A Comprehensive Survey on Distributed Transactions based Data Partitioning

R.D.Bharati
Computer Engineering and IT Department
(College of Engineering,Pune)
Pune,India
rdb15.comp@coep.ac.in

Dr.V. Z.Attar
Computer Engineering and IT Department
(College of Engineering,Pune)
Pune,India
vahida.comp@coep.ac.in

***Abstract*-Distributed Transactional systems need to be fast and scalable to increase the performance of the system. Many distributed databases achieve high throughput and scalability through data partitioning. The paper presents comprehensive survey of different techniques and parameters related to distributed transactions used in the Distributed databases. 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 and availability of data. The vital aim of these techniques is to reduce distributed transaction and increase scalability of distributed databases.**

***Index Terms*—Distributed Database, Data Partitioning, Distributed Transactions.**

I. INTRODUCTION

Data partitioning plays vital role to optimize the data access cost. It enhances the performance, availability and query result of the database. Partitioning allows to access data if and when we need. Due to data partitioning like range, hash[11], schema level partitioning, graph level partitioning[26] time required to access a query can be reduced. The distributed database is collection of logical databases spread physically across multiple locations and connected by network. The data processing in distributed database is effective technique which improves performance, reliability and availability of database system. There are three different partitioning techniques applying for data manipulation.

1) Horizontal partitioning:

In horizontal partitioning tables are partitioned into sets of rows that are physically stored and accessed. In this technique data are partitioned horizontally and store on different partition. Today most of the relational databases use horizontal data partitioning techniques. In this technique[5],[31] data are partitioned horizontally and store on different partition. Today most of the relational databases use horizontal data partitioning techniques. This technique follows the static partitioning principles.

2) Vertical partitioning:

Vertical partitioning [7],[8],[9] fragments table, index and view into sets of columns. Vertical partitioning divides the table vertically into disjoint sets of column. This technique is also called column partitioning where partitioning done on the base of column.

3) Workload driven partitioning:

In this partitioning techniques data generated from web applications are partitioned on the base of access pattern and log pattern. This technique [14] improves availability of distributed transactions in the form of performance and response time.

II. LITERATURE SURVEY

Database Transaction is a unit of operations performed in database management systems. Operations like read and write in databases. When transactions said to be committed, the read and write operations performs successful. If not it may be aborted. There are four properties of databases where transactions ensure a) Atomicity b) Consistency c) Isolation and d) Durability. Execution of transaction in traditional database systems is easier the distributed database systems. Concurrency control mechanism ensure concurrent transactions integrity of database. On the basis of designing any distributed database systems classified as partitioning approach and storage approach.

A. Transaction Management in Distributed Databases

Transaction management is deals with consistency in database. There has been increasing demand in NOSQL and NEWSQL data storage systems to meet web applications. Work like Bigtable, Cassandra, Yahoo PNUTS[17] and Spanner[16]. Traditional database systems provide sophisticated data management features. Distributed transactions that access and update data on two or more nodes. Scalability of database system is achieved by transactions and query capability. The distributed commit protocols and concurrency control mechanisms have been proposed to maintain the ACID properties of distributed databases [24].

1. *Distributed Two phase commit protocol:*

Two phase commit protocol is standard widely used protocol for distributed transactions. It works in two phases. In first phase transaction manager tries to perform commit or update

978-1-5386-5257-2/18/$31.00 ©2018 IEEE

all the invited nodes. In second phase the transaction manager sends commit call to all nodes. It extends the effect of local commit actions to distributed that all sites involved to agree the distributed transactions execution. Three phase commit protocol eliminate some failure problem with two phase commit protocol [28].

2. *Write-ahead Logging protocol*

Write-ahead Logging protocol is used for maintaining log records, whether the log records is written synchronously or asynchronously. It must force the logs for update before the corresponding data storage. It reduce the number of disk writes and ensure data integrity.

3. *Optimistic Concurrency Control Protocol*

To avoid the drawbacks of distributed locking protocols a protocol used for serializable transaction. It is a concurrency control protocol [24] which assumes multiple transactions can complete without affecting each other's. Transactions can proceed without locking data resources. Before committing each transaction must verify whether data is modified by other transactions.

B. Partitioning Approach

1) Graph Based Approach:

In this approach database records are splits and represent as node and it is suitable for On-line transactional processing(OLTP) applications[4]. The Schism [18] has graph partitioning strategies which increase the scalability. It uses partitioning for enhancing scalability and replication for availability approach for distributed databases. Partitions are organized based on the graph which contains nodes and edges. Schism models the workload as a graph and enforces k- way min-cut graph partitioning algorithm[29] to lessen the effect of distributed transactions and increases the throughput. It uses static partitioning technique that is partition In[3],the authors propose automatic data partitioning in a Granola Based Distributed transactional memory. It is based on static byte code analysis for find transaction classes that can be executed using independent transaction model. They 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. The partitioning performs static analysis of data and byte-code rewriting. Graph is represented which shows the workload. Quamar et al.[21] proposed data partitioning approach for OLTP workload. The hypergraph of workload shows the decreased cost and preservation overhead in this approach of data partitioning.

2) Range Based approach:

Z.Chen et al. [23] have proposed hybrid range partitioning strategy which improve scalability and data processing. As implementation range and hash partitioning strategy is apply on YCSB benchmark. Using this strategy the hotspot problem can be avoided. The Consistent Hash strategy cannot support well for range operator.

Salam H. Matalqa et. al.[31] presented the effect of horizontal partitioning on query response time using three partitioning strategies: No partitioning, list partitioning and range partitioning. Result indicated that partitioning provides better response time than no partitioning, on the other hand, range and list partitioning strategies showed little performance improvement with the different database sizes.

J.Kamal et al. [14] developed incremental partitioning method based on hash and range techniques for OLTP databases which minimize the distributed transitions. Using incremental partitioning they reduced communication cost between number of nodes and balance the load. Workload represented in hypergraph and K-way minimum cut clustering algorithm partitions the database and balance the load. Used uniform random tuples for clustering. Not suitable for analytical data.

Milan Vojnovic et al.[26] have proposed a work on sampling based range partition method. Using simple weighted sampling scheme that accounts for data input size imbalance and provide accuracy at low cost. This method is integrated with parallel query optimizer, which results in more accurate recommendation in shorter time.

3) Data mining approach:

Yaling Xun et al. [1] developed data partitioning using Hadoop cluster which partition the transaction with high similarity. Voronoi diagram based partitioning technique is used to analyse transactions. Map reduce job partitions large database to itemsets. Compare performance with parallel FP-Growth algorithm. Difficult to manage data on heterogeneous cluster using MapReduce model.

Yin-Fu-Huang et al.[2] concentrated on partitioning based on frequent items. The Apriori algorithm used for finding frequent item sets and cosine similarity measure used to find the significance of frequent patterns. The optimal method and candidate method has been developed for data partitioning. The Candidate method creates partitions based on candidate and then select best which is having low cost. The second method used a branch and build algorithms and it consider cost of each attribute until to get optimal solution. Partitioning done based on query patterns.

Brian Sauer et al.[12] have proposed data partitioning on horizontal cloud database using data mining techniques. In this work data mining and cluster analysis of database logs are used in order to partition the data on local server. In this work data mining and cluster analysis of database access logs are used for data partitioning. The technique is on NOSQL database, where queries are clustered with other queries which is having similarity from other groups.This method compares average response times with partition algorithms like no partitioned, range partitioning on a cloud based NOSQL Database Management Systems.

Lyazid Toumi et al.[10] presented EMeD-Part:Methodology based on data mining and discrete particle sworm optimization algorithms to solve horizontal partitioning problems in data warehouses which is Jaccard index ,hierarchical clustering and meta heuristic approach for partitioning and result are compared with Genetic algorithms[8].

4) Transactions Based Approach:

Alexandru Turce et al. [3] proposed static byte code analysis to find transactions. They 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. The partitioning performs static analysis of data and byte-code rewriting. Author presented automated data partitioning 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 [18] prototype with edges having weights.

S.Das et al. [20] have extended schema level partitioning for high scalability. In schema level data partitions it reduce the scalable transaction. The interrelated rows are placed in single partition which done on partitioning key. ElasTrans has been evaluated by clustering of machines under EC2.They demonstrate schema level data partitioning can be reduce the transactions to single partition. TPC-C benchmark is used to evaluate performance in Amazons cloud. This approach focuses on only scalability not availability. ElasTras comes into picture from the scalable Key-Value stores to decrease distributed transactions and eliminate scalability bottlenecks, as all know that partitioning the database is used for providing scalability. ElasTraS are based upon schema level partitioning scheme of the database to assist capability, despite the fact that transactions are restricted to single partitions. Discover design concepts which are utilized in designing scalable systems concurrently assuming transactional guarantees. ElasTras is appropriate for internet applications which access patterns are static. ElasTras only focus on scalability, not availability so no need to concentrate on replication.

Cloud TPS [30] for a scalable transaction in the web applications which has followed strict ACID properties. It splits transaction manager within multiple Local Transaction Manager (LTMs) to support scalable transactions. The items are assigned to Local Transaction Manager. By applying consistent hashing techniques it accomplish high scalability. Cloud TPS based on static partitioning technique to enhance scalability and suitable for those applications, which is based ion access pattern.

Wang at al. [6] proposed online aggregation system using map reduce framework on cloud which is based on contents repartitioning. Authors have proposed block placement strategy to reduce redundant I/O cost. To increase the resource utilization the contents aware partitioning is considered.

C. Distributed Data Storage Approach

As possibility of distributed deadlock if distributed transactions allows in systems need to detect and implement concurrency control mechanism. It can cause distributed transaction to be aborted or restarted to avoid latency in throughput.

On the basis of data storage in distributed databases approach studied different distributed database models.

1) Calvin

A. Thomson et al.[22],[32] designed a transactional scheduling and data replication layer above the storage system to reduce contention cost associate with distributed transaction. It supports multiple consistency levels of replicas. Calvin handle deterministic locking protocol. Calvin Support horizontal scalability of database and unconstraint distributed transactions while supporting asynchronous and Paxos-based synchronous replications.

2) E-Store

Rebecca Taft et al.[13] developed E-store an elastic partitioning framework for distributed online transactional processing(OLTP) databases. It addresses two tier data partitioning strategy-store is designed to maintain performance and diverse overload by balancing tuples access across elastic partitions. The framework consist of E-Monitor and E-Planner who decides there is need of node or not and re-organize data.

3) Megastore

Jason Baker et al.[15] presented Megastore is a strongly consistent and globally distributed database. It is based on Paxon replication consensus algorithm for distributed data. Megastore combines the scalability of a key value data store with regard to a traditional RDBMS and offers the strong consistency and high availability. The author has used Google BigTable supporting arbitrary read and write throughput. Megastore gives full ACID properties inside the partitions. This system partitions the data into multiple entity groups i.e. the set of correlated data items which independently replicated over the set of servers and each update is replicated synchronously across the identical partitions with acceptable latency. The transaction which requires the strong consistent view of the database to fulfill its execution restricts its execution to a single entity group. proposed graph based partitioning for the Relational Cloud system which reduce the distributed transactions while load balancing .They proposed workload aware data partitioning approach to improve scalability. Graph partitioning techniques is used for transactional workload.

4) G-Store

Sudipto Das et al.[19] designed a scalable data store based on key group abstraction and key grouping protocol which supports transactional multi access guaranty for key value store which guarantees dynamic and non-overlapping group of keys using key value store. It is a scalable data store for transactional multi key access. G-Store provides the scalability of the Key-Value store and handles number of concurrent group creations and support multi key operations transactions. Optimistic concurrency control is used for guaranteeing serializable transactions. Write Ahead Log protocol ensures atomicity and durability properties of transactions.

5) Relational Cloud

Carlo Curino et al.[25]designed relational cloud uses workload aware approach to multi-tenancy which identify workload and graph partitioning algorithms to achieve linear for transactional workload. Relational Cloud defines three significance challenges like efficient multi-latency, elastic scalability and database privacy.

	Calvin	G-Store	E-Store	Mega-store	ElsTranS	Relational Cloud
Key Access	Multiple Key	Multiple Key	Multiple Key	Entity Groups	Single	Multiple Key
Concurrency Control Protocol	Strict Two Phase	Optimistic Concurrency Control	2PC (E-Monitor)	2PC	Traditional Distributed Database	CryptDB
Replication	Asynchronous and Paxos based Synchronous	Synchronous	Synchronous	Paxos based	Synchro-nous	Synchro-nous
Partitioning	Horizontal Scalability	Range and Hash	Range	Schema level	Schema level	Graph Based, workload-aware partitioning

Table 1: Comparison of different Distributed Data Storages

III. CONCLUSION

The paper presents comprehensive survey of different techniques and parameters related to data partitioning used in the Distributed databases. We analyze different distributed data storages with respect to data replication, Concurrency control mechanism, key access and data partitioning. The vital aim of these techniques is to reduce distributed transaction and improve system 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. This paper highlights the recent advancement in distributed and cloud based data partitioning which improve systems performance.

REFERENCES

[1] Y.Xun, J.Zhang, X.Qin and X.Zhao, "FiDoop-DP:Data Partitioning in Frequent Itemset Mining on Hadoop Clusters," IEEE Transactions on parallel and distributed systems, vol. 28, no. 1, pp. 101-114, 2017

[2] Y.Fu Huang and C.Ju Lai, "Integrating frequent pattern clustering and branch- and-bound approaches for Data Partitioning," Elsevier Journal of Information Sciences (328),pp. 288-301, 2016

[3] Alexandru Turcu, R. Palmieri, B. Ravindran, and S. Hirve, "Automated Data Partitioning for Highly Scalable and Strongly Consistent Transactions," IEEE Transactions on Parallel and Distributed systems, vol. 27, No. 1,pp. 1-14, 2016

[4] Marco Serafini and Rebecca Taft, "Clay: Fine Grained Adaptive Partitioning for General Database Schemas,"Proceedings of the VLDB Endowment, vol. 10, no. 4 pp. 445-456, 2016

[5] K. Herrmann, H.Voigt, and W.Lehner, "Online horizontal partitioning of heterogeneous data," it Information Technology Methods and Applications of Informatics and Information Technology,vol.56(1), pp.4-12, 2014

[6] Y.X.Wang,J.Z.Luo,A.B.Song and F.Dong, "Partition-Based Online Aggregation with Shared Sampling in the Cloud.," Journal of computer science and technology, vol.28, no.6, pp. 989-1011,2013

[7] Sanjay Agrawal ,Vivek Narasayya and Beverly Yang, "Integrating Vertical and Horizontal Partitioning into Automated Physical Database Design," SIGMOD, 2004

[8] C. Cheng,W.Lee, K. Wong, "Genetic Algorithm-Based Clustering Approach for Database Partitioning," IEEE Transactions on Systems, a Man, and Cybernetics part c: applications and reviews, vol. 32,no. 3, pp.215- 230, 2002

[9] Bakshi Rohit Prasad, Unmesh Kishor Bendale and Sonali Agarwal, "Distributed Feature Selection using Vertical Partitioning for High Dimensional Data," In Proc.International Conference on Advances in Computing, Communications and Informatics, 2016

[10] Lyazid Toumi, Abdelouahab Moussaoui and Ahmet Ugur, "EMeD-Part: An Efficient Methodology for Horizontal Partitioning in Data Warehouse," in Proc. International Conference on Intelligent Information Processing,Security and Advanced Communication, no. 43, Batna, Algeria, 2015

[11] Caio H. Costa, Joao Vianney B. M. Filho, Paulo Henrique M. Maia, Francisco Carlos and M. B.Oliveira, "Sharding by Hash Partitioning-A database scalability pattern to achieve evenly sharded database," In Proc. 17th International Conference on Enterprise Information System, 2015

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

[13] Rebecca Taft, Essam Mansour, Marco Serafini, Jennie DugganF, Aaron J. Elmore N Ashraf Aboulnaga, Andrew Pavlo and Michael Stonebraker, " EStore: FineGrained Elastic Partitioning for Distributed Transaction Processing Systems," 41st International Conference on Very Large Data Bases, Vol. 8, No. 3, pp. 245-256,2015

[14] Joarder Kamal, Manzur Murshed and Rajkumar Buyya, "Workload-Aware Incremental Repartitioning of Shared-Nothing Distributed Databases for Scalable OLTP Applications," ElsevierFGCS Special Issue: UCC2014

[15] J. Baker, C. Bond, J. C. Corbett, J. Furman, A. Khorlin, J. Larson, J.-M. Leon, Y. Li, A. Lloyd, and V. Yushprakh, "Megastore: Providing scalable, highly available storage for interactive services," In CIDR, vol. 11, pp. 223-234, 2011

[16] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman,S. Ghemawat, A. Gubarev, C. Heiser and P. Hochschild, "Spanner: Googles globally distributed database," in Proceeding OSDI'12 Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation, Pp. 251-264, 2013

[17] B. F. Cooper, R. Ramakrishnan, U. Srivastava, A. Silberstein, P. Bohannon, H.A. Jacobsen, N. Puz, D. Weaver, and R. Yerneni, "Pnuts: Yahoo!s hosted data serving platform," Proceedings of the VLDB Endowment, pp.1277-1288, 2008

[18] C Curino, E. Jones, Y. Zhang and S. Madden, "Schism: a Workload Driven Approach to Database Replication and Partitioning," 36th International Conference on Very Large Data Bases, 2010

[19] Sudipto Das, Divyank Agrawal, and Amr El Abbadi, "G-store: a scalable data store for transactional multi Managing Transactional Database for the Cloud," UCSB Computer

Science Technical Report, 2010.
[20] Sudipto Das,Shashank Agrawal, Divyank Agrawal, and Amr El Abbadi, "ElasTraS: An elastic,Scalable and transactional Data store in the clou," HotCloud,pp.131-142, 2009
[21] A. Quamar, K. A. Kumar, and A. Deshpande, "Sword: scalable workload- aware data placement for transactional workloads," In International Conference on Extending Database Technology, pp. 430-441, 2013
[22] A. Thomson, T. Diamond, S.C. Weng, K. Ren, P. Shao, and D. J. Abadi, "Calvin: fast distributed transactions for partitioned database systems," In Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, ACM, 2012
[23] Zhikun Chen, Shuqiang Yang, Shuang Tan, Ge Zhang and Huiyu Yang, "Hybrid Range Consistent Hash Partitioning Strategy–A New Data Partition Strategy for NoSQL Database," In Proc. 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2013
[24] P.V.Tamer Ozsu,,Patrick Valduriez, "Principles of Distributed Database Systems," Springer, 2011
[25] Curino C, Jones EPC, Popa RA, Malviya N, Wu E, Madden S and 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
[26] Milan Vojnovic, Fei Xu and Jingren Zhou, "Sampling Based Range Partition Methods for Big Data Analytics," Technical Report, Microsoft Research MSR-TR,2012
[27] Tilmann Rabl and Hans Arno Jacobsen, "Query Centric Partitioning and Allocation for Partially Replicated Database Systems, " SIGMOD17, pp. 315-330, 2017
[28] Philip Bernstein, Eric Newcomer, "Principles of Transaction Processing" Elsevier 2009
[29] Jiewen Huang and Daniel J. Abadi, "LEOPARD: Lightweight Edge Oriented Partitioning and Replication for Dynamic Graphs," Proceedings of the VLDB Endowment, vol. 9, no. 7, pp. 540-551, 2016
[30] W.Zhou and G.Pierre, "CloudTPS:Scalable Transactions for Web Applications in the Cloud," IEEE Trancasctions on Services Computing.vol.5,no.4,pp. 525-539, 2012
[31] Salam H. Matalqa and Suleiman H. Mustafa, " The Effect of Horizontal Database Table Partitioning on Query Performance," The International Arab Journal of Information Technology, vol. 13, no. 1A, pp. 184-189, 2016
[32] A. Thomson, T. Diamond, S.C. Weng, K. Ren, P. Shao, and D. J. Abadi, "Fast Distributed Transactions and Strongly Consistent Replication for OLTP Database Systems," ACM Transactions on Database Systems, vol. 39, no. 2, 2014.