on the consistency index. The proof can be easily devised using ANOVA techniques and is not described in this work.

The time interval T is further subdivided into sub intervals each of time t.

1. Average number of reads (R) on a data item in time T
2. Average number of updates (U) on the data item in time T
3. Consensus period (CP) for an agreement by all replicas on a single update.

Experimental observations show that the reads which arrive during this period of consensus are *incorrect* reads as shown in the Fig 3.

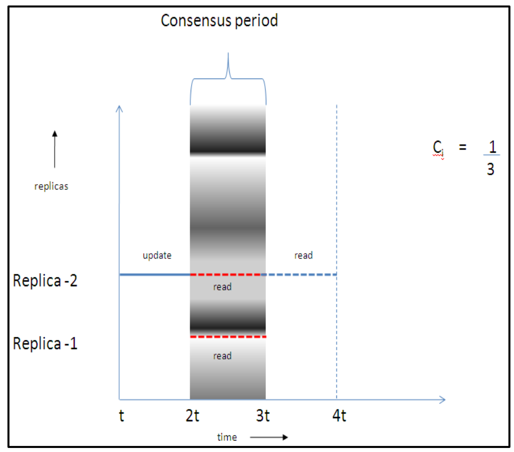
Fig 3. Incorrect reads in Consensus

4.1 Poissans Distribution and C_i

Let observed time T be divided into N subintervals of duration t such that the probability of reads as well as updates in any sub interval t is equal. The application of Poissons distribution is thus suggestive. If one update takes time *nt* for consensus,(n=1 in Fig 3) maximum total consensus period(T_c) for U non concurrent updates in a schedule will be

$$T_c = U * n * t \qquad (1)$$

If we assume R reads in the schedule i.e in time T, then average number of reads in time T_c is given by



$$\lambda = R * U * \frac{n}{N} \qquad (2)$$

Let R_c be the number of reads on the data item within the total consensus period (T_c) which guarantees a consistency index C_i given by

$$C_i = 1 - \frac{Rc}{R} \qquad (3)$$

Given average number of R reads in T time period, and λ as average number of reads in T_c time period, the probability that there will be exactly R_c reads in T_c will be given by Poisson's formula.

$$P(X = R_c) = e^{-\lambda} \frac{\lambda^{Rc}}{Rc\,!} \qquad (4)$$

where X is a stochastic variable for number of reads in the consensus period.
For a schedule S, we thus obtain the probability that a given data item would guarantee an intended consistency index. Intended C_i can be achieved with in variance with the consensus period and is under future work.

5. MODELING TAVPD SHARDING USING SET THEORY

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

Fig 4. Attribute Usage Matrix

Attributes/ Transactions	a_1	a_2	..	a_n
T1	R	0	…	W
T2	0	R	..	0
……	…	…	.	..
Tm	0	0	.	R

(R→ read access, W→ Write access, 0 → no access)

The discussion shows the mathematical representation of the process of TAVPD sharding. As discussed earlier, it is a process of grouping the coaccesed attributes. Hence we implement it using the set theory. The input to our scheme is a 2D matrix called attribute access matrix(AAM) which gives the data of all the attributes accessed by every transaction. However Attribute Usage Matrix(AUM) refers to the usage of the attributes by the transactions, Fig .4

Let A be the set of all attributes in the database of a candidate application and $|A|= N$

P_k be the set of all primary key attributes ,$P_k \subseteq A$

NP_k be the set of all non- primary key attributes,

$NP_k \subseteq A$, $NP_k \cup P_k = A$

Let T be the set of all transactions in the system and

$|T| = M$

Step- 1 Creating transaction aware partitions

AT_i be the set of attributes which transaction T_i accesses, $AT_i=\{a|a\in A \text{ and } AAM_{ti,a} =1\}$.

P is the set of partitions formed, $P = \{AT_i, i=1 \text{ to } M\}$

$PK(a_i)$= Primary Key of attribute a_i such that $a_i \subseteq A$

Let $PK(AT_i) = \cup\{x|x=PK(a_i), a_i \in AT_i\}$

Step-2 Handling the overlaps of dynamic data in partitions

A TAVPD partition set P should be a solution set such that P_i and P_j should be non-overlapping except primary keys and static non primary keys.

Now if there exists a partition $AT_i, AT_j \in P$ such that $AT_i \cap AT_j \not\subseteq P_k$ i.e there exists attribute(s) in the partitions which is non-key and redundant in both.

Let $AT_{i,j} = (AT_i \cap AT_j) - P_k$

i.e non key attributes common to both partitions.

Addressing every element $a_j \in AT_{i,j}$

1. If $AUM[T_i, a_j] = R$ and $AUM[T_j, a_j] =R$ i.e both the transactions require the attribute for reading only,
then only replication is allowed.
2. If $AUM[T_i, a_j] = W$ and $AUM[T_j, a_j] = R$ i.e one of the transaction reads and one of them writes, then
If$(Ci(a_j)) <$ threshold)
$a_{ti,j}$ should be removed from one of the partitions. The candidate set can be found out by calculating the remote attribute access cost incurred for both the transactions as discussed in the next unit and choosing the arrangement which incurs minimum cost
$AT_{i,}=AT_i-a_j$ OR $AT_{j,}= AT_j-a_j$
Else
replication is allowed.
3. If $AUM[T_i, a_j] = W$ and $AUM[T_j, a_j] = W$ i.e both the transactions write, then
If$(Ci(a_j)) <$ threshold)
a_j should be removed from one of the partitions. The candidate set can be found out by calculating the remote attribute access cost incurred for both the transactions as discussed in the next unit and choosing the arrangement which incurs minimum cost
$AT_{i,}=AT_i-a_j$ OR $AT_{j,}= AT_j-a_j$
Else
replication is allowed.

Step-3 Creating a partition which is in optimized second normal form.

S1→ Set of partitions at the end of Step-1

$| S1 |= R$

$S1PK_i$ = set of primary key for partition $P_i \subseteq S1$

For every partition P_i if there exists an attribute a_i ,such that $PK(a_i) \subseteq S1PK_i$ then $PK(a_i) \cup a_i$ can be separated in a new partition iff $Ci(a_i) <$ threshold.

Step -4 Merger of partitions where PK(p1) = PK(p2)

A new Partition with PK(p1) U P1 U P2 is created.

6. IMPLEMENTATION WITH TPCC

The prototype of the TAVPD scheme is implemented using the TPCC workload benchmark[2] an industry standard for OLTP which models an online shopping application. The original relational database of TPCC was migrated on Amazon SimpleDB as shown in Fig. 2.Using TAVPD, we created 5 domains to represent the nine tables in normalized TPCC schema. We populated 1000 item records were populated. The number of customers was simulated as 100.TPCC benchmarks for the size of order(10 order line items) were followed. The transaction mix abided to TPCC workload as NewOrder- 45%, Payment- 43%, OrderStatus- 4%, Delivery- 4% and StockLevel- 4%. However the impact of very large number of rows and its impact on the response time of the transactions is left as the future work as

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

the scheme can be complemented with appropriate load balancing and caching techniques. However the performance of TAVPD in an isolated manner can be definitely observed.

7. EVALUATION OF A TAVPD SHARDS

The performance criteria used here is the transaction processing cost and the response time. Throughput is definitely improved with response time.

Processing Cost

The overall transaction processing cost in a distributed environment consists of local transaction processing cost and the remote transaction processing cost. In a centralized database system with memory hierarchy, irrelevant attributes in the partition incur an overhead in storage and access especially when the number of tuples is very high. This is significant where the transactions access only subset of the attributes of an object at a time. This is referred as *local irrelevant cost(E_m)*. A partitioning scheme should lead to a smaller irrelevant attribute access cost. In a distributed database management system, when the relevant attributes (i.e., attributes accessed by a transaction) are in different data fragments and allocated to different sites, there is a *remote data access cost(E_R)*. In other words, each site must be able to process the transactions locally with minimal access to data located at remote sites. Introduction of replication can reduce the remote attribute access cost. But this would also add to the cost of maintaining consistency in the data across the partitions. Hence we assume a partitioning scheme with selective repetition of non-key attributes. We use the findings of Muthuraj J. in his work on evaluation of vertical partition evaluator[3].

The local transaction processing cost(E_m) for a partitioning scheme is given by

$$E_M^2 = \sum_{i=1}^{M}\sum_{t=1}^{T}\left[q_t^2 * |S_{it}|\left(1 - \frac{|S_{it}|}{n_i}\right)\right]$$

The remote transaction processing cost(E_R) for a partitioning scheme is given by

$$E_R^2 = \sum_{t=1}^{T}\Delta_{i=1}^{M}\sum_{k\neq i}\left[q_t^2 * |R_{itk}|\frac{|R_{itk}|}{n_{itk}^r}\right]$$

n : Total number of attributes in a relation that is being partitioned.
T : Total number of transactions that are under consideration.
q_t : frequency of transaction t for t = 1; 2; : : : ; T .
M : Total number of fragments of a partition.
n_i : Number of attributes in fragment i.
n^r_{ikt} : Total number of attributes that are in fragment k accessed remotely with respect to fragment i by transaction t.
f^i_{tj} : frequency of transaction t accessing attribute j in fragment i
A_{ij} : Attribute Vector for attribute j in fragment i.
S_{it} : Set of attributes contained in fragment i that the transaction t accesses; It is empty if t does not need fragment i.
|Sit| : number of attributes in fragment i that the transaction t accesses.
R^i_{tk} : Set of relevant attributes in fragment k accessed remotely with respect to fragment i by transaction t;
these are attributes not in fragment i but needed by t
$|R^i_{tk}|$: number of relevant attributes in fragment k accessed remotely with respect to fragment i by transaction t.

Table 2. Results Of The TAVPD Scheme With Normalized Partitioning Scheme For TPCC Schema

Response Time

PARAMETER	NORMALIZED SCHEME	TAVPD SCHEME
Local transaction processing cost(E_m^2)	72	117.61
Remote transaction processing Cost(E_R^2)	428.37	397.5

The TAVPD scheme is compared with the normalized schema of TPCC with respect to response time of the five transactions. The performance of every transaction is shown in the graphs given below from Fig.5 to Fig. 9

8. CONCLUSION

Consistency is an optimization problem in databases. Instead performance metrics like cost, response time and throughput significantly affect the success of an OLTP application in cloud data stores. In cloud based applications, normalized database scheme leads to poor performance with respect to these metrics. Hence denormalized database designs are essential to gain good performance. Transaction aware vertical partitioning achieves reduced cost, low response time. At the same time managing the isolation of a single row transaction can be handled by the cloud data stores with less overhead on the application and database managers. Vertical partitioning of data causes denormalization. This denormalization can be kept at optimum level and complemented with selective consistency. Our approach implements selective consistency on the data by associating consistency index to every data item. This discriminates the data on the basis of its criticality with respect to number of accesses it undergoes in an observed time.

9. FUTURE WORK

The scheme can be complemented with an appropriate caching and a load balancing

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

mechanism which would exploit the principle of selective consistency by discriminating the data on the basis of its consistency index. The scheme would then promise scalability with good response time, reduced cost and optimized consistency.

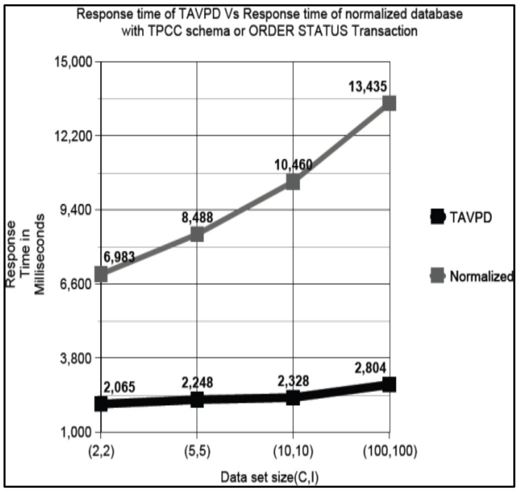


Fig 5. Response Time Of TAVPD And Normalized TPCC Scheme (Order Status)

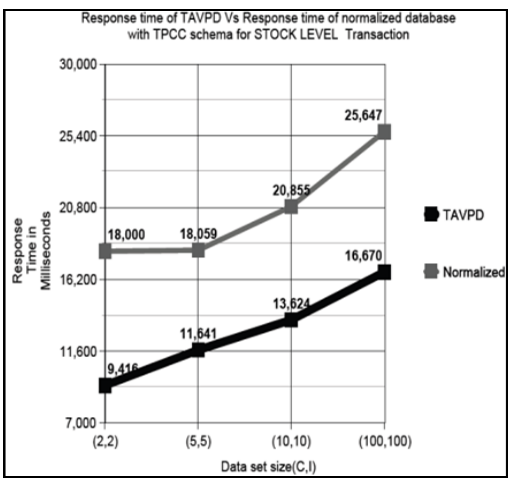


Fig 6. Response Time Of TAVPD And Normalized TPCC Scheme (New Order)

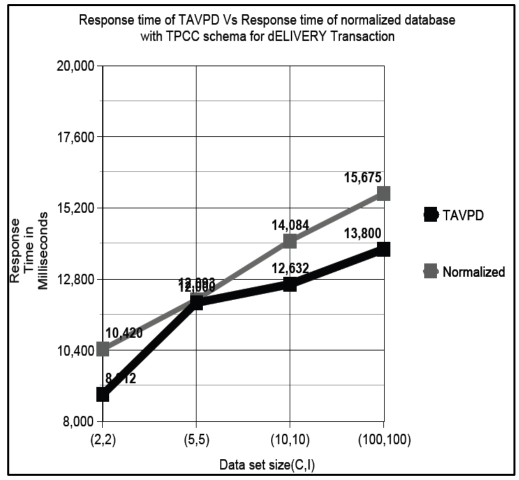


Fig 7. Response Time Of TAVPD And Normalized TPCC Scheme (Stock Level)

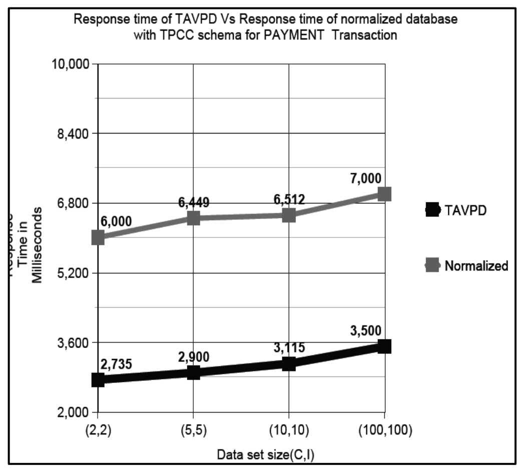


Fig 8. Response Time Of TAVPD And Normalized TPCC Scheme (Delivery)

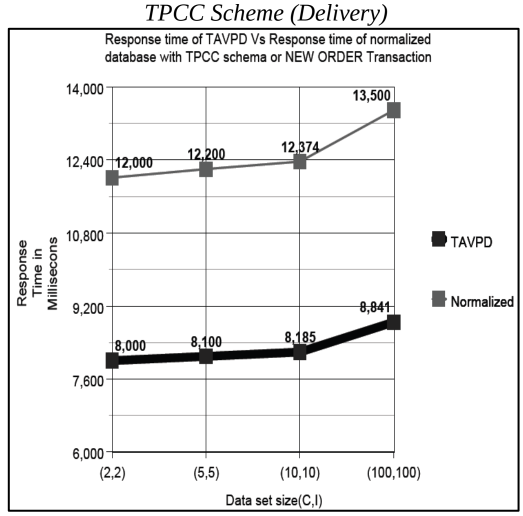


Fig 9. Response Time Of TAVPD And Normalized TPCC Scheme (Payment)

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

REFERENCES:

[1]. S Hoberman,"Data Modeler's Workbench: Tools And Techniques For Analysis And Design", *John Wiley & Sons*, 01-Sep-2008.

[2]. TPCC Benchmark Standard specification revision5.11,February 2010 *.www.tpc.org/**tpcc**/*

[3]. Jeyakumar Muthuraj, "A Formal Approach To The Vertical Partitioning Problem In Distributed Database Design", *thesis report by University of Florida*,1992.

[4]. ZhouWei, Guillaume Pierre, and Chi-Hung Chi. "CloudTPS: Scalable transactions for Web applications in the cloud"., *IEEE Transactions On Service Computing*, Feb 2011 pp 1-16

[5]. S. Das, S. Agarwal, D. Agrawal, A. El Abbadi. *ElasTraS:* "An elastic, Scalable, and Self Managing Transactional Database for the cloud", *UCSB Computer Science Technical Report* 2010-04, Santa Barbara,CA, USA (2010).

[6]. David Redding, Daniela Florescu & Donald Kossmann, "Rethinking Cost and Performance of Database Systems", *ACM SIGMOD Record*, Vol 38, Issue 1, March 2009 pgs. 43-48

[7]. ZhouWei, Guillaume Pierre, and Chi-Hung Chi, "CloudTPS: Scalable transactions for Web applications in the cloud", *Technical Report* IR-CS-053,VrijeUniversiteit Amsterdam, February 2010.

[8]. Jason Baker, Chris Bond, James C. Corbett, JJ Furman, Andrey Khorlin, James Larson,Jean Michel L´eon, Yawei Li, Alexander Lloyd, VadimYushprakh Google, Inc. "Megastore: Providing Scalable, Highly Available Storage for Interactive Services", *Conference On Innovative Database Research (CIDR) 2011*.

[9]. Sudipto Das , Divyakant Agrawal , Amr El Abbadi, "Elastras: An Elastic Transactional Data Store In The Cloud", *Proceedings of the 2009 conference on Hot topics in cloud computing*, p.7-7, June 15, 2009, San Diego, California

[10]. Daniela Florescu ,Donald Kossmann, "Rethinking Cost and Performance of Database Systems",*ACM SIGMOD* Volume 38 Issue 1, March 2009

[11]. HaifenYu , Amin Vahdat , "Design and Evaluation of a Conit-Based Continuous Consistency Model for Replicated Services", *ACM Transactions On Computer Systems* (2002).

[12]. E. Brewer, "Towards Robust Distributed Systems," *Proceedings of 19th.ACM Symosium on Principles of Distributed Computing* 2000, pp. 7-10.

[13]. C. Curino, Y. Zhang, E. Jones, S. Madden, "Schism: A Workload-Driven Approach To Database Replication And Partitioning", *Proceedings of the VLDB Endowment* (PVLDB) 3(1) (2010).

[14]. S. Das, D. Agrawal, and A. El Abbadi. "G-Store: A Scalable Data Store for Transactional Multi Key Access In The Cloud", *SoCC '10: Proceedings of the 1st ACM symposium on Cloud computing*, pages 163{174, New York, NY, USA, 2010

[15]. SherifSakr, Anna Liu, Daniel M. Batista, and Mohammad Alomari, "A Survey of Large Scale Data Management Approaches in Cloud Environments", *IEEE Communications Surveys & Tutorials*, Vol. 13, No. 3, Third Quarter 2011 311

[16]. J. Gray and L. Lamport, "Consensus On Transaction Commit", *ACM Transactions on Database Systems*., vol 31(1), March 2006, pp 133-160.

[17]. Aguilera, M.K., Merchant, A., Shah, M., Veitch, A. and Karamanolis, C., "Sinfonia: A New Paradigm For Building Scalable Distributed Systems",. *ACM Trans. Comput. Syst*. v11 i3. 11:1-11:49

[18]. T. Kraska, M. Hentschel, G. Alonso, and D. Kossmann., "Consistency Rationing in the Cloud: Pay only when it matters", *PVLDB*, 2(1):253{264), 2009.

[19]. Hadoop distributed file System http://Hadoop.Apache.Org/Hdfs/

[20]. R. Kennedy. "The Use of Access Frequencies in DataBase Organization" Ph.D Dissertation, The Wharton School, Univ. of Pennsylvania, 155 pp, May 1973.

[21]. J. Hoffer, and D. Severance. "The Uses of Cluster Analysis in Physical Database Design" *In Proceedings of 1st International Conference on VLDB*, Framingham, MA,1975, pp. 69 - 86.

[22]. S. Navathe, S. Ceri, G. Wiederhold, and J. Dou. "Vertical Partitioning Algorithms for Database Design" *ACM Transactions on Database Systems*, Vol. 9,No. 4, Dec. 1984