Graph Based Workload Driven Partitioning System by Using MongoDB

Arvind Sahu, Swati Ahirrao
Department of Computer Engineering, Symbiosis International University, India

Article Info

Article history:

Received May 24, 2017
Revised Jan 18, 2018
Accepted Feb 11, 2018

Keywords:

Graph Partitioning,
NoSQL Database,
Online Transaction Processing (OLTP),
Scalability
TPC-C benchmark,

ABSTRACT

The web applications and websites of the enterprises are accessed by a huge number of users with the expectation of reliability and high availability. Social networking sites are generating the data exponentially large amount of data. It is a challenging task to store data efficiently. SQL and NoSQL are mostly used to store data. As RDBMS cannot handle the unstructured data and huge volume of data, so NoSQL is better choice for web applications. Graph database is one of the efficient ways to store data in NoSQL. Graph database allows us to store data in the form of relation. In Graph representation each tuple is represented by node and the relationship is represented by edge. But, to handle the exponentially growth of data into a single server might decrease the performance and increases the response time. Data partitioning is a good choice to maintain a moderate performance even the workload increases. There are many data partitioning techniques like Range, Hash and Round robin but they are not efficient for the small transactions that access a less number of tuples. NoSQL data stores provide scalability and availability by using various partitioning methods. To access the Scalability, Graph partitioning is an efficient way that can be easily represent and process that data. To balance the load data are partitioned horizontally and allocate data across the geographical available data stores. If the partitions are not formed properly result becomes expensive distributed transactions in terms of response time. So the partitioning of the tuple should be based on relation. In proposed system, Schism technique is used for partitioning the Graph. Schism is a workload aware graph partitioning technique. After partitioning the related tuples should come into a single partition. The individual node from the graph is mapped to the unique partition. The overall aim of Graph partitioning is to maintain nodes onto different distributed partition so that related data come onto the same cluster.

Corresponding Author:

Swati Ahirrao,
Department of Computer Engineering,
Symbiosis International University, Pune, India.
Email: swatia@sitpune.edu.in

1. INTRODUCTION

Nowadays, data is generated by their sources is very huge in volume that is called as Big-Data. The companies need a strong foundation for Big-Data handling. So many companies giving more importance to NoSQL database as compare to traditional relational database because relational databases are unable to scale and lack of flexibility. Relational databases also cannot handle globally massive amounts of unstructured data. So it is required to store and process that huge amount of data by using NoSQL database. NoSQL database is persistence solution in the world of Big Data. NoSQL database is more suitable as compare to RDBMS in terms of scalability, volume, variety and efficiency. NoSQL database is capable to handle all type of data like structured, semi-structured, and unstructured data.

To achieve scalability in NoSQL the data should be partitioned and then distribute those partitions to the different geographically available servers. Partitioning is the most effective technique which is used to access scalability in a system. Scalability is to process the entire generated workload at the defined time boundaries for an application that is running on a number of server nodes. Some most commonly used partitioning techniques are round-robin partitioning, range partitioning and hash partitioning. In round-robin partitioning, each tuple is uniformly distributed to an alternate partition. In range partitioning, the tuples are partitioned according with a range of partition key. In hash partitioning, the tuples are partitioned by using hash key. But these partitioning methods are not efficient for the small workload transactions that access a less number of tuples. Round-robin partitioning, range partitioning and hash partitioning methods are unable to represent sensible n-to-n relationship between tuples. For example social networking like Facebook data appear with the n-to-n relationships that is hard to partitioning by using round-robin partitioning, range partitioning and hash partitioning methods. Sometimes the related tuples come into different partition and unrelated tuples come under same partition which is root cause of distributed transaction. In Fine-Grained partitioning, related individual tuples are combined together in the same partition to reduce the distributed transaction but lookup table is very expensive because of its size. In case of Many-to-many relationship it is hard to partition by using Fine-Grained partitioning. ElasTras [1] used schema level partitioning to improve scalability. In ElasTras Scalability is achieved by the execution of transactions onto the multiple partitions. ElasTras also not formed effective partitions. In Spectral partitioning methods [2-3], partitions produced are effective but very expensive because they require the computation of eigenvector corresponding to the fiedler vector. In Geometric partitioning algorithms [4], are very fast to perform partitions but formed partitions quality are worse than partitions of Spectral partitioning.

The web applications and websites of the enterprises are accessed by a huge number of users with the expectation of reliability and high availability. Social networking sites are generating the data in very huge amount and at very fast rate, so to handle that data we need of both computer software scalability and hardware scalability. The concept behind partitioning or shading is to spread data of a table across numerous partitions. Partitioning techniques should be picked up appropriately so that after partitioning the result become more accurate and effective.

When data is stored in SQL data bases, it can be partitioned to provide efficiency and scalability. Data partitioning is possible in SQL data store but SQL data stores are unable to partition the data based on relationship. In SQL data partitioning increases the multisite/distributed transactions because sometimes related rows come into different partition.

Hence, it is necessary to find an effective partitioning technique, which will consider the relativity among the tuples or data at the time of partitioning. In proposed system, Schism [5] approach is used for partitioning the NoSQL data base. Schism is a workload driven approach for data partitioning. In Schism for Graph Partitioning METIS [2] algorithm is used. METIS provides balanced partitions after partitioning the data. Proposed system improves Scalability and reduces the number of Distributed transactions.

With the goal to acquire ideal data placement method in different partitions, here numerous workload-aware partitioning and load balancing techniques have been examined. Graph partitioning technique is easy to understand and the relation between data can represent effectively in the form of Graph. In the Graph database, the nodes represent the database tuples and nodes are connected by an edge if two or more nodes are participating in the same transaction. Graph based workload aware partitioning technique is basically applied for scalability but sometimes if the data partitions are not formed appropriately then it leads to the expensive distribute transactions. Here expensive in terms of response time and resource utilization or network usage. So to overcome the problem of expensive distributed transactions min-cut k balanced partitions come into the picture. In min-cut balanced partitions, data partitioning happens based on some relation that data contains, which minimize the multisite/distributed transactions. After partitioning the database, the decision tree classifier techniques are applied to extract the set of rules based on that rules data is partitioned. When scalability improves then maintenance of data becomes easy and time taken to process a query reduces. Ultimately, along with the scalability performance also increases.

The rest of the paper is organized as follows: in Section 2, papers related to scalability and graph database partitioning is discussed. Section 3 gives Brief overview of proposed system. In Section 4, Design of graph based workload driven partitioning system is presented. In Section 5, Implementation details explain implementation and the performance evaluation of the partitioning system. In Section 6 and Section 7, experimental setup and results are described respectively. Finally in Section 8 conclusion of the paper.

2. RELATED WORK

Scalable database and distributed database is emerging and interested topic for the database research group. Accordingly, many techniques for scalable database transaction are already present. In this chapter, all the existing partitioning algorithms and graph partitioning techniques are discussed.

Curino [5] have proposed Schism technique for database replication and partitioning for OLTP databases. Schism is a data driven graph partitioning algorithm which improves the scalability of the system. Schism uses METIS algorithm for partitioning the graph. Schism formed the balanced partitions that reduce the number of distributed transactions. K-way partitioning algorithm is used to find refined partition in METIS. K-way applies for further partitioning the data that reduces the effect of distributed transactions and increases the throughput.

Schism provides good partitioning performance and easy to integrate into existing databases. In Schism representation of n-to-n relationship effectively and allow multiple schema layouts. Schism is a better approach for replication and partitioning of social networking data. In Schism workload changes can affect the performance and aggressive replication is used.

Karypis [6] have proposed METIS which follows Multilevel Graph Partitioning. METIS is an open source graph partitioning technique. In Multilevel graph partitioning there are three phases. Firstly, in graph coarsening phase the graph developed by incoming workload is partitioned. Second, in initial partitioning the graphs are further partitioned into the small graphs for refinement. Initial partitioning produces the balanced partitions. Third, in un-coarsening is for rebuilding the final graph after refinement. METIS is fast as compare to other partitioning algorithm. METIS provides quality of partitions and required less memory because of dynamic memory allocation. In METIS, we cannot major exact amount of required memory.

Tatarowicz [7] have proposed Fine-Grained Partitioning for distributed database. In proposed partitioning approach related individual tuples are combined together in the same partition. Lookup table used to maintain a smooth partitioning because it is needed to store the location of each tuple. The lookup table stores location of each tuple that acts as a metadata table. Many-to-many relationship is hard to partition because number of distributed transactions increase in case of Many-to-many relationship. In Fine-Grained partitioning the number of distributed transactions can be reduced by well assignment of tuples to each partition. It reduces the main memory consumption and improves the performance. Sometimes to maintain Fine Grained Partitioning routing (Lookup) tables is very expensive because of its size.

Quamar [8] have proposed Scalable Workload-Aware Data partitioning for Transactional data processing. Hypergraph is used to represent the workload and reduces the overheads of partitioning by hyperedge. Overheads happen at the time of data placement or query execution at runtime that is reduces by SWORD technique. Workload changes are handled by incremental data repartitioning technique. For high availability it uses active replication technique. By Active replication number of distributed transactions reduces, accessibility increases and load balancing can be accessed. SWORD is used to minimize the cost of distributed transactions. SWORD represents data in compressed format so number of generated partitions would be minimal.

Lu Wang [9] have proposed a Multi-level Label Propagation (MLP) method for partitioning of the graph. All the web users are represented on the graph. If graph partitioned efficiently then load balancing becomes easy and less communication overhead. The quality of generating partitions by MLP approach is compared to the small size graph. To evaluate quality of the formed partitions it is compared to METIS. In MLP formed partitions are good with respect to time and memory. MLP are more effective and scalable partitioning which can scale up to billions of nodes. MLP algorithm cannot use for general-purpose.

Chris [10] have proposed a Min-max Cut algorithm. Aim of Min-Max is to increase number of similar nodes within a cluster and reduce the number of similar nodes between different clusters. Similar nodes between two sub-graphs should be less. Similar nodes within each sub-graph should be higher. Min-max cut is used to improve the clustering accuracy and gives an optimal solution.

Tae-Young [11] have proposed k-way Graph Partitioning Algorithm to reduce the size of the graph. The recursive spectral bisection method reduces the size of the graph by the breaking edges and vertices. The recursive spectral bisection method is used for clustering small graph in a k-way. Balancing constraints should maintain to all partitions of graphs. In Multi-Level Partitioning for recursive partitioning k-way algorithm is used. K-way takes less time to partition the graph. K-way is a good option for partitioning small graphs only.

Dominique [12] introduce Parallel Hill-Climbing algorithm for graph partitioning. This algorithm is used for share memory for refinement algorithm parallel. Hill-climbing takes imaginary move on different vertices so that it can find new local minima. The refinement algorithm produces high-quality partitioning. Hill-Scanning is the refinement algorithm for refining k-way partition. Hill-Scanning has better performance than other parallel refinement algorithm. Hill-Scanning it is much faster and provides better quality partition.

Maria [13] introduces a new shared memory parallel algorithm that is Coupling-Aware Graph Partitioning Algorithm. Coupling-Aware is used to produce high-quality partitions as compared to the k-way partitioning algorithm. Co-partitioning allows scalability in term of processing unit. In Coupling-aware partitioning there is no need to partition codes independently or separately. It can reuse available codes through a coupling framework without combining them into a standalone application. Coupling –Aware reduces the coupling communications costs. Coupling-Aware provides better global graph edge cut.

James [14] have proposed Simulated Bee Colony Algorithm to partition the graph. Simulated Bee Colony is totally inspired by the foraging behavior of honey bees. In SBC algorithm the partial result can be reused further if compatible in a scenario. SBC produces very high quality of partitions. SBC is used when the quality is most important and performance can be moderate. SBC algorithm is not suitable for real-time applications. SBC performance is low that is why parallel processing is not possible

3. PROPOSED SYSTEM OVERVIEW

In our proposed system, Firstly system stores the data in NoSQL database that is MongoDB and then represents that data in the form of graph by using Neo4j. Graph representation is necessary for easy understanding where tuples are represents as node and transaction represent by edge. Graph partitioning technique is applied to find the minimum cut balanced partition so that the distribute relation between nodes should be minimal. Finally, the Decision tree classifier is applied to develop some sort of rules to take decision regarding partition strategy. On these rules the partition of data depends.

Here Schism approach is used for partitioning the data. Schism is a workload driven approach for database partitioning. In Workload driven partitioning systems, the partition of data happens based on some relation between data. The relation found based on access patterns. In transaction, the data that is mostly accessed together contains high relation, so related data comes into a single partition of database.

The architecture of the proposed system “Graph based workload driven partitioning system” is shown in figure. Our proposed system is designed to work as a Scalable transactional system on the top of distributed storage system. Here MongoDB is distributed storage system which implements CRUD (Create/Insert, Read, Update and Delete) operations.

Administrator Node is the actual server which performs partitioning by using the algorithm here Schism is using to partition the data. Shannon Information gain is calculated to find the relation between the data. Decision tree decides migration of data. Backend database is MongoDB. MongoDB is a NoSQL database store, which actually contains the data.

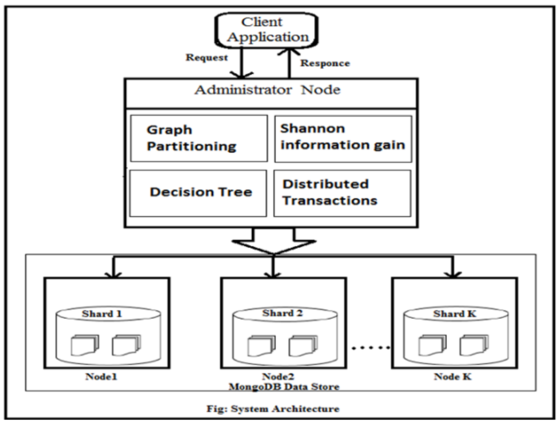

Figure 1. System Architecture

In our proposed system, the input is workload trace and a number of partitions required and the output is the balanced partitions. Because of balanced partitions the number of distributed transaction reduces and performance increases

3.1. Partition the Graph

After Graph representation, the Graph partitioning algorithm is applied to partition the Graph. Balanced partitions formed after Graph partitioning. Each particular node/tuple belongs to a database partition and each database partition allocate to a physical separated node.

3.2. Shannon Information Gain

Shannon information gain is needed for effective classification or partition of database. Shannon information gain is used to calculate the entropy/impurity of the data. Decision tree takes decision based on Shannon information gain. If two tuples has nearer Shannon information gain it means they contain some relation and come into a single partition.

3.3. Distributed Transactions

In Distributed system, the transactional data available at the different physical locations. The distribution of data should be based on some analysis or relation might reduce the number of distributed transaction. Practically Distributed systems are good choice because it provides transparency, scalability and high performance.

4. GRAPH BASED WORKLOAD DRIVEN PARTITIONING ALGORITHM

In proposed system, Schism Approach is using for partitioning the graph. Schism approach works in five steps.

4.1. Data Pre-processing Phase

4.2. Graph Representation Phase

Big data is being generated by web applications. Graph representation of data provides a proper and easy understanding, mapping and relation between data. After the data pre-processing step, whole data or workload trace is represented in the form of Graph. Each node represents a tuple and an edge represents the frequency of tuples within a transaction. Nodes, which are accessed in the same transaction is connected by an edges.

4.3. Graph Partitioning Phase

In this phase, partitioning of the graph performed. Metis Algorithm psuedcode-Metis algorithm works in three steps:

a. Coarsening- During the Coarsening phase, merging of node until the small graph is formed.
b. Initial partitioning- During the Initial partitioning phase, k-way partitioning of the smaller graph is computed.
c. Un-coarsening phase- During the Un-coarsening phase, refined partitions are formed after refinement.

4.4. Explanation Phase

In this phase, create the rules and store that rules into database. Based on these rules decision are taken. Find a compact model that captures the (tuple; partition) mappings produced by the partitioning phase. Use decision trees to produce understandable rule-based output. These rules are combination of tuple and partition (tuple: partition). That describes the tuple and stored location.

4.5. Validation Phase

5. IMPLEMENTATION DETAILS

Calculate Shannon Information Gain: Here in this step distribution factor of the input parameters are analyzed by using Shannon information gain. Shannon information gain yields distribution entity in between 0 and 1. Any values that are nearer to 1, is always consider as the highly distributed factor and it is consider as the most important factor. So this step applies info gain method on clustered data from graph based on the user and warehouse to check for the probability of the highest order warehouse with respect to the input data. For this purpose system uses the Shannon information gain theory which can be state with the help of the following equation 1:

$$IG(E) = -(P / T)\log(P / T) - (N / T)\log(N / T) \quad (1)$$

where

P	= Frequency of the warehouse entity’s present count in the clusters
N	= Non presence count
T	= Total number of clusters
IG (E)	= Information Gain for the given Entity

On applying this equation, that yields the value in between 0 to 1. The data size, that are having value nearer to 1 indicating the highest priority in the list.

Develop Decision Tree: Decision Tree extract some rules based on Shannon information gain and take decision based on that rules. Based on the graph partition clusters and Shannon information gain decision tree is created. After decision tree creation a matrix will create for the decision protocols. Matrix contains optimized routing partitioned data with respect to the warehouses. This can be representing with the following algorithm 1.

Algorithm 1: Generate Decision Tree (db, p):dt

```
// F1, F2, F3 are fitness functions
Step 0: Start
Step 1: While stop criterion is not met
Step 2: Evaluate fitness of dt attribute it to F1
Step 3: Perform x mutations to Storage Nodes
Step 4: Store F1 into F2
Step 5: While j<number of mutations to Prediction Nodes
Step 6: Select and mutate a node
Step 7: Evaluate fitness F3 of mutated tree
Step 8: If F3>F2
Step 9: Accept mutated tree
Step 10: Attribute F3 to F2
Step 11: End-if
Step 12: End-While
Step 13: If F2>F1
Step 14: Attribute resulting tree to dt
Step 15: Attribute F2 to F1
Step 16: End-If
Step 17: End-While
Step 18: Return dt
Step 19: Stop
```

6. EXPERIMENTAL SETUP

To implement the proposed system java programming language and java-based NetBeans IDE is used. For experimental prove three hardware separated machines are considered that have core i3 processor with 4GB of RAM. Here three Hardware separated machines are used for the distributed paradigm. Proposed System uses NoSQL MongoDB for the graph-based workload driven partitioning system. To stimulate the transactional workload in the form of graph the Neo4j graph database is used. The response time of developed system is validating by using TPC-C benchmark.

7. RESULT

Proposed system uses Mongo DB NoSQL database for the experiment. The developed system put under hammer in many scenarios to prove its authenticity as mentioned in below tests. To evalute performance of the system response time, throughput and number of distributed transactions are considering.

When the system is subjected to get the throughput for 10 warehouses an experiment is conducted to get the same, whose result is tabulated in Table 1

Table 1. Throughput and Time Response for 10 Warehouse

No of Transcations	Single DB Time in Seconds	Multiple DB Time in Seconds	Throughput for Single DB (No of Transactions per Seconds	Throughput for Multiple DB (No of Transactions per Seconds
20	8.547	5.47	2.34000234	3.65630713
40	18.789	8.482	2.12890521	4.715868899
60	26.771	15.774	2.241231183	3.803727653
80	37.44	20.475	2.136752137	3.907203907

When a plot is drawn for this throughput results we get some interesting facts that can be seen Figure 3. Where we can observer that throughput is increased sweepingly for the multiple DB transactions. That means on distribution of the databases number of transactions per second will increase to yield better results

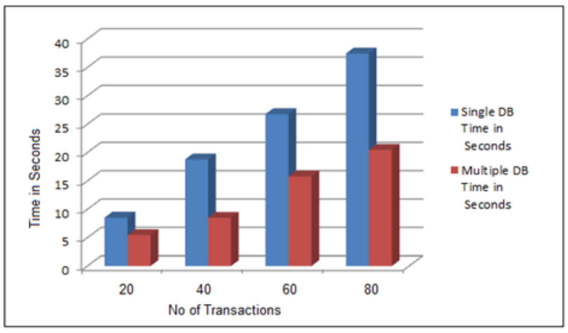

Figure 2. Time Delay Comparison for 10 Warehouses

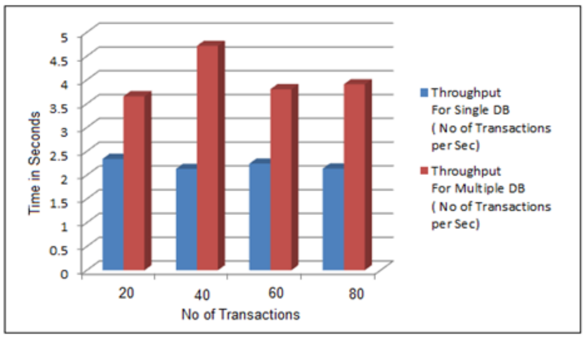

Figure 3. Throughput Comparison for 10 Warehouses

When the system is subjected to get the throughput for 15 warehouses an experiment is conducted to get the same, whose result is tabulated in Table 2.

Table 2. Throughput and Time Response for 15 Warehouses

No of Transcations	Single DB Time in Seconds	Multiple DB Time in Seconds	Throughput for Single DB (No of Transactions per Seconds	Throughput for Multiple DB (No of Transactions per Seconds
20	8.78	4.98	2.277904328	4.016064257
40	17.77	7.87	2.250984806	5.082592122
60	21.58	13.5	2.780352178	4.444444444
80	34.25	16.52	2.335766423	4.842615012

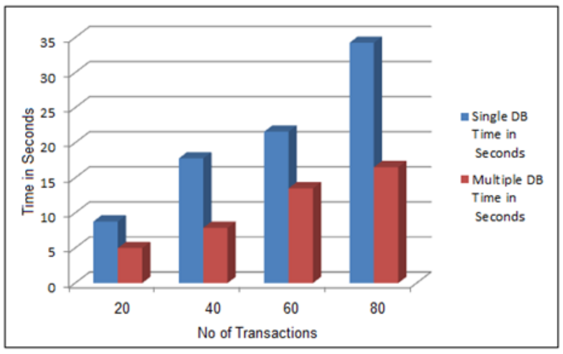

Figure 4. Time Delay Comparison for 15 Warehouses

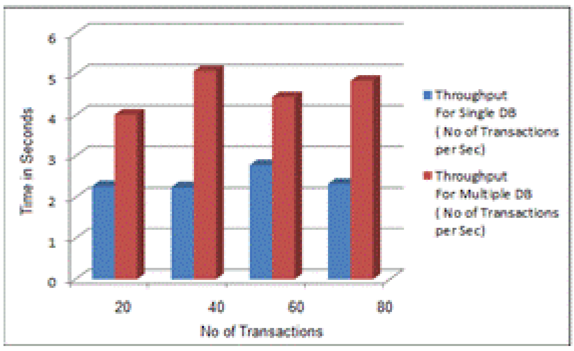

Figure 5. Throughput Comparison for 15 Warehouses

When the system is subjected to get the throughput for 20 warehouses an experiment is conducted to get the same, whose result is tabulated in Table 3.

Table 3. Throughput and Time Response for 20 Warehouses

No of Transcations	Single DB Time in Seconds	Multiple DB Time in Seconds	Throughput for Single DB (No of Transactions per Seconds	Throughput for Multiple DB (No of Transactions per Seconds
20	7.44	4.4	2.688172043	4.545454545
40	15.66	7.01	2.554278416	5.706134094
60	19.22	11.2	3.121748179	5.357142857
80	32.02	15	2.498438476	5.333333333

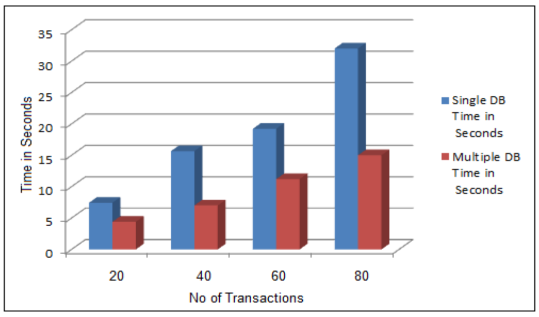

Figure 6. Time Delay Comparison for 20 Warehouses

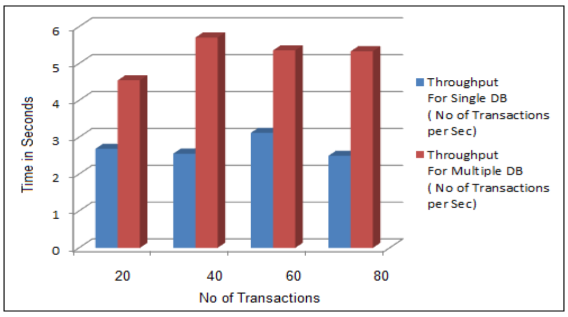

Figure 7. Throughput Comparison for 20 Warehouses

When the system is subjected to get the throughput for 25 warehouses an experiment is conducted to get the same, whose result is tabulated in Table 4.

Table 4. Throughput and Time Response for 25 Warehouses

No of Transcations	Single DB Time in Seconds	Multiple DB Time in Seconds	Throughput for Single DB (No of Transactions per Seconds	Throughput for Multiple DB (No of Transactions per Seconds
20	6.21	2.2	3.220611916	9.090909091
40	12.1	6.4	3.305785124	6.25
60	17.2	10.24	3.488372093	5.859375
80	29.2	13	2.739726027	6.153846154

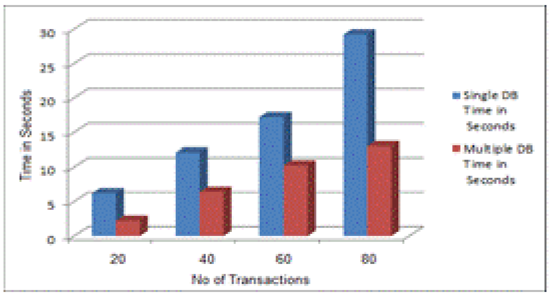

Figure 8. Time Delay Comparison for 25 warehouses

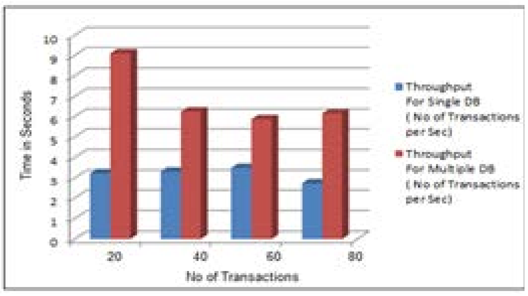

Figure 9. Throughput comparison for 25 warehouses

8. CONCLUSION

In proposed algorithm, graph based workload driven partitioning system for NoSQL database is presented. Workload is represented in the form of graph to take the advantage of the relationship between data. Based on that relationship tuples are combined together into a single partition. Partitioning based on relation reduces the number of distributed transactions and increase Throughput. TPC-C benchmark is used for validation of proposed system. Proposed system improves the response time and throughput, which is considerably less than the TPC-C schema. Hence, proposed approach shows better results in terms of distributed transactions and throughput.

REFERENCES

[1] Das, Sudipto, Divyakant Agrawal, and Amr El Abbadi. "ElasTraS: An elastic, scalable, and self-managing transactional database for the cloud." *ACM Transactions on Database Systems (TODS)* 38.1: 5, 2013.
[2] Alex Pothen, H. D. Simon, Lie Wang, and Stephen T. Bernard. “Towards a fast implementation of spectral nested dissection”. In *Supercomputing ’92 Proceedings*, pages 42–51, 1992.
[3] Alex Pothen, Horst D. Simon, and Kang-Pu Liou. “Partitioning sparse matrices with eigenvectors of graphs”. *SIAM Journal of Matrix Analysis and Applications*, 11(3):430–452, 1990.
[4] Gary L. Miller, Shang-Hua Teng, and Stephen A. “Vavasis. A unified geometric approach to graph separators”. In *Proceedings of 31st Annual Symposium on Foundations of Computer Science*, pages 538–547, 1991.
[5] Curino, Carlo, et al. "Schism: a workload-driven approach to database replication and partitioning." *Proceedings of the VLDB Endowment* 3.1-2: 48-57, 2010.
[6] Karypis, George, and Vipin Kumar. "METIS unstructured graph partitioning and sparse matrix ordering system, version 2.0.", 1995.
[7] Tatarowicz, Aubrey L., et al. "Lookup tables: Fine-grained partitioning for distributed databases." *2012 IEEE 28th International Conference on Data Engineering.IEEE,* 2012.
[8] Quamar, Abdul, K. Ashwin Kumar, and AmolDeshpande. "SWORD: scalable workload-aware data placement for transactional workloads." *Proceedings of the 16th International Conference on Extending Database Technology.ACM,* 2013.
[9] Wang, Lu, et al. "How to partition a billion-node graph." *2014 IEEE 30th International Conference on Data Engineering.IEEE*, 2014.
[10] Ding, Chris HQ, et al. "A min-max cut algorithm for graph partitioning and data clustering." *Data Mining, 2001.ICDM 2001, Proceedings IEEE International Conference on IEEE*, 2001.
[11] Choe, Tae-Young, and Chan-Ik Park. "A k-way graph partitioning algorithm based on clustering by eigenvector." *International Conference on Computational Science*. Springer Berlin Heidelberg, 2004.APA.
[12] LaSalle, Dominique, and George Karypis. "A Parallel Hill-Climbing Re_nement Algorithm for Graph Partitioning." 2015.
[13] Predari, Maria, and Aur_elienEsnard. "Coupling-aware graph partitioning algorithms: Preliminary study." 2014 21st *International Conference on High Performance Computing (HiPC)*. IEEE, 2014.
[14] McCa_rey, James D. "Graph partitioning using a Simulated Bee Colony algorithm." *Information Reuse and Integration (IRI), 2011 IEEE International Conference on IEEE*, 2011.