Intelligent Automation & Soft Computing

Tech Science Press

DOI: 10.32604/iasc.2023.035503
Article

Hybrid Graph Partitioning with OLB Approach in Distributed Transactions

Rajesh Bharati* **and Vahida Attar**

Computer Engineering Department, College of Engineering, Pune, 411005, India
*Corresponding Author: Rajesh Bharati. Email: rdbharati@gmail.com
Received: 24 August 2022; Accepted: 13 January 2023

Abstract: Online Transaction Processing (OLTP) gets support from data partitioning to achieve better performance and scalability. The primary objective of database and application developers is to provide scalable and reliable database systems. This research presents a novel method for data partitioning and load balancing for scalable transactions. Data is efficiently partitioned using the hybrid graph partitioning method. Optimized load balancing (OLB) approach is applied to calculate the weight factor, average workload, and partition efficiency. The presented approach is appropriate for various online data transaction applications. The quality of the proposed approach is examined using OLTP database benchmark. The performance of the proposed methodology significantly outperformed with respect to metrics like throughput, response time, and CPU utilization.

Keywords: Data–partitioning; scalability; optimization; throughput

1 Introduction

Data partitioning is always done first when creating a database for an application. To store and handle the data, a database system must be developed in distribution. For large workloads to perform well, partitioning method is essential [1]. The primary benefit of data partitioning is storing large data based on their category [2]. Moreover, the data partitioning process improves the management, performance, and accessibility of many applications. Furthermore, the partitioning process decreases owners' total expense for storing large data [3]. Data partitioning is utilized to access the data at any time. The effective data partitioning process makes it more accessible from the partitions [4]. The existing data partitioning methodologies are hash partitioning, range partitioning, horizontal partitioning and so on [5,6]. In a distributed framework, data partitioning represents the RDF data. This is based on the arrangement of data in distributed networks [7].

Nowadays, the OLTP database is the widely utilized data processing system. In OLTP systems, scalability is the central issue because of the increased online transactions. The scalability in data transactions can increase the transaction throughput [8,9]. The general examples of OLTP applications are the entry of orders, sale retails and economic transactions. The essential characteristics of OLTP systems are lesser response time, the large many user applications, and simple transactions. In ElasTras [9] data partitioning is done on the schema level to achieve scalability. The TPC-C benchmark with two

distinct metrics, such as response time and throughput, is evaluated at the schema level. Sauer et al. [10] represented partitioning using a data mining approach on a horizontal database at a cloud data store. In analysis of database cluster logs are monitored by data mining techniques. The method compares the average result of a round-robin, k-means and no partition techniques.

The OLTP-based applications are subject to unexpected variations in workloads [11]. Hence, the optimal data partitioning approach should support the unpredicted workloads of OLTP applications [12,13]. The partitioning approach is utilized to reduce the performance degradation for multiple locations of transactions [14,15]. Moreover, every transaction is related, enabling the transactions to run locally. Generally, the relational data is partitioned into horizontal and vertical manner. In horizontal partitioning, the data tables are separated into a set of tuples. In vertical partitioning, data tables are separated into set of disjoint columns. However, the horizontal and vertical partitioning approaches are inefficient in real-world applications [16]. The existing methodologies in data partitioning are Hash, Round-robin, List and Range partitioning [17]. These approaches are not capable for the increasing number of transaction loads. Furthermore, these types of partitioning approaches are not considered the relation among the data at the time of data partitioning. But, effective partitioning should split the data with their associated pattern. Only some of the existing approaches partition the data better but need to proportion the related data items [18]. Therefore, an effective data partitioning approach must consider both effective data partitioning with most associated data items [19]. This kind of partitioning decreases the undesirable effects of the transactions. The existing data partitioning approaches depend on the analysis of statistical and classification examination. The limitation of these approaches is that they do not evaluate the relevance of data in the partitions [20].

Instead of keeping whole data table in one site, table is converted into number of fragments called partitions. Generally, hash, range, horizontal or vertical partitioning is utilized, and these types of partitions are not effective. So, the hybrid graph partitioning approach is presented with simultaneous graph and vertical partitions. Here, the sub partitioning is based on related data in column form. As a result of each partition containing its related data, this technique makes data access useful. Moreover, an innovative data partitioning approach and optimized load balancing for scalable transactions are presented. TPC Benchmark E (TPC-E) is an OLTP database benchmark. The presented approach partitions the TPC-E dataset into the hybridized form of vertical and graph partitioning. Here, the partitions and the sub partitions are associated with the related data contents and thereby the data can be retrieved from the partitioned database effectively. This process decreases the response time of data transactions.

A hybrid graph data partitioning algorithm is proposed to efficiently fragment the database. An improved load balancing approach is proposed to balance the workload transactions in the data partitioning. The performance of the developed method is enhanced by scalable workload transactions. For many applications that require online data transactions, the provided technique is appropriate. Additionally, the TPC-E workload dataset is used to evaluate the effectiveness of the partitioning with distributed transactions that is being provided.

The organization of this paper is described as: Section 2 reviews the recent related works, Section 3 provides the detailed description about the presented methodology, Section 4 illustrates mathematical background, Section 5 represents results, and the paper is concluded in Section 6.

2 Related Work

Numerous recent inventions have been made by researchers that offer a consistency guarantee on a data feature. Performance, scalability, and consistency are the main goals of such a system. The data

objects accessed by transactions are retained on a single partition in graph partitioning. Schism [3] is an automatic partitioning model it minimizes distributed transactions. This work provides a cost model for processing a transaction that believes the number of partitions the transaction needs to access and the overall skewness of data access. ElasTras [9] does data partitioning at the schema level to achieve scalability. The TPC-C benchmark was assessed using two distinct measures, such as response time and throughput. Gao et al. [21] developed a partitioning methodology for scheduling the query in the distributed dataset. Here, the introduced method considered the TPC-H dataset for partitioning the data. The queries are partitioned according to the expense of that data. The genetic optimization approach is utilized to decrease the difficulties in data partitions. The Alco approach is designed on cost based fragments.

Zhang et al. [22] developed a workload-based data partitioning and replication methodology named as Apara. Here, two heuristic approaches were presented for effective data partitioning and replication. Moreover, two variant cost models take place for optimal data partitions. The performance of the data partitioning is examined with the TPC-H dataset. The presented Apara model provides better outcomes with the TPC-H workloads. Various distributed data storages are compared in [23] with regard to data replication, concurrency control mechanisms, key access, and data partitioning.

Zhu et al. [24] developed an approach for optimal ordering of data in numerous warehouses to reduce data partitioning. Here, k-links heuristic clustering methodology was presented to categorize the data optimally among various partitions. The approach represents the distribution of data items to be decrease in number of partitions. The accurate data partition reduces the complexity of the data transactions. This methodology was utilized for the online data partitioning. The major limitation of the developed approach was not providing load balancing with the data partitions.

Ahirrao et al. [25] presented an innovative workload based data partitioning methodology for scalable data transactions. This developed scalable data transaction was utilized for NoSQL cloud data stores. The TPC-C dataset is used to evaluate the provided methodology, which operates on scalable data. In [26] the CPU utilization efficiency performance is examined with the different existing methodologies like RCD+ (Run time correlation discovery), RCD, and TADs (Temporal approximate dependencies). In the study [27,28] design variables are shown to be in their best possible states, and multi-objective optimization is used to increase the normalized values of mass, maximum deflection and collapsing load. In [29] Data is seamlessly rearranged using a real time reconfiguration approach, with minimal performance impact on transactions. Plans for reconfiguration prioritize using already-existing data replicas while copying data asynchronously and concurrently from several replicas to minimize data transfer. A secure query processing approach to ensure access control policies safe when querying data from distributed partitions [30]. The suggested method uses vertical partitioning to create a collection of secure sub-schemas that are kept in distinct partitions in the distributed system.

The load balancing strategies and a review of the current data partitions have been addressed. All the existing approaches are utilized different forms of data partitioning. But the major drawback of existing approach is not considered better load balancing with the data partitions. Some of the approaches provide good partitions but fail in load balancing. This affects the performance of the system. Hence, random partitioning also affects the performance and increases the system's complexity and time. Therefore, the presented approach overcomes the issues in the existing and improves the performances with the combination of hybrid graph partitioning and improved load balancing.

3 Proposed Methodology

Data partitioning is used to implement the database's scalability. Multiple partitions will be designed from one table. When the data scales very large, we can add more nodes to store and analyze some partitions. Additionally, it will increase the system's ability to store and process. The primary goal of this proposed work is to partition workload with Hybrid Graph Partitioning (HVP) and optimized load balancing. Initially, the data items in the datasets are partitioned through the hybrid graph partitioning approach. Afterward, the partitions efficiency and average partition loads are computed. Furthermore, the weight factor is calculated based on these estimated measures. Finally, optimized load balancing is performed by updating the computed weight factor. This methodology effectively balances the load and provides optimized data partitions. This optimized load balancing enhances the performance of the data partitions. Fig. 1 shows a schematic diagram of the proposed methodology. Here, the weight factor is calculated based on the partition efficiency and the average partition load. Efficiency is computed on distributed transactions and the total number of transactions executed in partitions. It updates effective load balancing and reduces distributed transactions. In terms of parameters the load is computed by the total amount of transactions processed on the data partition. Efficiency depends on distributed transactions and total transactions. The weight factor is calculated based on the average partition load and the partition efficiency.

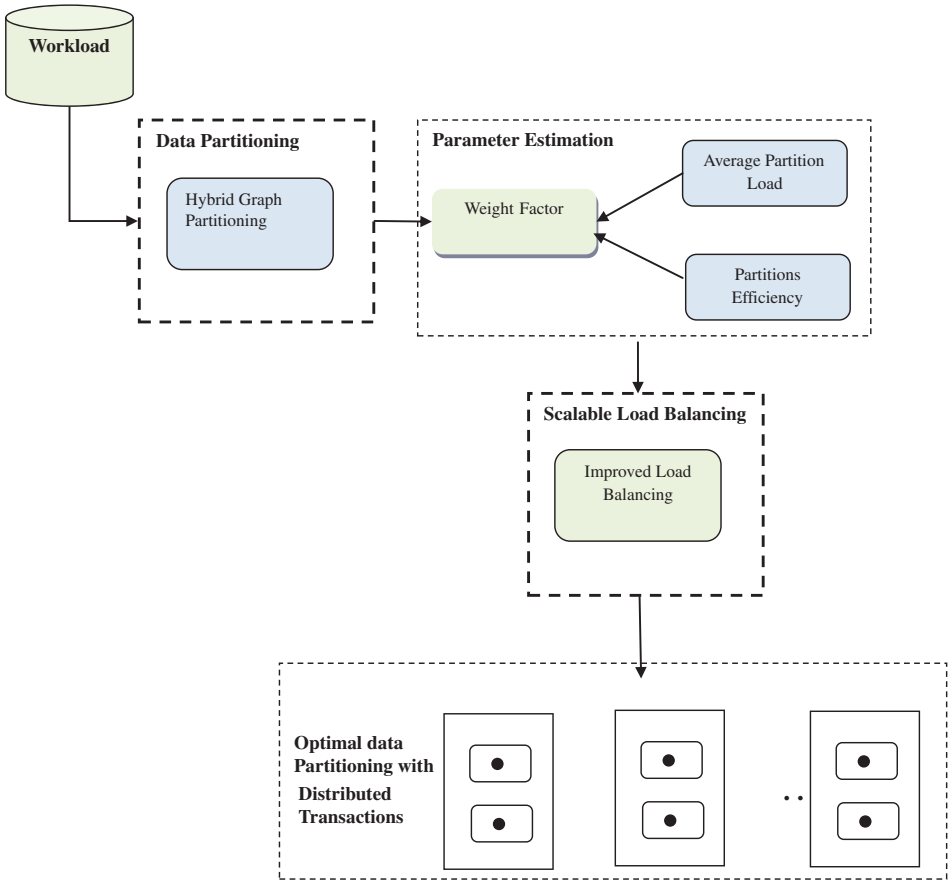

Figure 1: Schematic diagram of Hybrid Graph Partitioning and Optimized Load Balancing Partitions

3.1 Hybrid Graph Partitioning (HVP)

The main importance of using vertical partitioning is to decrease the complexity. Moreover, this partitioning decreases the performance cost of frequent access of data items. To maintain secure data, the most significant data is partitioned differently. Data partitioning is the distribution of a logical database or its subcomponents into separate, independent segments.

In vertical partitioning, the input database table ($\overline{T}$) is partitioned into number of tables ($\overline{T}_1, \overline{T}_2, \overline{T}_3 \ldots . \overline{T}_N$). Moreover, the attributes present in the tables are $\overline{A}_1, \overline{A}_2, \overline{A}_3 \ldots . \overline{A}_N$ respectively. The partitioning expressions are described as,

$$\overline{A}_1 \cup \overline{A}_N = \overline{A} \tag{1}$$

$$\overline{A}_1 \cap \overline{A}_N = \overline{A}\ \overline{K}_a \tag{2}$$

$$\left|\overline{A}_1\right| > \left|\overline{A}\ \overline{K}_a\right| \tag{3}$$

$$\left|\overline{A}_N\right| \geq \left|\overline{A}\ \overline{K}_a\right| \tag{4}$$

where, attributes union and intersection is determines frequently access data and $\overline{A}\ \overline{K}_a$ signifies the sub-set which consists of all the key attributes of table $\overline{T}$. After this partitioning process, the data table $\overline{T}$is changed into number of partitioned tables $\overline{T}_1, \overline{T}_2, \overline{T}_3 \ldots . \overline{T}_N$. By utilizing the vertical partitioning procedure, regular table is generated and one for every sub tables. Eq. (4) represents total numbers of derived data items from datasets are greater than total number of records. The workload in table $\overline{T}$ is split into sub tables. If any of the queries relate to a non-key attribute, the specific result is modified with the subsequent alternative forms.

- **First alternative:** Consider one query corresponding sub table is $\overline{T}_1$, $\left(\overline{T}_1, \Delta_k, ID_k\right)$ then all the attributes are in $\overline{A}_1$. It is represented as $\left(\Delta_k \subset \overline{A}_1\right)$.
- **Second alternative:** Consider the query which corresponding to sub table $\overline{T}_2$, $\left(\overline{T}_2, \Delta_k, ID_k\right)$, then all the attributes are in $\overline{A}_2$. It is represented as $\left(\Delta_k \subset \overline{A}_2\right)$.

If two queries are in both cases, then the tables contain both attributes. The vertical partitioning is partition the data based on column partitions. In graph based data partitioning, data table is associated with the related contents. The graph partitioning is directly partition the data table into $\overline{k}$ set of tables. In a graph, each row acts as a node, and edges connect tuples that represent transactions. In the presented data partitions, both vertical and the graph based partitioning approach are hybridized for effective data partitioning.

In hybrid graph data partitioning, data items in the table are considered as a sub-tuples. Initially, graph is generated with a number of data items per each column and tuple. Then the edges between the tuples are admitted using the similar transaction data. Afterwards, the graph partitioning approach partitions the data graph into number of $\overline{k}$ balanced partitions. This graph partitioning approach decreases the amount of cross partition transactions.

The graph partitioning is depending on the quantity of the data items and the size of partitions. Data partitions are obtained based on the uniform load distribution. The size of partitions is computed through the subsequent condition (5).

$$Sp_{\min} = \frac{\overline{S}_{total}}{\overline{N}_C} * \overline{\alpha} \tag{5}$$

$$Sp_{\max} = \frac{\overline{S}_{full}}{\overline{N}_C} * \overline{\beta} \tag{6}$$

$$\overline{\alpha} + \overline{\beta} = 2 \tag{7}$$

where, $Sp_{\min}$ signifies the minimum size of partition, Sp_{max} represents the maximum size of partitions, $\overline{S}_{total}$ represents the total size of data in the dataset, $\overline{\alpha}$ and $\overline{\beta}$ are the variables that adjusting the minimum and maximum size of partitions respectively. Depends on size of the data partitions made with balanced workload.

The structure of hybrid graph partitioning is depicted in Fig. 2. The data partitioning with the presented hybrid graph partitioning provides the effective partitioning of data with their attributes. This provides the optimal partitions with their associated sub data. This process of partitioning makes easier the data transactions. The presented data partitioning reduces the complexity and lesser the response time. After the completion of data partitioning using the hybrid graph, the improved load balancing optimization approach is presented to balance the load effectively.

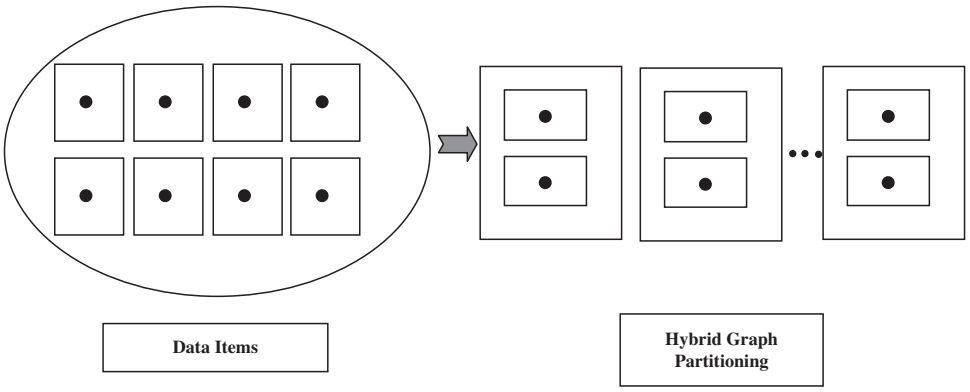

Figure 2: Data partitioning with hybrid graph partitioning

3.2 Optimized Load Balancing Partitions Using ILB Algorithm (OLBP)

The optimized load balancing is achieved in the data partitions using improved load balancing methodology. In the presented load balancing methodology, the partitioned data tables and the workloads are considered as an input. The partitions efficiency and the average load are computed, and their weight factor is computed. Afterwards, optimal data transactions in the data partitions are performed by updating the computed weight factor. The considered data items are represented as $\overline{d}_1, \overline{d}_2, \overline{d}_3 \ldots . \overline{d}_m$. The parameters applied for load balancing in the subsequent estimation.

4 Mathematical Background

The efficiency of the partitioning approach is computed through the subsequent condition (8).

$$\overline{P}_{eff} = 1 - \frac{\overline{t}_{distributed}}{\overline{T}_{total}} \tag{8}$$

Here, $\overline{P}_{eff}$ signifies the efficiency of data partitions, $\overline{t}_{distributed}$ signifies the distributed transactions, and $\overline{T}_{total}$ signifies the total number of transactions.

In workload aware data partitioning, data partitions are formed according to the amount of loads. Load is computed by the total amount of transactions processed on the that partition. This is computed by the subsequent condition (9),

$$\overline{P}(N) = \sum_{j=1}^{M} Sp_{max} \tag{9}$$

Here, N signifies the number of partitions, M signifies the number of loads in one partition. Then the average load is computed based on the total amount of transactions performed on all the partitions. The average partition load is computed through the subsequent condition (10),

$$\overline{P}(avg) = \frac{\overline{P}(N)}{N} \tag{10}$$

Here, $\overline{P}(avg)$ signifies the average load of the partitions, N signifies the partition number. Then, the weight factor is computed based on the partition efficiency and the average partition load. The computed weight factor is updated to effectively balance the loads in the partitions.

Algorithm 1: Pseudo code of Improved load balancing with data partitions

Improved load balancing optimization (ILB) algorithm
Begin
For each partition do
Compute partition efficiency,
Average partition load
End
Foreach partition do
Compute weight factor
End
Sort the partitions load in ascending order
Repeat
Until partitions end
Choose the top ranked partitions
Until partitions end
Return the top partitions as the best partitions with effective load balancing
End

In scalable workload based partitioning, performance is enhanced with distributed transactions. Here, the partitioned data is given as input to the load balancing. This optimized load balancing balances workloads when increasing the number of transactions. The presented load balancing approach considers the efficiency of partitions and the average partition load. The weight factor is computed based on the average partition load and the partition efficiency. The optimal weight factor based on the data partitions is computed through the subsequent condition (11),

$$\overline{W}_F = \frac{\overline{P}_{eff} + \overline{P}(avg)}{2} \tag{11}$$

Here, $\overline{W}_F$ signifies the weight factor, $\overline{P}_{eff}$ signifies the partitions efficiency, and $\overline{P}(avg)$ signifies the average partition load. The workload is ordered and updated in the table partitions effectively based on the computed weights. The main objective of data partitioning is the formation of partitions to

maximize the efficiency of the data transactions. The effective load balancing is attained by utilizing the average load of partitions. The presented optimized load balancing approach provides the optimal load balancing for large workloads. Output data of Hybrid graph partitioning approach is given as an input to improved load balancing optimization. Initially, partition efficiency and the average value of the partition load are computed, and the corresponding weight factors are computed for effective load balancing. Based on the computed weight factor value, optimization algorithm prioritizes the partitions and optimal load balancing is attained. The pseudo code of improved load balancing optimization is provided in proposed algorithm.

The increasing number of loads in the environment reduces the performance of the system. Hence, effective load balancing is performed to balance the workloads optimally. Data partitioning and load balancing together improve performance throughput and response time.

5 Experimentation

This section examines the experimental results of the presented workload aware data partitioning methodology. The presented methodology is implemented with MYSQL, HBase [31], and DynamoDB with the TPC-E benchmark dataset [32]. This dataset provides a large amount of data records for OLTP applications. The presented data partitioning performance is compared with the existing approaches to prove its efficiency.

5.1 Dataset Description: TPC-E Dataset

TPC-E is standard benchmark dataset utilized for performing workloads of various applications. A wide variety of businesses, from banks and grocery stores to online E-Commerce websites and financial markets, rely mainly on on-line transaction processing (OLTP). OLTP has been a significant target for optimization for computer manufacturers, database software vendors, system software vendors, and the corresponding research communities due to its importance. This dataset contains the data records of Customers, Market, and Brokerage.

5.2 Performance Metrics

This section includes a number of significant performance metrics, including distributed transactions, throughput, response time, and CPU utilization. These performance metrics are described in the subsequent sub sections.

5.2.1 Throughput

This performance measure is evaluated by the proportion of total data transactions to the taken time. Throughput performance is computed in transactions per second. This performance provides the amount data transmission rate per second. The efficiency in throughput provides the effective transmission data per each second. The throughput performance is computed by the subsequent condition (12),

$$\overline{Th} = \frac{R_{data}}{t'} \tag{12}$$

Here, $\overline{Th}$ signifies the throughput measure, R_{data} signifies the data rate and t' signifies the time.

5.2.2 Response Time

Response time is evaluated based on the data transactions. The time taken to respond the data transaction is represented as response time. It is computed by the subsequent condition (13),

$$\overline{R}_{time} = \frac{\overline{d}_r}{t'} \tag{13}$$

Here, $\overline{R}_{time}$ signifies the response time, $\overline{d}_r$ signifies the retrieving data or the process of task and t' signifies the time taken to process the task. The response time of the transaction should be lesser for effective performance. If the response time of the transaction is higher, then the performance of the system degrades.

5.2.3 Distributed Transactions

Distributed transaction represents the group of operations on data that is performed between large set of data. A distributed transaction is one that occurs across two or more distinct partitions. Transaction is the combination of read and writes operations in the dataset for the evaluation. It is computed by the subsequent condition (14),

$$\overline{D}_t = \frac{\overline{L}_T}{P_{data}} \tag{14}$$

Here, $\overline{D}_t$ signifies the distributed transaction, P_{data} signifies the data transactions and $\overline{L}_T$ signifies the workload transactions.

5.2.4 CPU Utilization

This performance evaluation is computed based on the utilization of the CPU memory for processing of the presented approach. The OLTP workload based data transactions needs high storage systems for the processing of their data. The efficiency in the CPU utilization is improved by providing the optimal form of data transactions. It is computed by the subsequent condition (15),

$$\overline{U}_{CPU} = 100\% - \left(\%t'_{load}\right) \tag{15}$$

Here, $\overline{U}_{CPU}$ signifies the CPU utilization and t'_{load} signifies the percentage of time taken to process the load.

5.3 Performance Examination

In this section, the performance of presented data partitioning with optimized load balancing is examined. The performance of the presented approach in regards of throughput is depicted in Fig. 3.

In Fig. 4, the presented approach with distributed transactions throughput by varying number of concurrent users Here, the response time of the presented approach is examined by varying number of users like 1000, 2000, 3000, 4000 and 5000 correspondingly. This proved that the presented approach takes lesser response time than the different existing approaches. Moreover, the performance comparison on data partitions is depicted in Fig. 5.

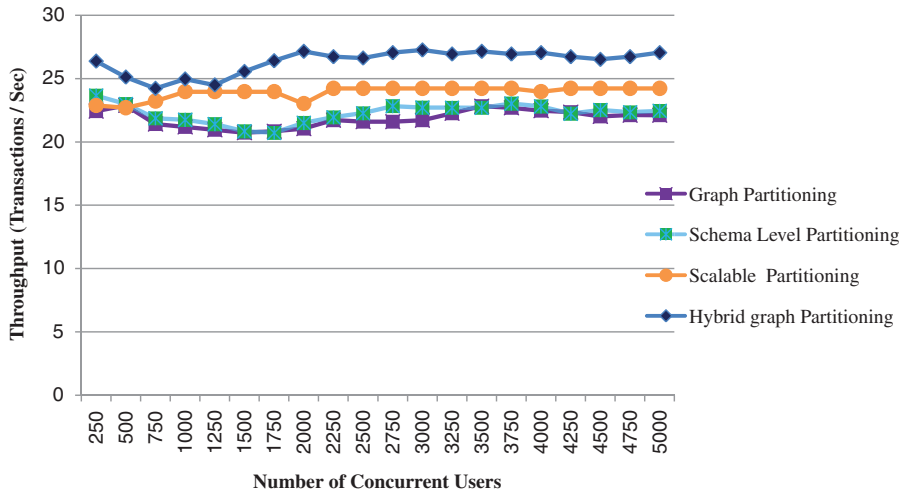

Figure 3: Hybrid graph partitioning throughput

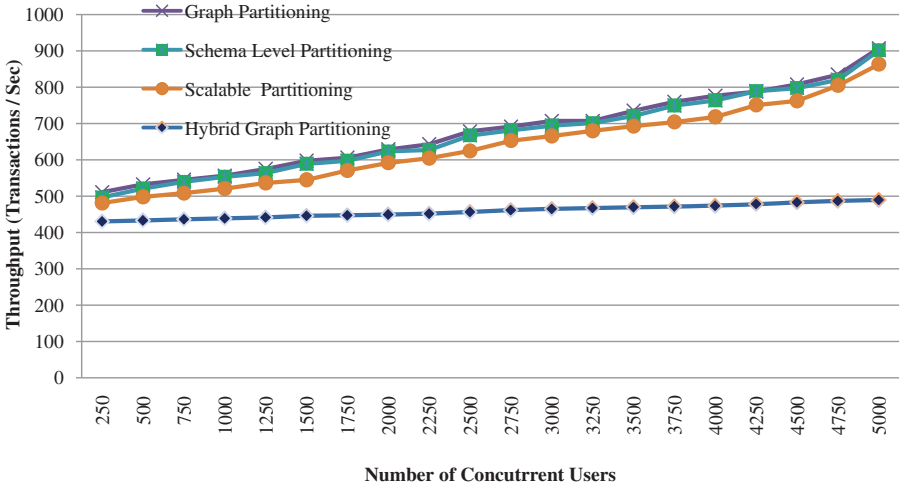

Figure 4: Distributed transactions by varying number of concurrent users

In Fig. 4, the performance validation on response time is compared. Here, the presented approach response time performance is examined with the different existing methodologies like schema level data partitioning, scalable workload driven data partitioning, and graph partitioning. Here, the response time of the presented approach is examined by varying number of user workloads like 1000, 2000, 3000, 4000 and 5000 correspondingly. This proved that the presented approach taking lesser response time than the different existing approaches. Moreover, the performance comparison on data partitions is depicted in Fig. 5.

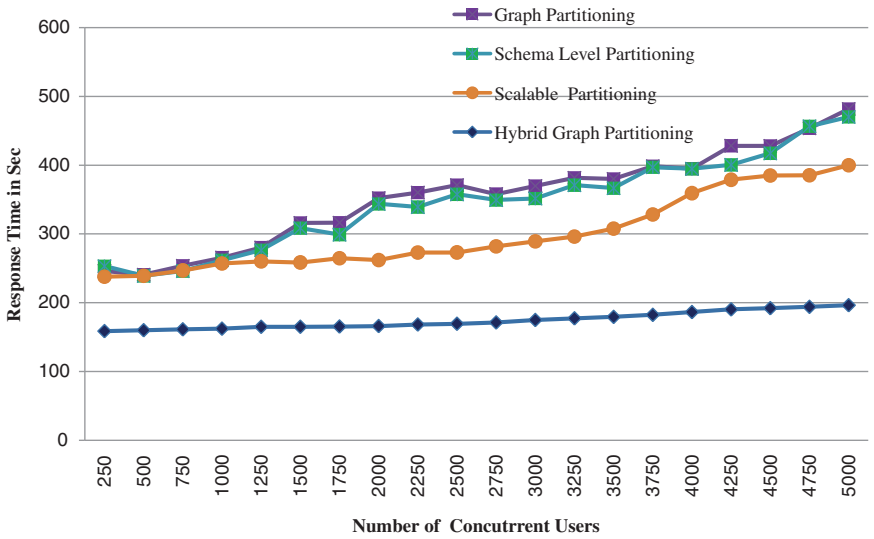

Figure 5: Response Time through Hybrid Graph partitioning

The performance of the presented approach is compared with the different data partitioning approaches. This proved that the presented hybrid graph partitioning approach provides the improved performance than the different existing approaches like horizontal, vertical and no partitioning approaches. The transactions range is enhanced by the presented hybrid graph approach. Moreover, the performance comparison on distributed transactions is depicted in Fig. 5. The performance of the presented approach is evaluated with distributed transactions. The presented methodology performance is examined with the various existing methodologies like graph partitioning, schema level based data partitioning, and scalable workload based data partitioning. Here, the distributed transactions are validated through the varying number of users like 1000, 2000, 3000, 4000 and 5000 correspondingly. The efficiency in CPU utilization will improve the performance of the OLTP data transactions. Generally, the OLTP transactions require high data storage for keeping large amount of data records. The efficiency of the CPU utilization will enhance the performance of data transactions.

6 Conclusion

This paper presented an effective hybrid graph data partitioning with optimized load balancing. Initially, the data items are partitioned into effective partitions by utilizing hybrid graph partitioning. Then the partitions efficiency and the average load of the partitions are computed with the corresponding weight factor. Furthermore, optimized load balancing is provided to prove the performance of data transactions. The presented approach is validated with the standard OLTP benchmark dataset. The performance of the presented approach is examined, and the performance of the proposed methodology significantly outperformed them in terms of metrics like throughput, response time, distributed transactions, and CPU utilization.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] Q. Waseem, M. Maarof, M. Idris and A. Nazir, "A taxonomy and survey of data partitioning algorithms for Big data distributed systems," in *Proc. 3rd Int. Conf. on Micro-Electronics and Telecommunication Engineering*, Ghaziabad, India, pp. 447–457, 2020.

[2] R. D. Bharati and V. Z. Attar, "Performance analysis of scalable transactions in distributed data store," in *Proc. Int. Conf. on Computing in Engineering & Technology*, vol. 303, Lonere, India, pp. 542–548, 2022.

[3] C. Curino, E. Jones, Y. Zhang, and S. Madden, "Schism: A workload-driven approach to database replication and partitioning," in *Proc. VLDB Endowment*, Singapore, vol. 3, no. 1, pp. 48–57, 2010.

[4] M. Zhang,Y. Zhuo, C. Wang, M. Gao, Y. Wu *et al.,* "GraphP: Reducing communication for PIM-based graph processing with efficient data partition," in *Proc. IEEE Int. Symp. on High Performance Computer Architecture*, Vienna, Austria, pp. 544–557, 2018.

[5] A. A. Haneen, A. Noraziah, R. Gupta and M. A.Fakherldin, "Review on data partitioning strategies in Big data environment," *Advanced Science Letters*, vol. 23, pp. 11101–11104, 2017.

[6] M. Boissier and K. Daniel, "Workload-driven horizontal partitioning and pruning for large HTAP systems,"in *Proc. Int. Conf. on Data Engineering Workshops*, Paris, France, pp. 116–121, 2018.

[7] I. Abdelaziz, R. Harbi, Z. Khayyat and P. Kalnis, "A survey and experimental comparison of distributed SPARQL engines for very large RDF data," in *Proc. VLDB Endowment*, Munich, Germany, pp. 2049–2060, 2017.

[8] G. Prasad, A. Cheung and D. Suciu, "Handling highly contended OLTP workloads using fast dynamic partitioning," in *Proc. ACM SIGMOD Int. Conf. on Management of Data*, Portland, USA, pp. 527–542, 2020.

[9] S. Das, D. Agrawal and A. El Abbadi, "Elastras: An elastic, scalable, and self managing transactional database for the cloud," *ACM Transactions Database Systems*, vol. 38, no. 1, pp. 1–45, 2013.

[10] B. Sauer and W. Hao, "Horizontal cloud database partitioning with data mining techniques," in *Proc. 12th Annual IEEE Consumer Communications and Networking Conf.*, Las Vegas, NV, pp. 796–801, 2015.

[11] R. Taft, N. El-Sayed, M. Serafini,Y. Lu, A. Aboulnaga *et al.,*"P-Store: An elastic database system with predictive provisioning," in *Proc. Int. Conf. on Management of Data*, Houston, USA, pp. 205–219, 2018.

[12] Y. Sheng, A. Tomasic, T. Zhanga and A. Pavlo, "Scheduling OLTP transactions via learned abort prediction," in *Proc. Int. Workshop on Exploiting Artificial Intelligence Techniques for Data Management*, Amsterdam, Netherland, pp. 1–8, 2019.

[13] K. Kaur and V. Laxmi, "A novel method of data partitioning using genetic algorithm workload driven approach utilizing machine learning," *Cognitive Computing in Human Cognition*, vol. 17, pp. 49–60, 2020.

[14] A. 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, pp. 106–118, 2016.

[15] Z. Wei, G. Pierre and C. Chi, "CloudTPS: Scalable transactions for web applications in the cloud," *IEEE Transactions on Services Computing*, vol. 5, pp. 525–539, 2012.

[16] G. A. Schreiner, D. Duarte, G. D. Bianco and R. D. S. Mello, "A hybrid partitioning strategy for NewSQL databases: The VoltDB case," in *Proc. Int. Conf. on Information Integration and Web-Based Applications & Services*, Munich, Germany, pp. 353–360, 2019.

[17] J. Lee, K. Kim, H. Lee, M. Andrei and S. Ko, "Asymmetric-partition replication for highly scalable distributed transaction processing in practice," in *Proc. of the VLDB Endowment*, Tokyo, Japan, vol. 13, pp. 3112–3124, 2020.

[18] W. Lim, P. Ahmed and M. Ali, "Data and resource maximization in business-to-business marketing experiments: Methodological insights from data partitioning," *Industrial Marketing Management*, vol. 76, pp. 136–143, 2019.

[19] S. Goyal, P. Bedi, A. Rajawat, R. Shaw and A. Ghosh, "Multi-objective fuzzy-swarm optimizer for data partitioning," in *Proc. Advanced Computing and Intelligent Technologies*, New Delhi, India, pp. 307–318, 2022.

[20] M. Mahmud, J. Huang, S. Salloum and T. Emara, "A survey of data partitioning and sampling methods to support big data analysis," *Big Data Mining and Analytics*, vol. 3, no. 1, pp. 85–101, 2020.
[21] J. Gao, W. Liu, Z. Li, J. Zhang and L. Shen, "A general fragments allocation method for join query in distributed database," *Information Sciences*, vol. 512, pp. 1249–1263, 2020.
[22] X. Zhang, C. Zhang, Y. Li, R. Zhang and A. Zhou, "Apara: Workload-aware data partition and replication for parallel databases," in *Proc. Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint Int. Conf. on Web and Big Data*, Chengdu, China, pp. 191–206, 2019.
[23] R. D. Bharati and V. Z. Attar, "A comprehensive survey on distributed transactions based data partitioning," in *Proc. Int. Conf. on Computing Communication Control and Automation*, Pune, India, pp. 1–5, 2018.
[24] Z. Zhu, X. Hu, K. Huang and Y. Yuan, "Optimization of product category allocation in multiple warehouses to minimize splitting of online supermarket customer orders," *European Journal of Operational Research*, vol. 290, pp. 556–571, 2021.
[25] S. Ahirrao and R. Ingle, "Scalable transactions in cloud data stores," *Journal of Cloud Computing*, vol. 4, pp. 1–14, 2015.
[26] R. Li, C. Wang, F. Liao and H. Zhu, "RCD+: A partitioning method for data streams based on multiple queries," *IEEE Access*, vol. 8, pp. 52517–52527, 2020.
[27] A. J. Moshayedi, A. S. Roy, A. Kolahdooz and Y. Shuxin, "Deep learning application pros and cons over algorithm," *EAI Endorsed Transactions on AI and Robotics*, vol. 1, no. 1, pp. e7, 2022.
[28] E. Taati and N. Sina, "Multi-objective optimization of functionally graded materials, thickness and aspect ratio in micro-beams embedded in an elastic medium," *Structural and Multidisciplinary Optimization*, vol. 58, pp. 265–285, 2018.
[29] S. Shen, X. Wei, R. Chen, H. Chen and B. Zang, "DrTM+B: Replication-driven live reconfiguration for fast and general distributed transaction processing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 10, pp. 2628–2643, 2022.
[30] A. Jebali, S. Sassi, A. Jemai and R. Chbeir, "Secure data outsourcing in presence of the inference problem,"*Journal of Parallel and Distributed Computing*, vol. 160, pp. 1–15, 2022.
[31] http://hadoop.apache.org/hbase
[32] http://www.tpc.org/tpce